Android™

Notes for Professionals

1000+ pages

of professional hints and tricks

Contents

About

Please feel free to share this PDF with anyone for free,
latest version of this book can be downloaded from:
https://goalkicker.com/AndroidBook

This *Android™ Notes for Professionals* book is compiled from Stack Overflow Documentation, the content is written by the beautiful people at Stack Overflow. Text content is released under Creative Commons BY-SA, see credits at the end of this book whom contributed to the various chapters. Images may be copyright of their respective owners unless otherwise specified

This is an unofficial free book created for educational purposes and is not affiliated with official Android™ group(s) or company(s) nor Stack Overflow. All trademarks and registered trademarks are the property of their respective company owners

The information presented in this book is not guaranteed to be correct nor accurate, use at your own risk

Please send feedback and corrections to web@petercv.com

Chapter 1: Getting started with Android

Version	API Level	Version Code	Release Date
1.0	1	BASE	2008-09-23
1.1	2	BASE_1_1	2009-02-09
1.5	3	CUPCAKE	2009-04-27
1.6	4	DONUT	2009-09-15
2.0	5	ECLAIR	2009-10-26
2.0.1	6	ECLAIR_0_1	2009-12-03
2.1.x	7	ECLAIR_MR1	2010-01-12
2.2.x	8	FROYO	2010-05-20
2.3	9	GINGERBREAD	2010-12-06
2.3.3	10	GINGERBREAD_MR1	2011-02-09
3.0.x	11	HONEYCOMB	2011-02-22
3.1.x	12	HONEYCOMB_MR1	2011-05-10
3.2.x	13	HONEYCOMB_MR2	2011-07-15
4.0	14	ICE_CREAM_SANDWICH	2011-10-18
4.0.3	15	ICE_CREAM_SANDWICH_MR1	2011-12-16
4.1	16	JELLY_BEAN	2012-07-09
4.2	17	JELLY_BEAN_MR1	2012-11-13
4.3	18	JELLY_BEAN_MR2	2013-07-24
4.4	19	KITKAT	2013-10-31
4.4W	20	KITKAT_WATCH	2014-06-25
5.0	21	LOLLIPOP	2014-11-12
5.1	22	LOLLIPOP_MR1	2015-03-09
6.0	23	M (Marshmallow)	2015-10-05
7.0	24	N (Nougat)	2016-08-22
7.1	25	N_MR1 (Nougat MR1)	2016-10-04
8.0	26	O (Developer Preview 4)	2017-07-24

Section 1.1: Creating a New Project

Set up Android Studio

Start by setting up Android Studio and then open it. Now, you're ready to make your first Android App!

> **Note:** this guide is based on Android Studio 2.2, but the process on other versions is mainly the same.

Configure Your Project
Basic Configuration

You can start a new project in two ways:

- Click `Start a New Android Studio Project` from the welcome screen.
- Navigate to `File` → `New Project` if you already have a project open.

Next, you need to describe your application by filling out some fields:

1. **Application Name** - This name will be shown to the user.

   Example: `Hello World`. You can always change it later in `AndroidManifest.xml` file.

2. **Company Domain** - This is the qualifier for your project's package name.

   Example: `stackoverflow.com`.

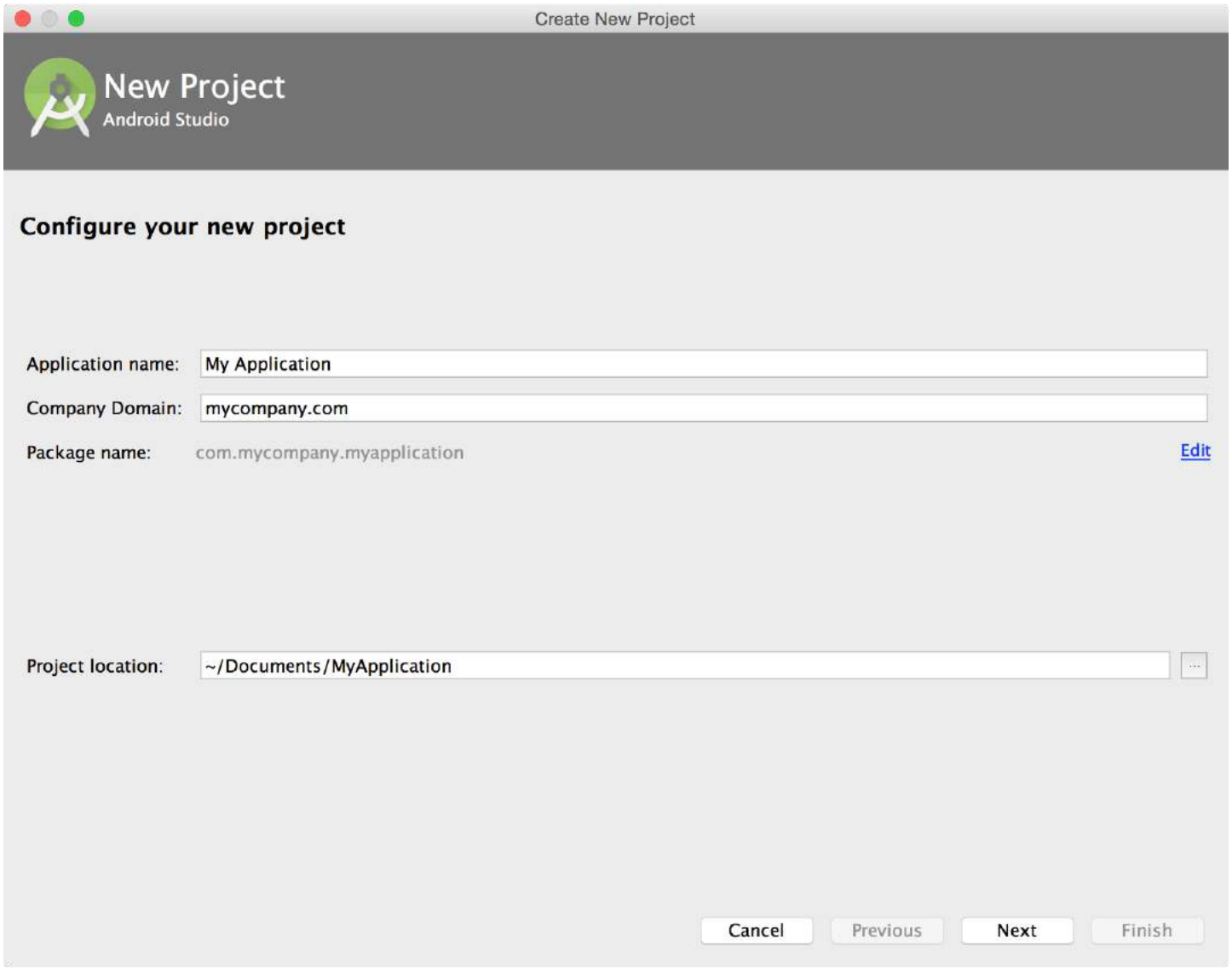
3. **Package Name** (aka `applicationId`) - This is the *fully qualified* project package name.

   It should follow *Reverse Domain Name Notation* (aka *Reverse DNS*): *Top Level Domain* **.** *Company Domain* **.** [*Company Segment* **.**] *Application Name*.

   Example: `com.stackoverflow.android.helloworld` or `com.stackoverflow.helloworld`. You can always change your *applicationId* by overriding it in your gradle file.

> Don't use the default prefix "com.example" unless you don't intend to submit your application to the Google Play Store. The package name will be your unique **applicationId** in Google Play.

4. **Project Location** - This is the directory where your project will be stored.



Select Form Factors and API Level

The next window lets you select the form factors supported by your app, such as phone, tablet, TV, Wear, and Google Glass. The selected form factors become the app modules within the project. For each form factor, you can also select the API Level for that app. To get more information, click **Help me choose**

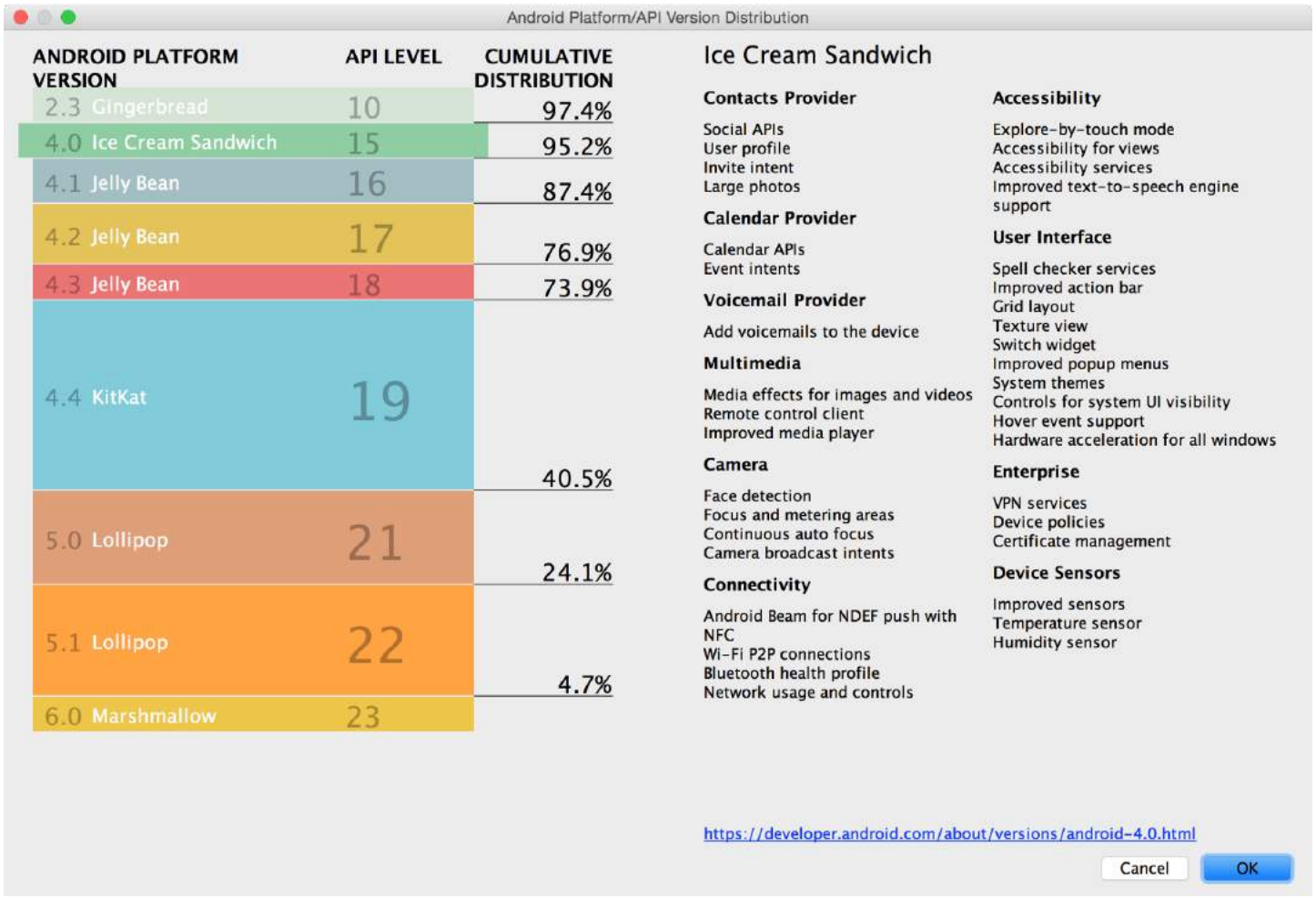
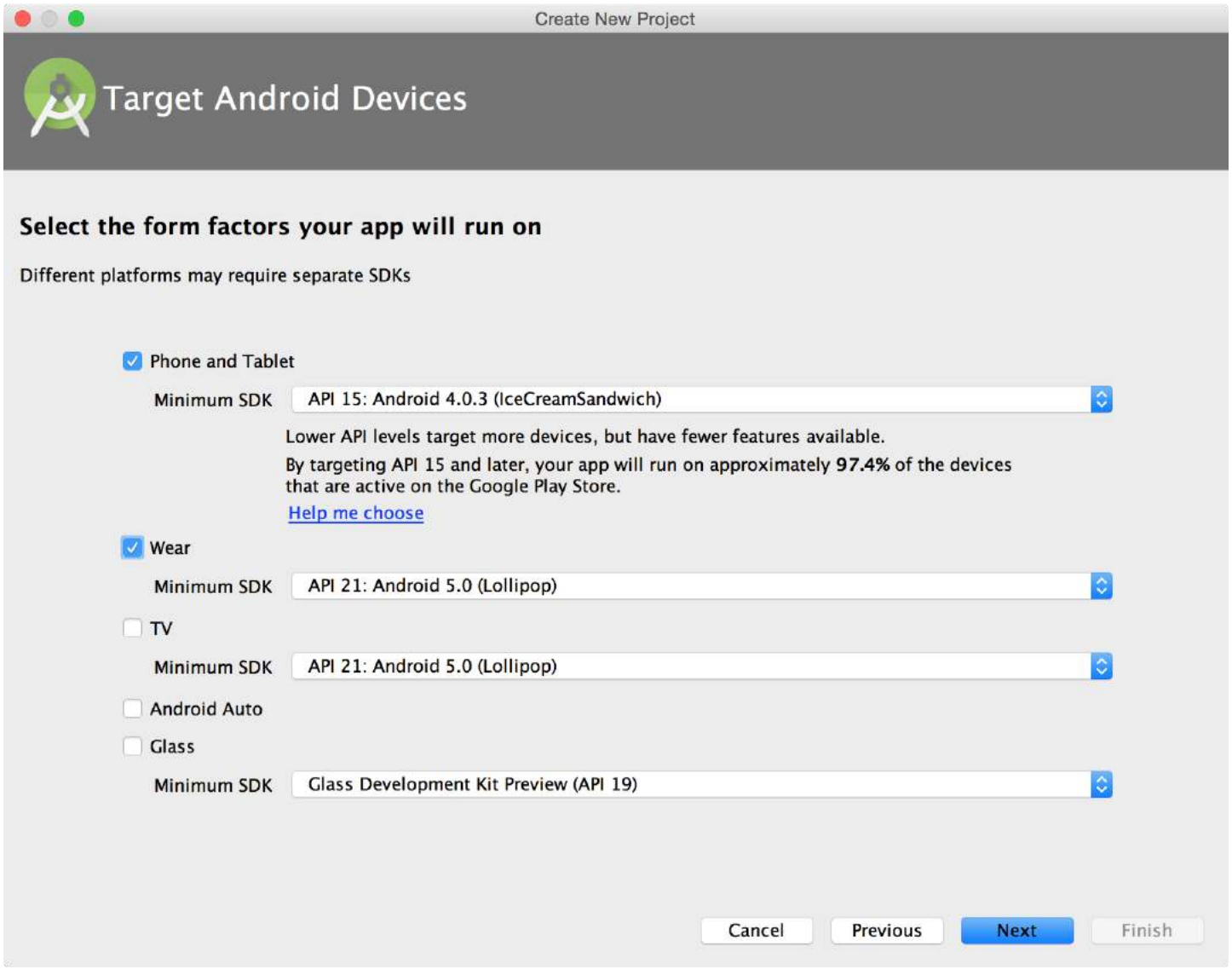


Chart of the current Android version distributions, shown when you click Help me choose.

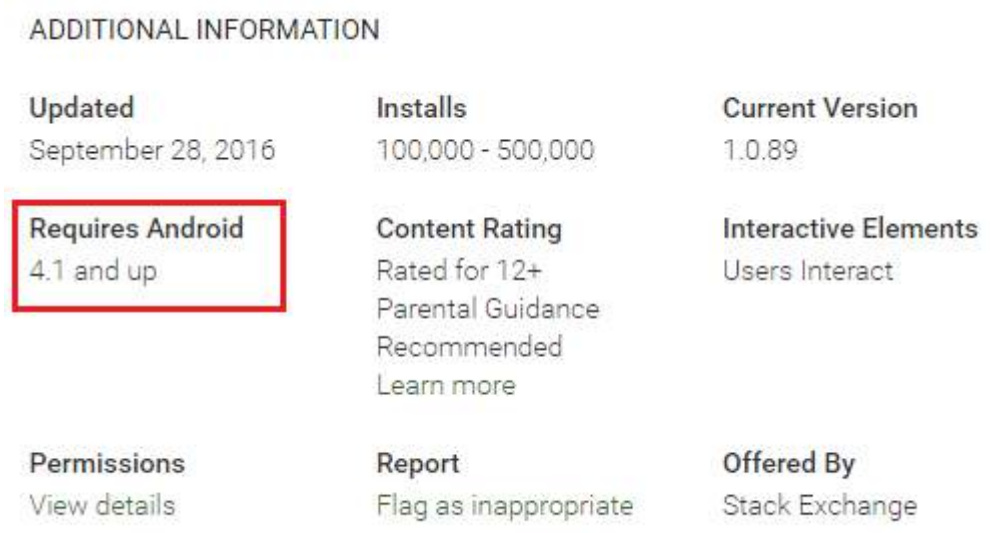
The Android Platform Distribution window shows the distribution of mobile devices running each version of Android, as shown in Figure 2. Click on an API level to see a list of features introduced in the corresponding version of Android. This helps you choose the minimum API Level that has all the features that your apps needs, so you can reach as many devices as possible. Then click **OK**.

Now, choose what platforms and version of Android SDK the application will support.



For now, select only *Phone and Tablet*.

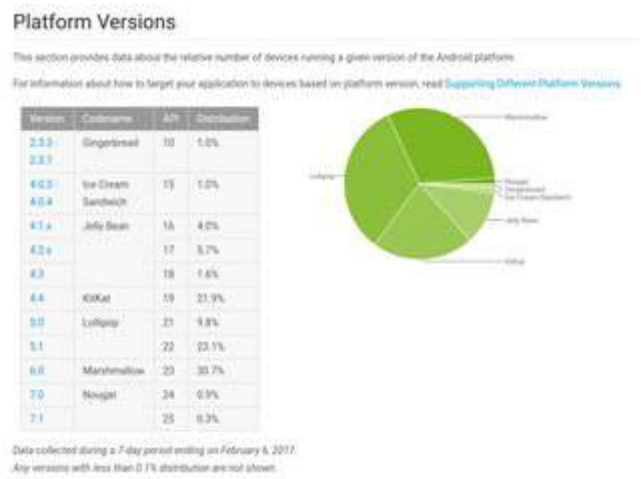
The **Minimum SDK** is the lower bound for your app. It is one of the signals the `Google Play Store` uses to determine which devices an app can be installed on. For example, Stack Exchange's app supports Android 4.1+.



Android Studio will tell you (approximately) what percentage of devices will be supported given the specified minimum SDK.

> Lower API levels target more devices but have fewer features available.

When deciding on the **Minimum SDK**, you should consider the Dashboards stats, which will give you version information about the devices that visited the Google Play Store globally in the last week.



From: Dashboards on Android Developer website.

Add an activity

Now we are going to select a default activity for our application. In Android, an `Activity` is a single screen that will be presented to the user. An application can house multiple activities and navigate between them. For this example, choose `Empty Activity` and click next.

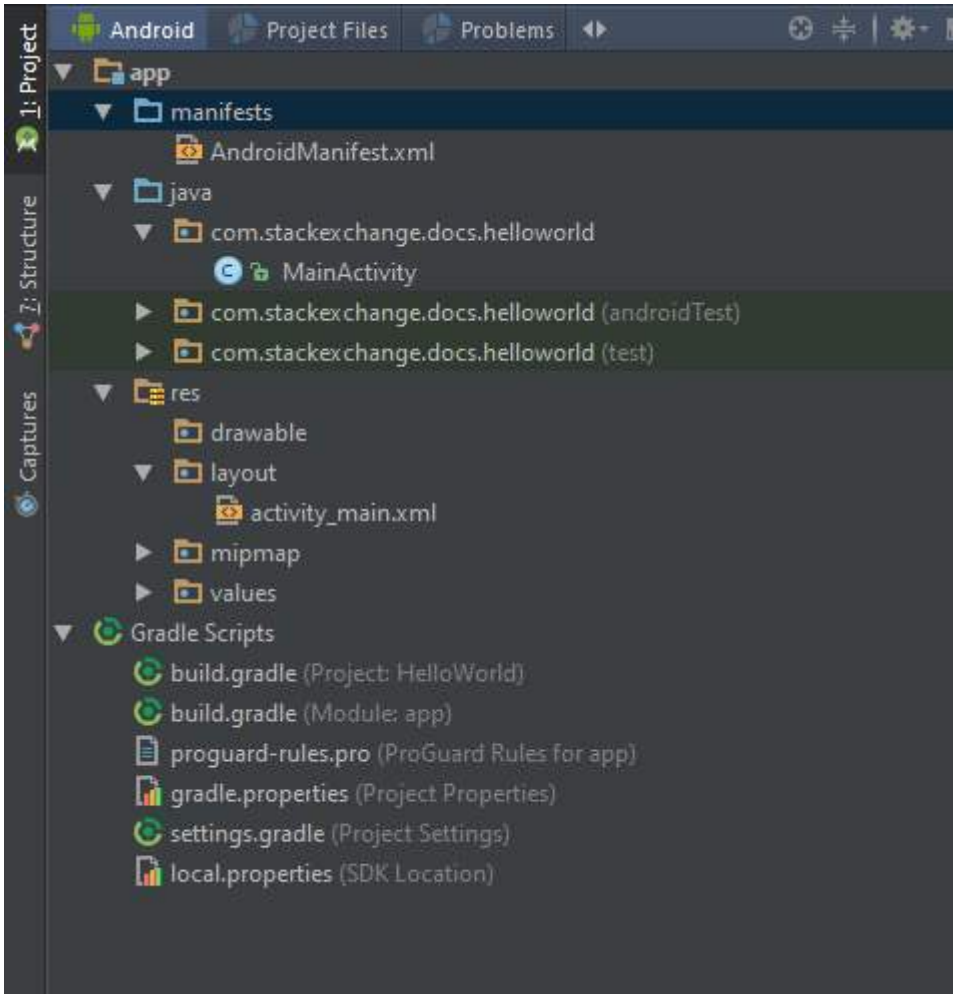
Here, if you wish, you can change the name of the activity and layout. A good practice is to keep `Activity` as a suffix for the activity name, and `activity_` as a prefix for the layout name. If we leave these as the default, Android Studio will generate an activity for us called `MainActivity`, and a layout file called `activity_main`. Now click `Finish`.

Android Studio will create and configure our project, which can take some time depending on the system.

Inspecting the Project

To understand how Android works, let's take a look at some of the files that were created for us.

On the left pane of Android Studio, we can see the structure of our Android application.



First, let's open `AndroidManifest.xml` by double clicking it. The Android manifest file describes some of the basic information about an Android application. It contains the declaration of our activities, as well as some more advanced components.

If an application needs access to a feature protected by a permission, it must declare that it requires that permission with a **`<uses-permission>`** element in the manifest. Then, when the application is installed on the device, the installer determines whether or not to grant the requested permission by checking the authorities that signed the application's certificates and, in some cases, asking the user. An application can also protect its own components (activities, services, broadcast receivers, and content providers) with permissions. It can employ any of the permissions defined by Android (listed in android.Manifest.permission) or declared by other applications. Or it can define its own.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.stackoverflow.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

```
    </application>
</manifest>
```

Next, let's open `activity_main.xml` which is located in `app/src/main/res/layout/`. This file contains declarations for the visual components of our MainActivity. You will see visual designer. This allows you to drag and drop elements onto the selected layout.

You can also switch to the xml layout designer by clicking "Text" at the bottom of Android Studio, as seen here:


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.stackexchange.docs.helloworld.MainActivity">

    <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Hello World!" />
</RelativeLayout>
```

You will see a widget called a `TextView` inside of this layout, with the `android:text` property set to "Hello World!". This is a block of text that will be shown to the user when they run the application.

You can read more about Layouts and attributes.

Next, let's take a look at `MainActivity`. This is the Java code that has been generated for `MainActivity`.

```
public class MainActivity extends AppCompatActivity {

    // The onCreate method is called when an Activity starts
```

```
    // This is where we will set up our layout
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // setContentView sets the Activity's layout to a specified XML layout
        // In our case we are using the activity_main layout
        setContentView(R.layout.activity_main);
    }
}
```

As defined in our Android manifest, `MainActivity` will launch by default when a user starts the `HelloWorld` app.

Lastly, open up the file named `build.gradle` located in `app/`.
Android Studio uses the build system **Gradle** to compile and build Android applications and libraries.

```
apply plugin: 'com.android.application'

android {
signingConfigs {
         applicationName {
             keyAlias 'applicationName'
             keyPassword 'password'
             storeFile file('../key/applicationName.jks')
             storePassword 'anotherPassword'
         }
     }
    compileSdkVersion 26
    buildToolsVersion "26.0.0"

    defaultConfig {
        applicationId "com.stackexchange.docs.helloworld"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:26.0.0'
}
```

This file contains information about the build and your app version, and you can also use it to add dependencies to external libraries. For now, let's not make any changes.

It is advisable to always select the latest version available for the dependencies:

- buildToolsVersion: 26.0.0
- com.android.support:appcompat-v7: 26.0.0 (July 2017)
- firebase: 11.0.4 (August 2017)

compileSdkVersion

`compileSdkVersion` is your way to tell *Gradle* what version of the Android SDK to compile your app with. Using the new Android SDK is a requirement to use any of the new APIs added in that level.

It should be emphasized that changing your `compileSdkVersion` does not change runtime behavior. While new compiler warnings/errors may be present when changing your `compileSdkVersion`, your `compileSdkVersion` is not included in your APK: it is purely used at compile time.

Therefore it is strongly recommended that you always compile with the latest SDK. You'll get all the benefits of new compilation checks on existing code, avoid newly deprecated APIs, and be ready to use new APIs.

minSdkVersion

If `compileSdkVersion` sets the newest APIs available to you, `minSdkVersion` is the *lower bound* for your app. The `minSdkVersion` is one of the signals the Google Play Store uses to determine which of a user's devices an app can be installed on.

It also plays an important role during development: by default lint runs against your project, warning you when you use any APIs above your `minSdkVersion`, helping you avoid the runtime issue of attempting to call an API that doesn't exist. Checking the system version at runtime is a common technique when using APIs only on newer platform versions.

targetSdkVersion

`targetSdkVersion` is the main way Android provides forward compatibility by not applying behavior changes unless the `targetSdkVersion` is updated. This allows you to use new APIs prior to working through the behavior changes. Updating to target the latest SDK should be a high priority for every app. That doesn't mean you have to use every new feature introduced nor should you blindly update your `targetSdkVersion` without testing.

targetSDKVersion is the version of Android which is the upper-limit for the available tools. If targetSDKVersion is less than 23, the app does not need to request permissions at runtime for an instance, even if the app is being run on API 23+. TargetSDKVersion does **not** prevent android versions above the picked Android version from running the app.

You can find more info about the Gradle plugin:

- A basic example
- Introduction to the Gradle plugin for android and the wrapper
- Introduction to the configuration of the build.gradle and the DSL methods

Running the Application

Now, let's run our HelloWorld application. You can either run an Android Virtual Device (which you can set up by using the AVD Manager in Android Studio, as described in the example below) or connect your own Android device through a USB cable.

Setting up an Android device

To run an application from Android Studio on your Android Device, you must enable `USB Debugging` in the `Developer Options` in the settings of your device.

```
Settings > Developer options > USB debugging
```

If `Developer Options` is not visible in the settings, navigate to `About Phone` and tap on the `Build Number` seven

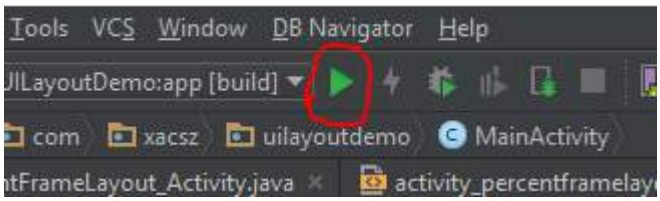
times. This will enable `Developer Options` to show up in your settings.

`Settings > About phone > Build number`

You also might need to change `build.gradle` configuration to build on a version that your device has.

Running from Android Studio

Click the green `Run` button from the toolbar at the top of Android Studio. In the window that appears, select whichever device you would like to run the app on (start an Android Virtual Device if necessary, or see Setting up an AVD (Android Virtual Device) if you need to set one up) and click `OK`.



On devices running Android 4.4 (KitKat) and possibly higher, a pop-up will be shown to authorize USB debugging. Click `OK` to accept.

The application will now install and run on your Android device or emulator.

APK file location

When you prepare your application for release, you configure, build, and test a release version of your application. The configuration tasks are straightforward, involving basic code cleanup and code modification tasks that help optimize your application. The build process is similar to the debug build process and can be done using JDK and Android SDK tools. The testing tasks serve as a final check, ensuring that your application performs as expected under real-world conditions. When you are finished preparing your application for release you have a signed APK file, which you can distribute directly to users or distribute through an application marketplace such as Google Play.

Android Studio

Since in the above examples Gradle is used, the location of the generated APK file is: `<Your Project Location>/app/build/outputs/apk/app-debug.apk`

IntelliJ

If you are a user of IntelliJ before switching to Studio, and are importing your IntelliJ project directly, then nothing changed. The location of the output will be the same under:

```
out/production/...
```

Note: this is will become deprecated sometimes around 1.0

Eclipse

If you are importing Android Eclipse project directly, do not do this! As soon as you have dependencies in your project (jars or Library Projects), this will not work and your project will not be properly setup. If you have no dependencies, then the apk would be under the same location as you'd find it in Eclipse:

```
bin/...
```

Section 1.2: Setting up Android Studio

Android Studio is the Android development IDE that is officially supported and recommended by Google. Android Studio comes bundled with the Android SDK Manager, which is a tool to download the `Android SDK` components required to start developing apps.

Installing Android Studio and `Android SDK` tools:

1. Download and install Android Studio.
2. Download the latest SDK Tools and SDK Platform-tools by opening the Android Studio, and then following the Android SDK Tool Updates instructions. You should install the latest available stable packages.

> *If you need to work on old projects that were built using older SDK versions, you may need to download these versions as well*

Since Android Studio 2.2, a copy of the latest OpenJDK comes bundled with the install and is the recommended JDK (Java Development Kit) for all Android Studio projects. This removes the requirement of having Oracle's JDK package installed. To use the bundled SDK, proceed as follows;

1. Open your project in Android Studio and select **File > Project Structure** in the menu bar.
2. In the **SDK Location** page and under **JDK location**, check the **Use embedded JDK** checkbox.
3. Click **OK**.

Configure Android Studio

Android Studio provides access to two configuration files through the **Help** menu:

- studio.vmoptions: Customize options for Studio's Java Virtual Machine (JVM), such as heap size and cache size. Note that on Linux machines this file may be named *studio64.vmoptions*, depending on your version of Android Studio.
- idea.properties: Customize Android Studio properties, such as the plugins folder path or maximum supported file size.

Change/add theme

You can change it as your preference.`File->Settings->Editor->Colors & Fonts->` and select a theme.Also you can download new themes from http://color-themes.com/ Once you have downloaded the `.jar.zip` file, go to `File -> Import Settings...` and choose the file downloaded.

Compiling Apps

Create a new project or open an existing project in Android Studio and press the green Play button  on the top toolbar to run it. If it is gray you need to wait a second to allow Android Studio to properly index some files, the progress of which can be seen in the bottom status bar.

If you want to create a project from the shell make sure that you have a `local.properties` file, which is created by Android Studio automatically. If you need to create the project without Android Studio you need a line starting with `sdk.dir=` followed by the path to your SDK installation.

Open a shell and go into the project's directory. Enter `./gradlew aR` and press enter. `aR` is a shortcut for `assembleRelease`, which will download all dependencies for you and build the app. The final APK file will be in `ProjectName/ModuleName/build/outputs/apk` and will been called `ModuleName-release.apk`.

Section 1.3: Android programming without an IDE

This is a minimalist Hello World example that uses only the most basic Android tools.

Requirements and assumptions

- Oracle JDK 1.7 or later
- Android SDK Tools (just the command line tools)

This example assumes Linux. You may have to adjust the syntax for your own platform.

Setting up the Android SDK

After unpacking the SDK release:

1. Install additional packages using the SDK manager. Don't use `android update sdk --no-ui` as instructed in the bundled Readme.txt; it downloads some 30 GB of unnecessary files. Instead use the interactive SDK manager `android sdk` to get the recommended minimum of packages.
2. Append the following JDK and SDK directories to your execution PATH. This is optional, but the instructions below assume it.
    - JDK/bin
    - SDK/platform-tools
    - SDK/tools
    - SDK/build-tools/LATEST *(as installed in step 1)*
3. Create an Android virtual device. Use the interactive AVD Manager (`android avd`). You might have to fiddle a bit and search for advice; the on-site instructions aren't always helpful.

    (You can also use your own device)

4. Run the device:

    ```
    emulator -avd DEVICE
    ```

5. If the device screen appears to be locked, then swipe to unlock it.

    Leave it running while you code the app.

Coding the app

0. Change to an empty working directory.
1. Make the source file:

    ```
    mkdir --parents src/dom/domain
    touch src/dom/domain/SayingHello.java
    ```

    Content:

    ```
    package dom.domain;
    import android.widget.TextView;
    ```

```
public final class SayingHello extends android.app.Activity
{
    protected @Override void onCreate( final android.os.Bundle activityState )
    {
        super.onCreate( activityState );
        final TextView textV = new TextView( SayingHello.this );
        textV.setText( "Hello world" );
        setContentView( textV );
    }
}
```

2. Add a manifest:

```
touch AndroidManifest.xml
```

Content:

```
<?xml version='1.0'?>
<manifest xmlns:a='http://schemas.android.com/apk/res/android'
package='dom.domain' a:versionCode='0' a:versionName='0'>
    <application a:label='Saying hello'>
        <activity a:name='dom.domain.SayingHello'>
             <intent-filter>
                <category a:name='android.intent.category.LAUNCHER'/>
                <action a:name='android.intent.action.MAIN'/>
                </intent-filter>
            </activity>
        </application>
    </manifest>
```

3. Make a sub-directory for the declared resources:

```
mkdir res
```

Leave it empty for now.

Building the code

0. Generate the source for the resource declarations. Substitute here the correct path to your **SDK**, and the installed **API** to build against (e.g. "android-23"):

```
aapt package -f \\
  -I SDK/platforms/android-API/android.jar \\
  -J src -m \\
  -M AndroidManifest.xml -S res -v
```

Resource declarations (described further below) are actually optional. Meantime the above call does nothing if res/ is still empty.

1. Compile the source code to Java bytecode (.java → .class):

```
javac \\
  -bootclasspath SDK/platforms/android-API/android.jar \\
  -classpath src -source 1.7 -target 1.7 \\
  src/dom/domain/*.java
```

2. Translate the bytecode from Java to Android (.class → .dex):

   First using Jill (.class → .jayce):

   ```
   java -jar SDK/build-tools/LATEST/jill.jar \\
     --output classes.jayce src
   ```

   Then Jack (.jayce → .dex):

   ```
   java -jar SDK/build-tools/LATEST/jack.jar \\
     --import classes.jayce --output-dex .
   ```

   Android bytecode used to be called "Dalvik executable code", and so "dex".

   You could replace steps 11 and 12 with a single call to Jack if you like; it can compile directly from Java source (.java → .dex). But there are advantages to compiling with `javac`. It's a better known, better documented and more widely applicable tool.

3. Package up the resource files, including the manifest:

   ```
   aapt package -f \\
     -F app.apkPart \\
     -I SDK/platforms/android-API/android.jar \\
     -M AndroidManifest.xml -S res -v
   ```

   That results in a partial APK file (Android application package).

4. Make the full APK using the `ApkBuilder` tool:

   ```
   java -classpath SDK/tools/lib/sdklib.jar \\
     com.android.sdklib.build.ApkBuilderMain \\
     app.apkUnalign \\
     -d -f classes.dex -v -z app.apkPart
   ```

   It warns, "THIS TOOL IS DEPRECATED. See --help for more information." If `--help` fails with an `ArrayIndexOutOfBoundsException`, then instead pass no arguments:

   ```
   java -classpath SDK/tools/lib/sdklib.jar \\
     com.android.sdklib.build.ApkBuilderMain
   ```

   It explains that the CLI (`ApkBuilderMain`) is deprecated in favour of directly calling the Java API (`ApkBuilder`). (If you know how to do that from the command line, please update this example.)

5. Optimize the data alignment of the APK (recommended practice):

   ```
   zipalign -f -v 4 app.apkUnalign app.apk
   ```

Installing and running

0. Install the app to the Android device:

```
adb install -r app.apk
```

1. Start the app:

   ```
   adb shell am start -n dom.domain/.SayingHello
   ```

   It should run and say hello.

That's all. That's what it takes to say hello using the basic Android tools.

Declaring a resource

This section is optional. Resource declarations aren't required for a simple "hello world" app. If they aren't required for your app either, then you could streamline the build somewhat by omitting step 10, and removing the reference to the res/ directory from step 13.

Otherwise, here's a brief example of how to declare a resource, and how to reference it.

0. Add a resource file:

   ```
   mkdir res/values
   touch res/values/values.xml
   ```

   Content:

   ```
   <?xml version='1.0'?>
   <resources>
       <string name='appLabel'>Saying hello</string>
   </resources>
   ```

1. Reference the resource from the XML manifest. This is a declarative style of reference:

   ```
   <!-- <application a:label='Saying hello'> -->
        <application a:label='@string/appLabel'>
   ```

2. Reference the same resource from the Java source. This is an imperative reference:

   ```
   // v.setText( "Hello world" );
      v.setText( "This app is called "
        + getResources().getString( R.string.appLabel ));
   ```

3. Test the above modifications by rebuilding, reinstalling and re-running the app (steps 10-17).

   It should restart and say, "This app is called Saying hello".

Uninstalling the app

```
adb uninstall dom.domain
```

See also

- original question - The original question that prompted this example
- working example - A working build script that uses the above commands

Section 1.4: Application Fundamentals

Android Apps are written in Java. The Android SDK tools compile the code, data and resource files into an APK (Android package). Generally, one APK file contains all the content of the app.

Each app runs on its own virtual machine(VM) so that app can run isolated from other apps. Android system works with the principle of least privilege. Each app only has access to the components which it requires to do its work, and no more. However, there are ways for an app to share data with other apps, such as by sharing Linux user id between app, or apps can request permission to access device data like SD card, contacts etc.

App Components

App components are the building blocks of an Android app. Each components plays a specific role in an Android app which serves a distinct purpose and has distinct life-cycles(the flow of how and when the component is created and destroyed). Here are the four types of app components:

1. **Activities:** An activity represents a single screen with a User Interface(UI). An Android app may have more than one activity. (e.g. an email app might have one activity to list all the emails, another to show the contents of each email, and another to compose new email.) All the activities in an App work together to create a User eXperience (UX).
2. **Services:** A service runs in the background to perform long-running operations or to perform work for a remote processes. A service does not provide any UI, it runs only in the background with the User's input. (e.g. a service can play music in the background while the user is in a different App, or it might download data from the internet without blocking user's interaction with the Android device.)
3. **Content Providers:** A content provider manages shared app data. There are four ways to store data in an app: it can be written to a file and stored in the file system, inserted or updated to a SQLite database, posted to the web, or saved in any other persistent storage location the App can access. Through content providers, other Apps can query or even modify the data. (e.g. Android system provides a content provider that manages the user's contact information so that any app which has permission can query the contacts.) Content providers can also be used to save the data which is private to the app for better data integrity.
4. **Broadcast receivers:** A broadcast receiver responds to the system-wide broadcasts of announcements (e.g. a broadcast announcing that the screen has turned off, the battery is low, etc.) or from Apps (e.g. to let other apps know that some data has been downloaded to the device and is available for them to use). Broadcast receivers don't have UIs but they can show notification in the status bar to alert the user. Usually broadcast receivers are used as a gateway to other components of the app, consisting mostly of activities and services.

One unique aspect of the Android system is that any app can start another app's component (e.g. if you want to make call, send SMS, open a web page, or view a photo, there is an app which already does that and your app can make use of it, instead of developing a new activity for the same task).

When the system starts a component, it starts the process for that app (if it isn't already running, i.e. only one foreground process per app can run at any given time on an Android system) and instantiates the classes needed for that component. Thus the component runs on the process of that App that it belongs to. Therefore, unlike apps on other systems, Android apps don't have a single entry point(there is no `main()` method).

Because the system runs each app in a separate process, one app cannot directly activate another app's components, however the Android system can. Thus to start another app's component, one app must send a message to the system that specifies an intent to start that component, then the system will start that component.

Context

Instances of the class `android.content.Context` provide the connection to the Android system which executes the application. Instance of Context is required to get access to the resources of the project and the global information

about the app's environment.

Let's have an easy to digest example: Consider you are in a hotel, and you want to eat something. You call room-service and ask them to bring you things or clean up things for you. Now think of this hotel as an Android app, yourself as an activity, and the room-service person is then your context, which provides you access to the hotel resources like room-service, food items etc.

Yet an other example, You are in a restaurant sitting on a table, each table has an attendant, when ever you want to order food items you ask the attendant to do so. The attendant then places your order and your food items gets served on your table. Again in this example, the restaurant is an Android App, the tables or the customers are App components, the food items are your App resources and the attendant is your context thus giving you a way to access the resources like food items.

Activating any of the above components requires the context's instance. Not just only the above, but almost every system resource: creation of the UI using views(discussed later), creating instance of system services, starting new activities or services -- all require context.

More detailed description is written here.

Section 1.5: Setting up an AVD (Android Virtual Device)

TL;DR It basically allows us to simulate real devices and test our apps without a real device.

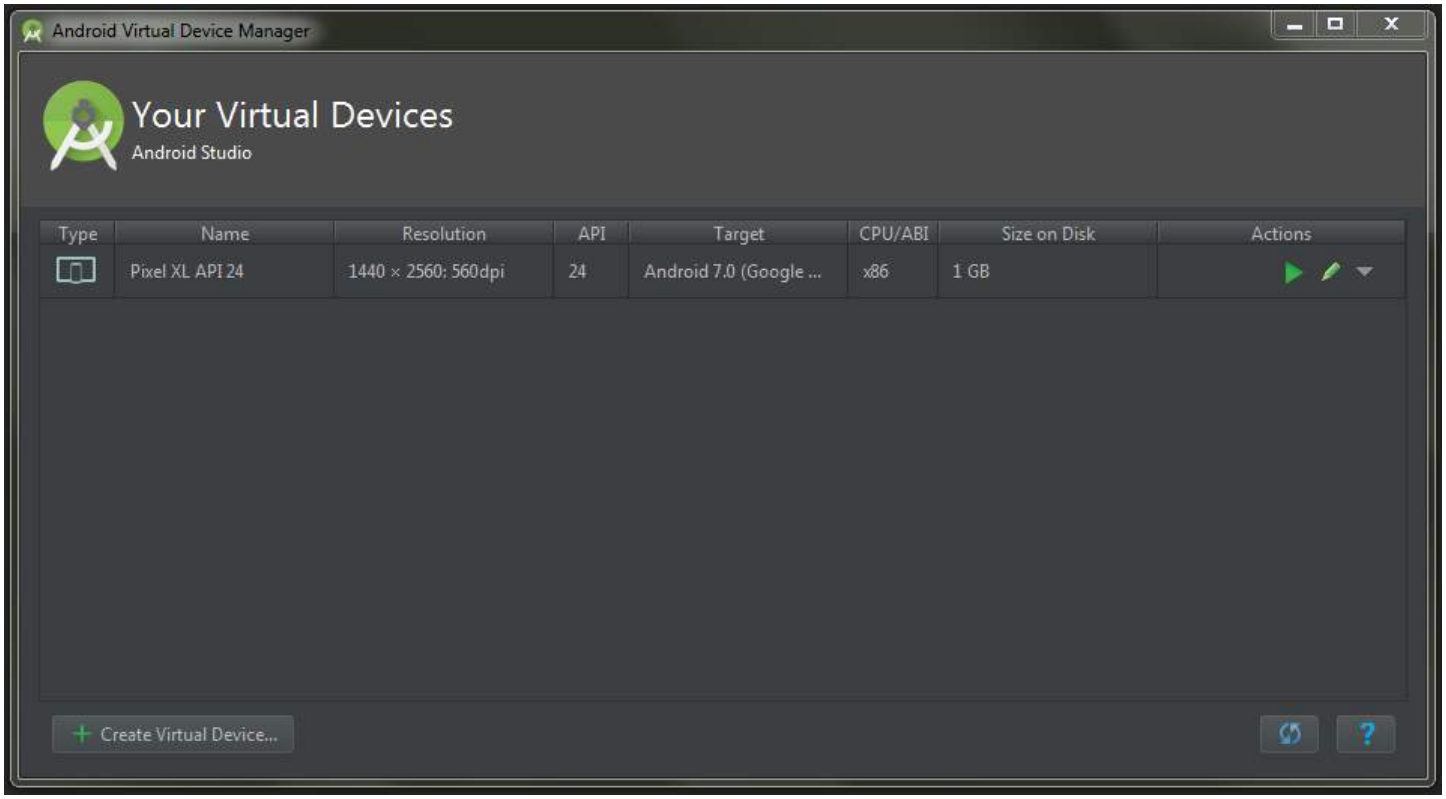
According to Android Developer Documentation,

> an ***Android Virtual Device (AVD)*** *definition* lets you define the characteristics of an Android Phone, Tablet, Android Wear, or Android TV device that you want to simulate in the Android Emulator. The AVD Manager helps you easily create and manage AVDs.

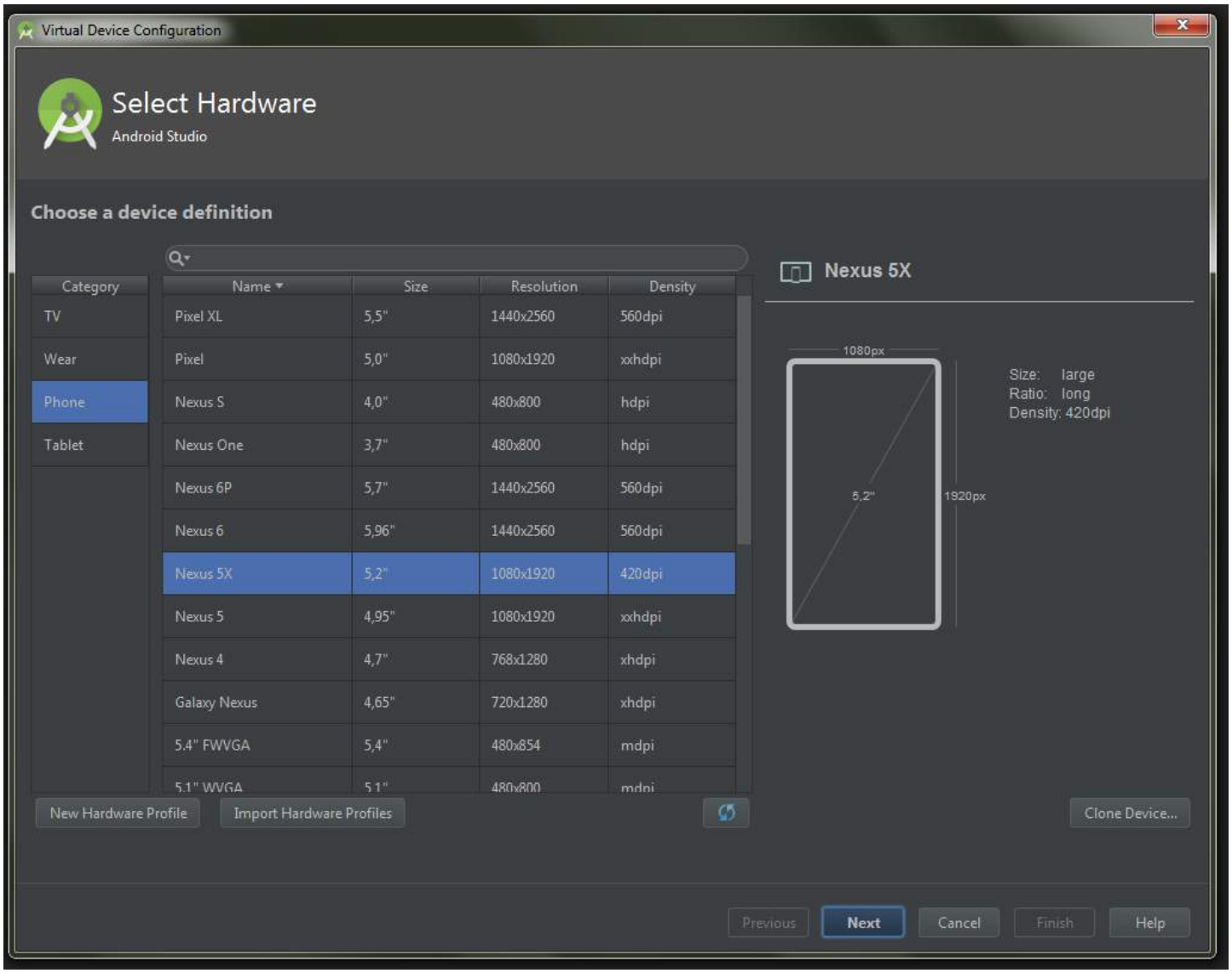
To set up an AVD, follow these steps:

1. Click this button to bring up the AVD Manager:

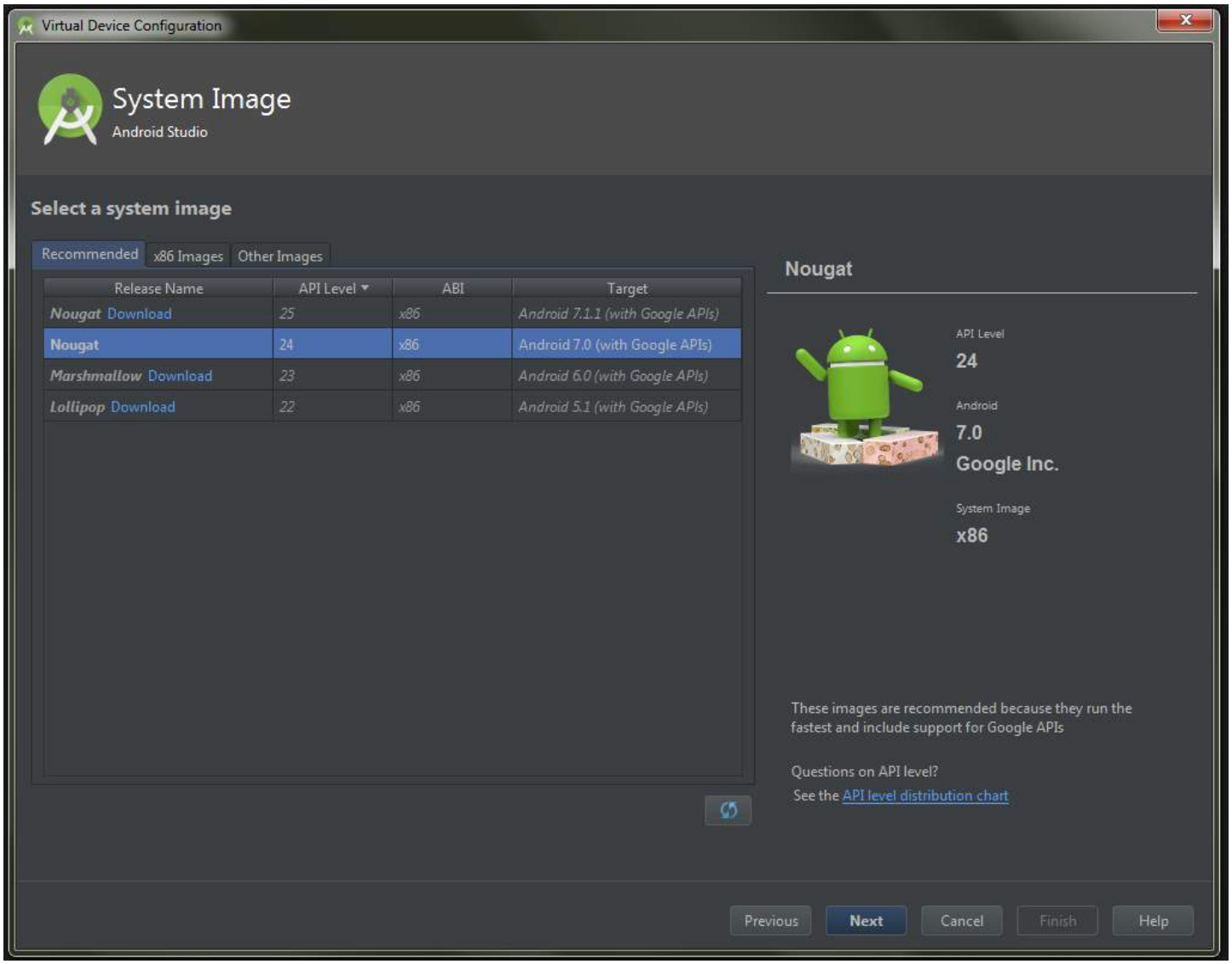
2. You should see a dialog like this:



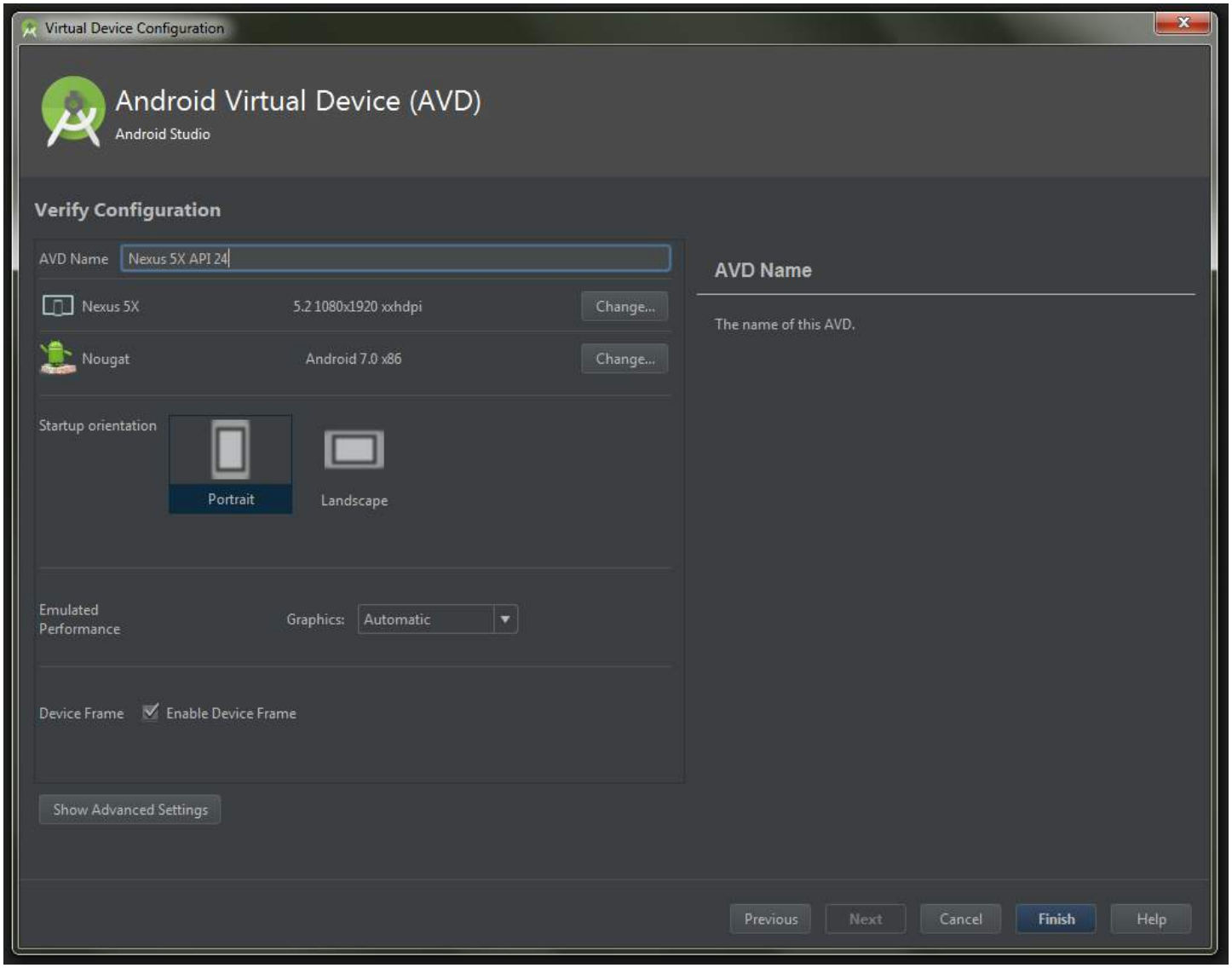
3. Now click the `+ Create Virtual Device...` button. This will bring up Virtual Device Configuration Dialog:



4. Select any device you want, then click `Next`:



5. Here you need to choose an Android version for your emulator. You might also need to download it first by clicking `Download`. After you've chosen a version, click `Next`.



6. Here, enter a name for your emulator, initial orientation, and whether you want to display a frame around it. After you chosen all these, click `Finish`.

7. Now you got a new AVD ready for launching your apps on it.

Chapter 2: Android Studio

Section 2.1: Setup Android Studio

System Requirements

- Microsoft® Windows® 8/7/Vista/2003 (32 or 64-bit).
- Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks)
- GNOME or KDE desktop

Installation

Window

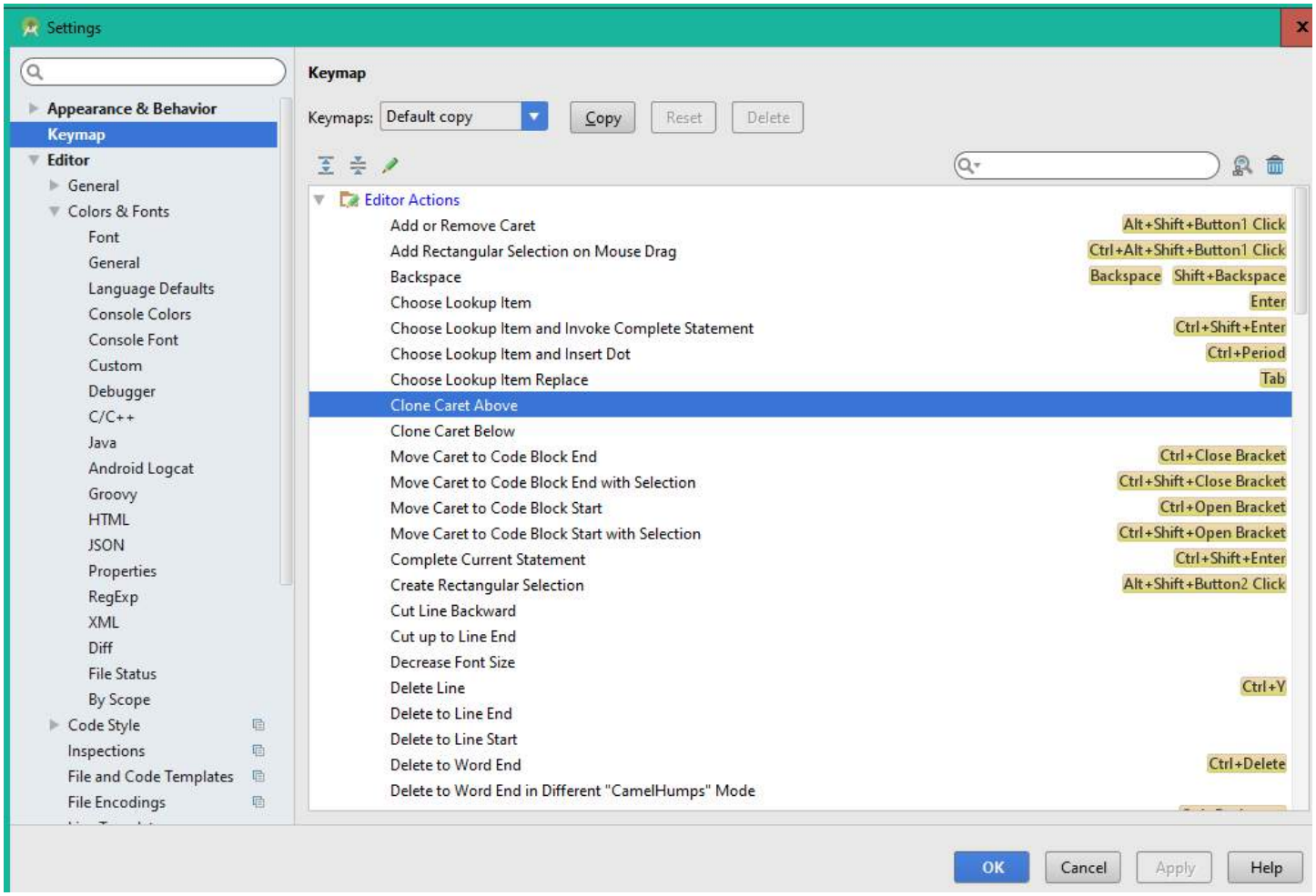
1. Download and install JDK (Java Development Kit) version 8
2. Download Android Studio
3. Launch `Android Studio.exe` then mention JDK path and download the latest SDK

Linux

1. Download and install JDK (Java Development Kit) version 8
2. Download Android Studio
3. Extract the zip file
4. Open terminal, cd to the extracted folder, cd to bin (example `cd android-studio/bin`)
5. Run ./studio.sh

Section 2.2: View And Add Shortcuts in Android Studio

By going to Settings >> Keymap A window will popup showing All the `Editor Actions` with the their name and shortcuts. Some of the `Editor Actions` do not have shortcuts. So right click on that and add a new shortcut to that. Check the image below



Section 2.3: Android Studio useful shortcuts

The following are some of the more common/useful shortcuts.

These are based on the default IntelliJ shortcut map. You can switch to other common IDE shortcut maps via `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>`

Action	Shortcut
Format code	CTRL + ALT + L
Add unimplemented methods	CTRL + I
Show logcat	ALT + 6
Build	CTRL + F9
Build and Run	CTRL + F10
Find	CTRL + F
Find in project	CTRL + SHIFT + F
Find and replace	CTRL + R
Find and replace in project	CTRL + SHIFT + R
Override methods	CTRL + O
Show project	ALT + 1
Hide project - logcat	SHIFT + ESC
Collapse all	CTRL + SHIFT + NumPad +
View Debug Points	CTRL + SHIFT + F8
Expand all	CTRL + SHIFT + NumPad -
Open Settings	ALT + s

Select Target (open current file in Project view)	ALT + F1 → ENTER
Search Everywhere	SHIFT → SHIFT (Double shift)
Code | Surround With	CTRL → ALT + T
Create method form selected code	ALT + CTRL

Refactor:

Action	Shortcut
Refactor This (menu/picker for all applicable refactor actions of the current element)	Mac CTRL + T - Win/Linux CTRL + ALT + T
Rename	SHIFT + F6
Extract Method	Mac CMD + ALT + M - Win/Linux CTRL + ALT + M
Extract Parameter	Mac CMD + ALT + P - Win/Linux CTRL + ALT + P
Extract Variable	Mac CMD + ALT + V - Win/Linux CTRL + ALT + V

Section 2.4: Android Studio Improve performance tip

Enable Offline Work:

1. Click File -> Settings. Search for "gradle" and click in `Offline work` box.
2. Go to Compiler (in same settings dialog just below `Gradle`) and add `--offline` to `Command-line Options` text box.

Improve Gradle Performance

Add following two line of code in your gradle.properties file.

```
org.gradle.daemon=true
org.gradle.parallel=true
```

Increasing the value of `-Xmx` and `-Xms` in `studio.vmoptions` file

```
-Xms1024m
-Xmx4096m
-XX:MaxPermSize=1024m
-XX:ReservedCodeCacheSize=256m
-XX:+UseCompressedOops
```

Window

> %USERPROFILE%.{FOLDER_NAME}\studio.exe.vmoptions and/or
> %USERPROFILE%.{FOLDER_NAME}\studio64.exe.vmoptions

Mac

> ~/Library/Preferences/{FOLDER_NAME}/studio.vmoptions

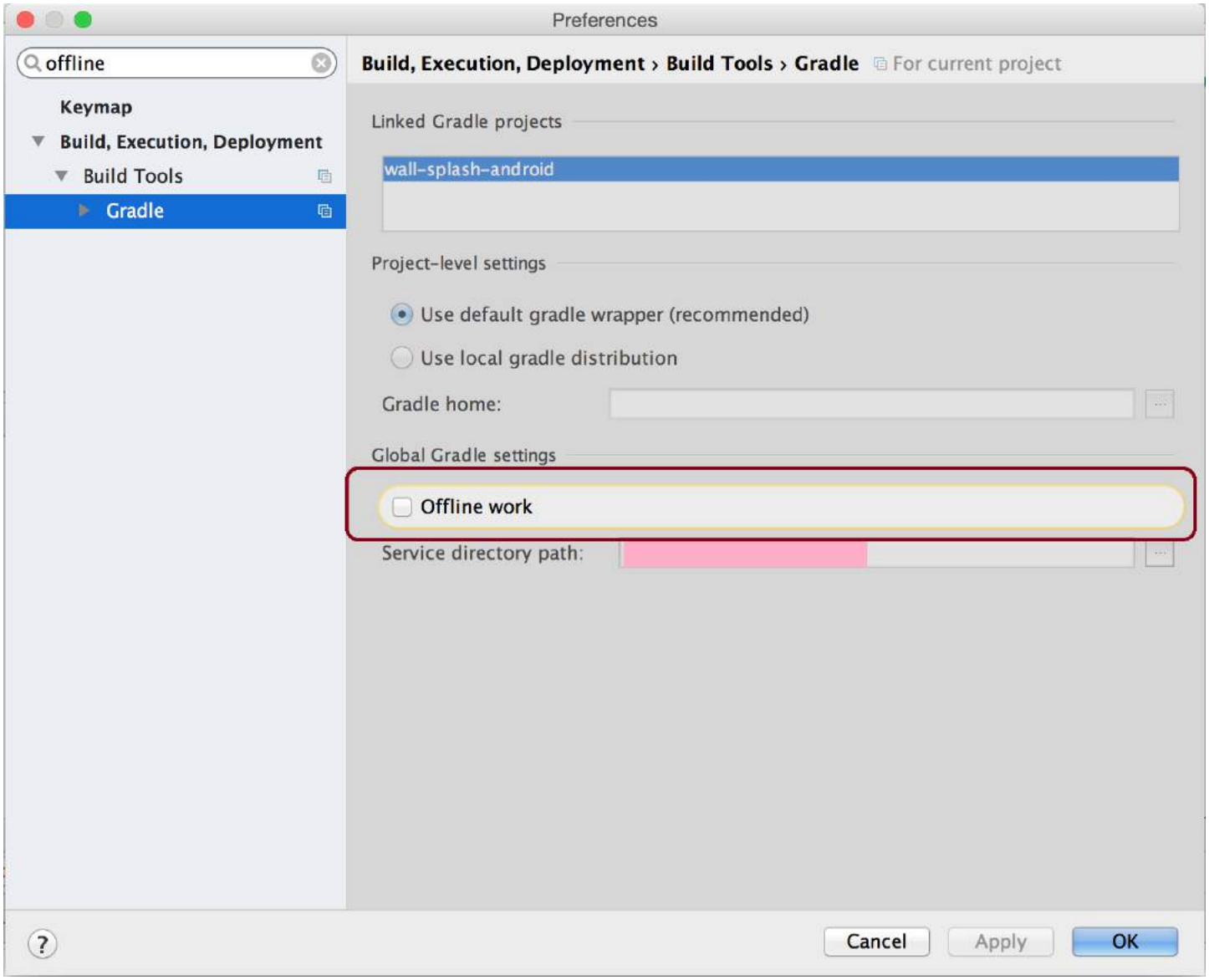
Linux

> ~/.{FOLDER_NAME}/studio.vmoptions and/or ~/.{FOLDER_NAME}/studio64.vmoptions

Section 2.5: Gradle build project takes forever

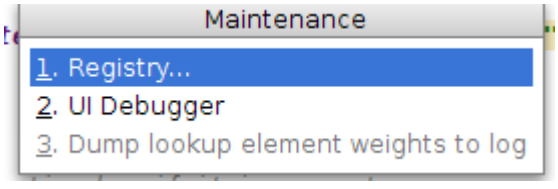
Android Studio -> **Preferences** -> **Gradle** -> Tick **Offline work** and then restart your Android studio.

Reference screenshot:

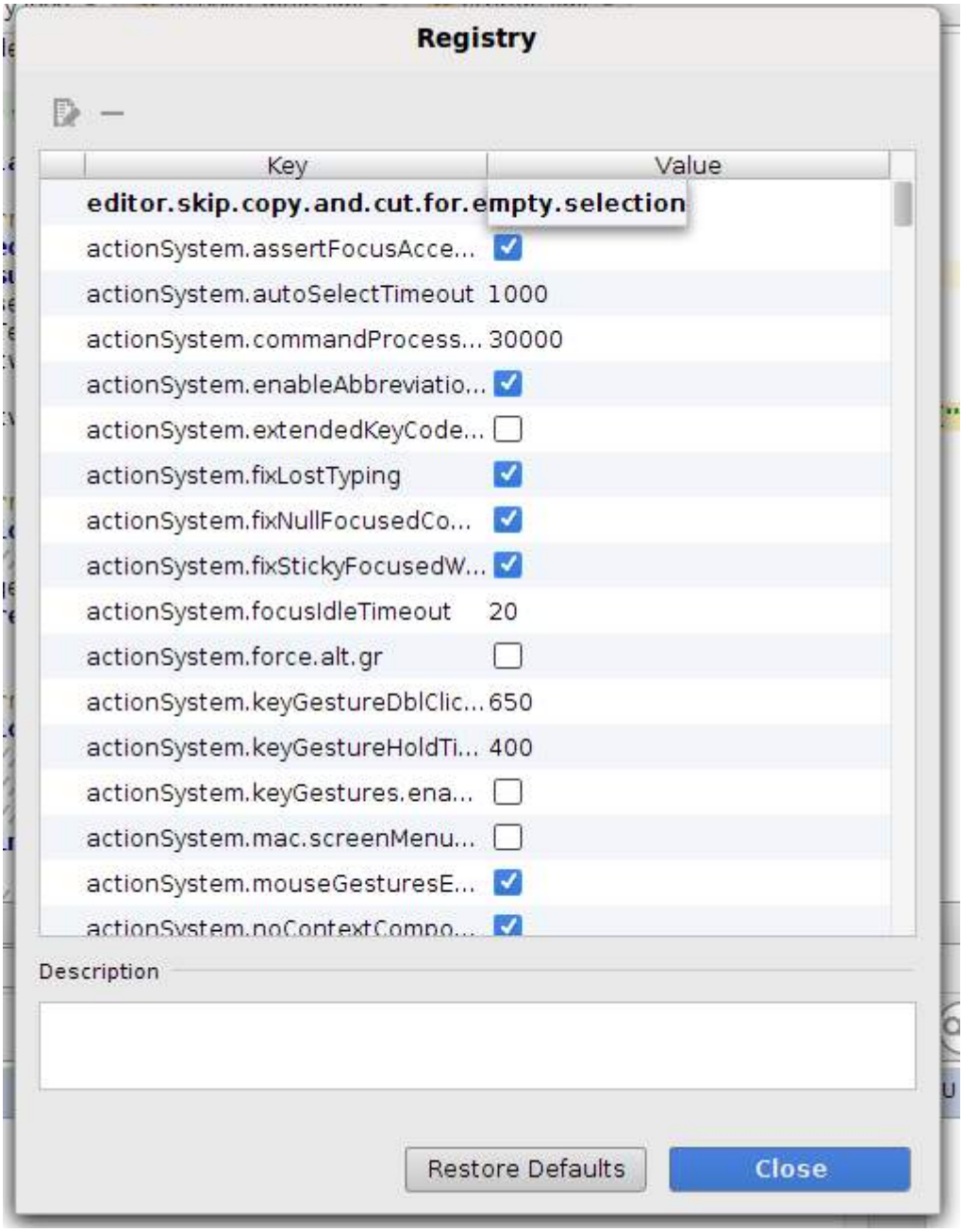


Section 2.6: Enable/Disable blank line copy

`ctrl + alt + shift + /` (`cmd + alt + shift + /` on `MacOS`) should show you the following dialog:



Clicking on `Registry` you will get



The key you want to enable/disable is

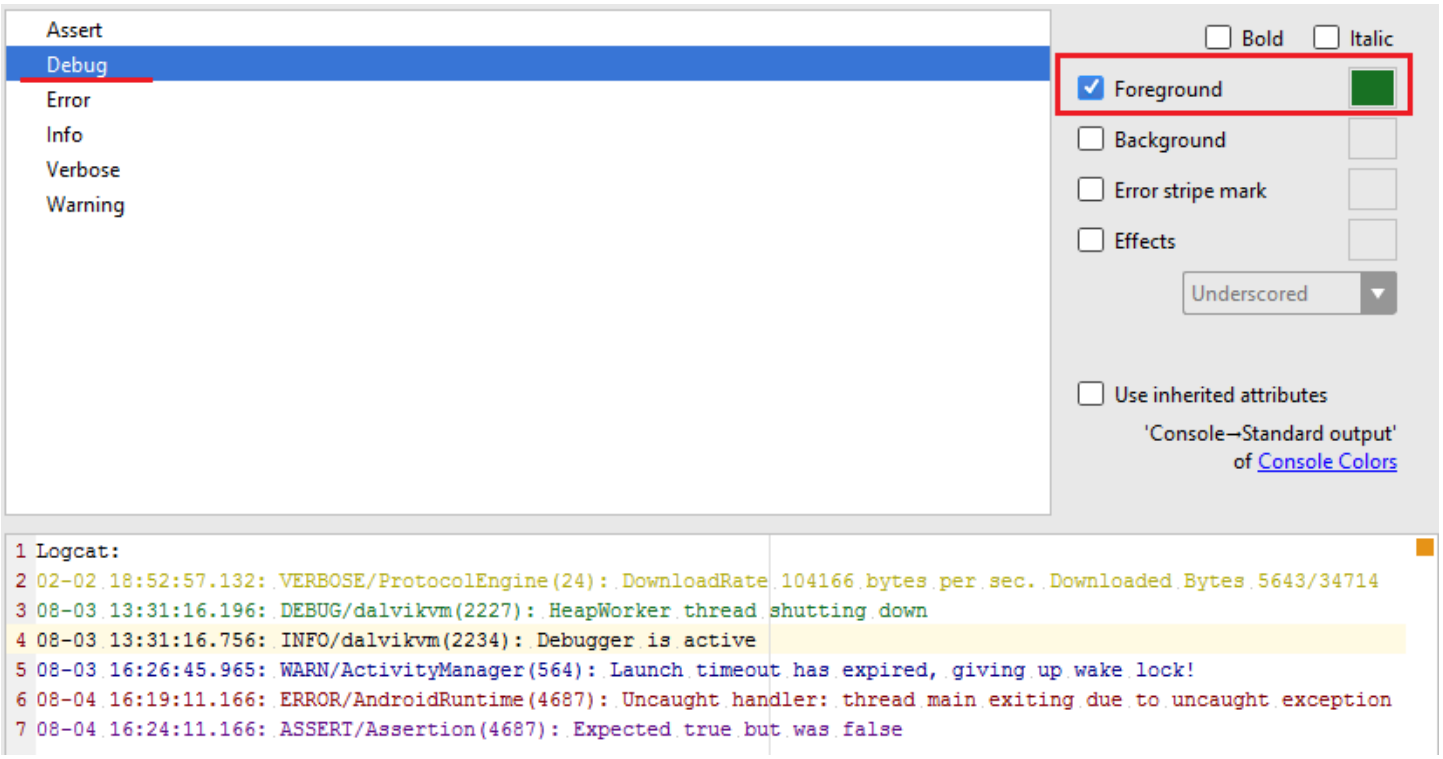
```
editor.skip.copy.and.cut.for.empty.selection
```

Tested on `Linux Ubuntu` and `MacOS`.

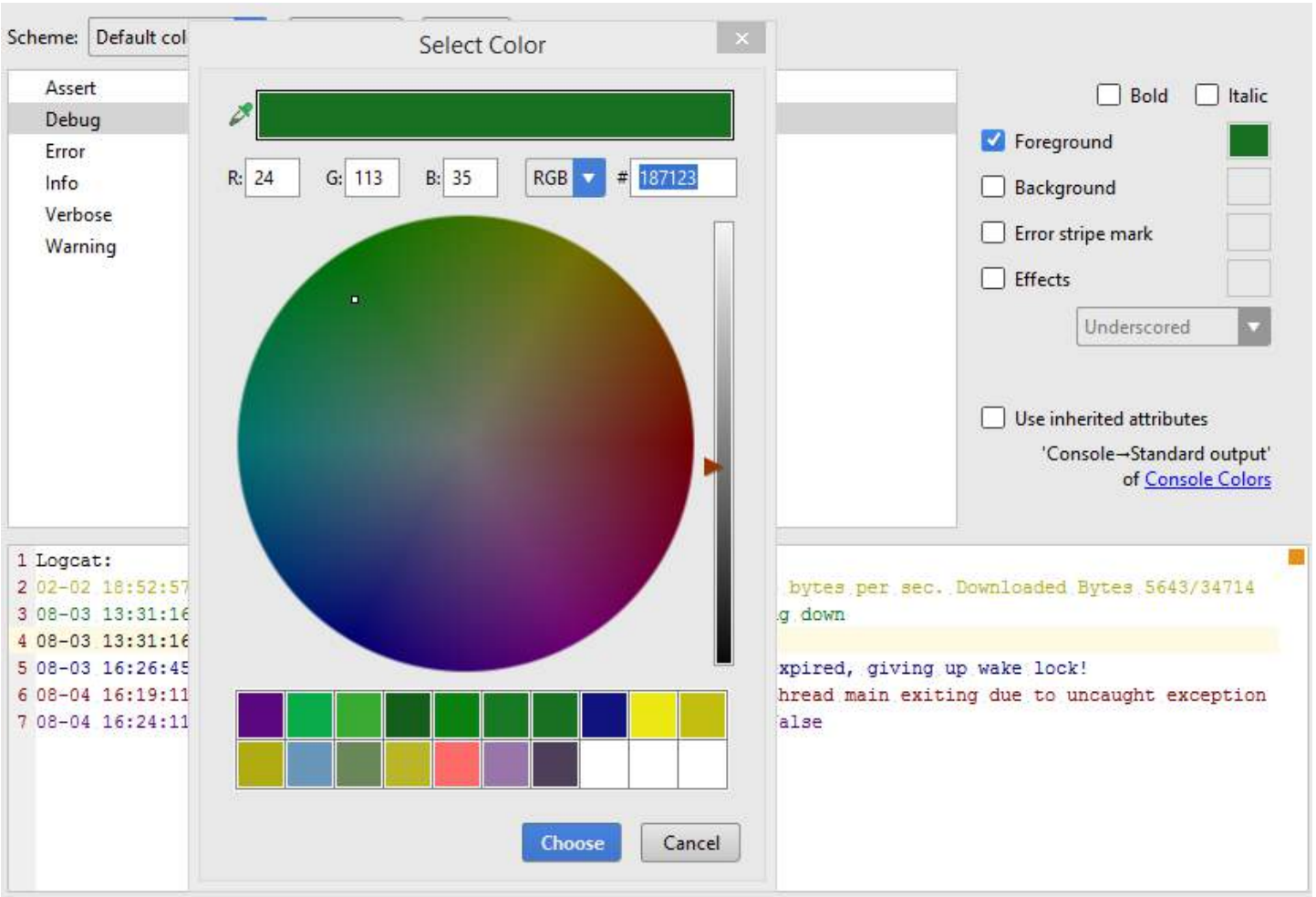
Section 2.7: Custom colors of logcat message based on message importance

Go to File -> Settings -> Editor -> Colors & Fonts -> Android Logcat

Change the colors as you need:



Choose the appropriate color:



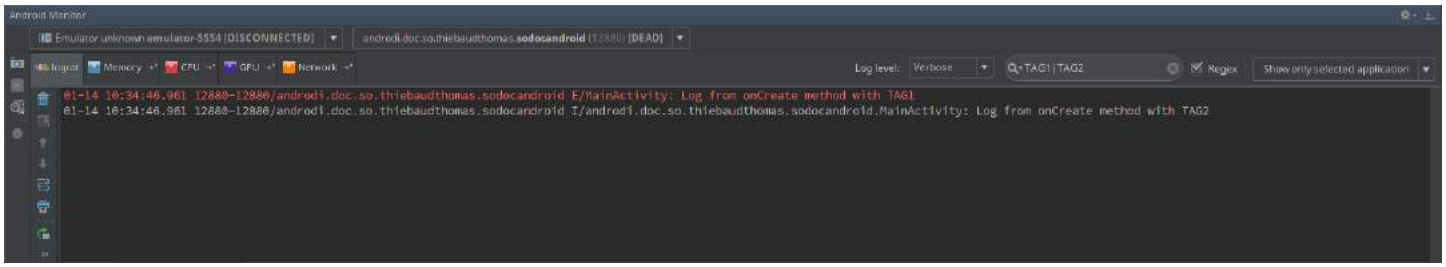
Section 2.8: Filter logs from UI

Android logs can be filtered directly from the UI. Using this code

```
public class MainActivity extends AppCompatActivity {
    private final static String TAG1 = MainActivity.class.getSimpleName();
    private final static String TAG2 = MainActivity.class.getCanonicalName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.e(TAG1,"Log from onCreate method with TAG1");
        Log.i(TAG2,"Log from onCreate method with TAG2");
    }
}
```

If I use the regex `TAG1|TAG2` and the level `verbose` I get

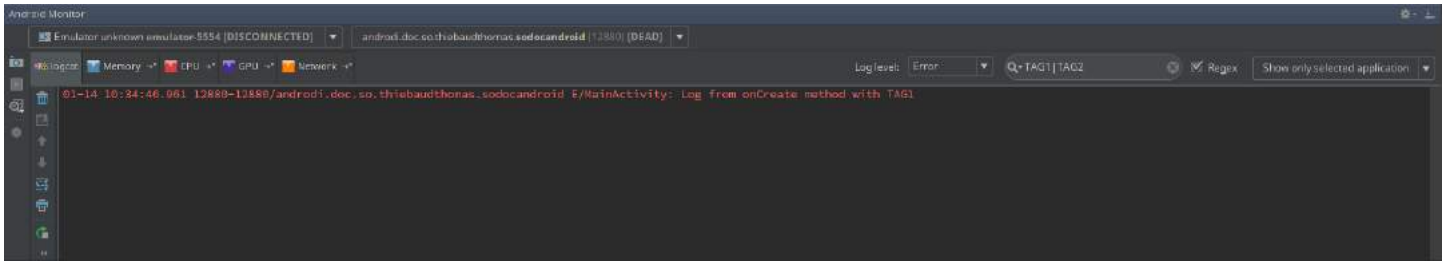
```
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log from
onCreate method with TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: Log from onCreate method with TAG2
```



The level can be set to get logs with given level and above. For example the `verbose` level will catch `verbose`, `debug`, `info`, `warn`, `error` and **`assert`** logs.

Using the same example, if I set the level to `error`, I only get

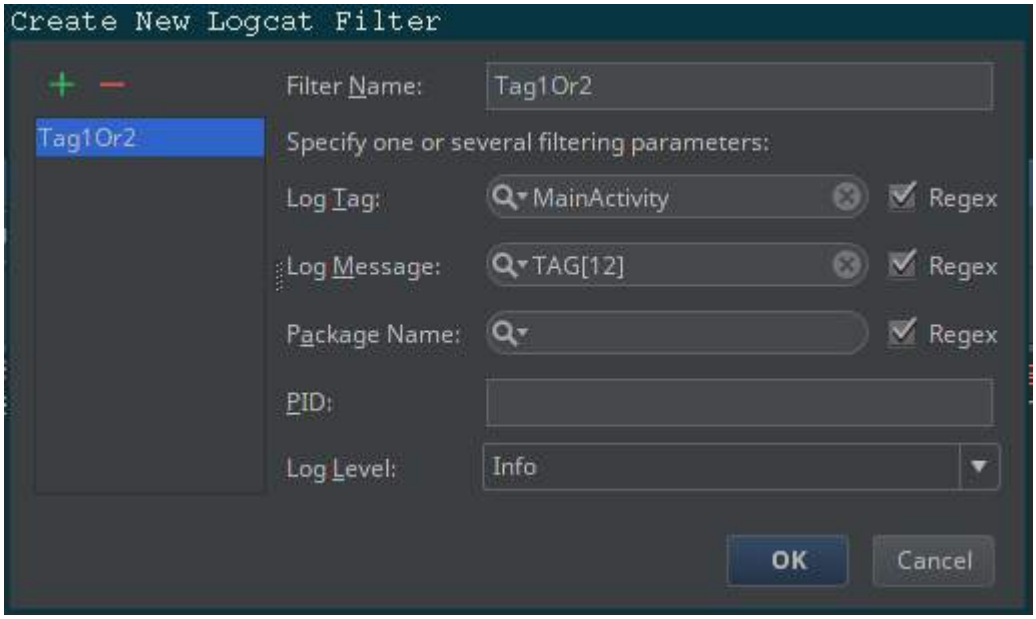
```
01-14 10:34:46.961 12880-12880/androdi.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log from
onCreate method with TAG1
```



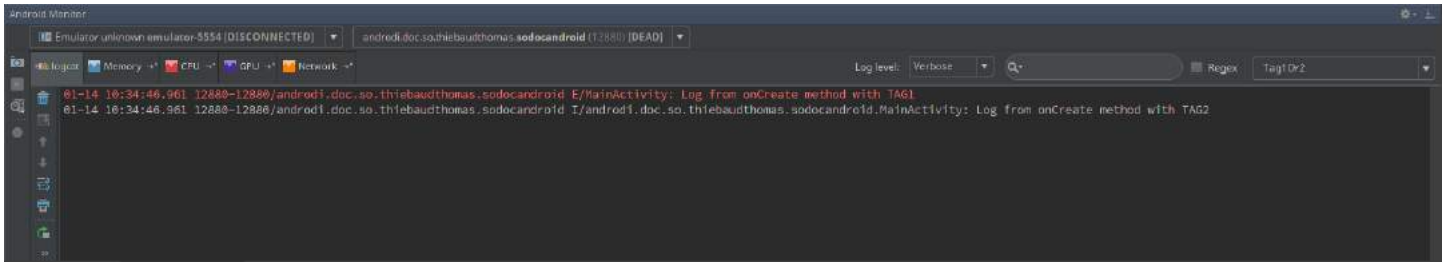
Section 2.9: Create filters configuration

Custom filters can be set and save from the UI. In the `AndroidMonitor` tab, click on the right dropdown (must contains `Show only selected application` or `No filters`) and select `Edit filter configuration`.

Enter the filter you want

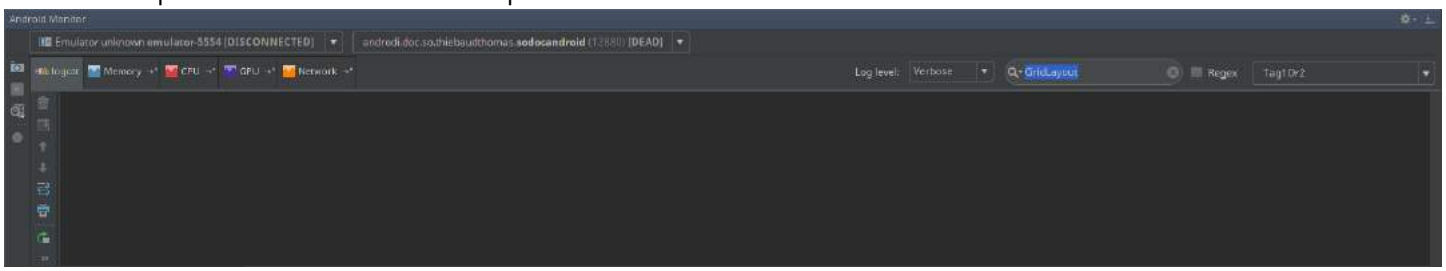


And use it (you can selected it from the same dropdown)

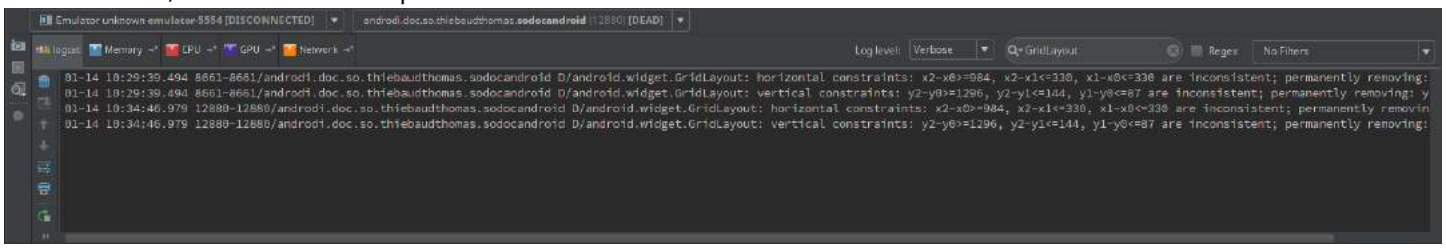


Important If you add an input in the filter bar, android studio will consider both your filter and your input.

With both input and filter there is no output

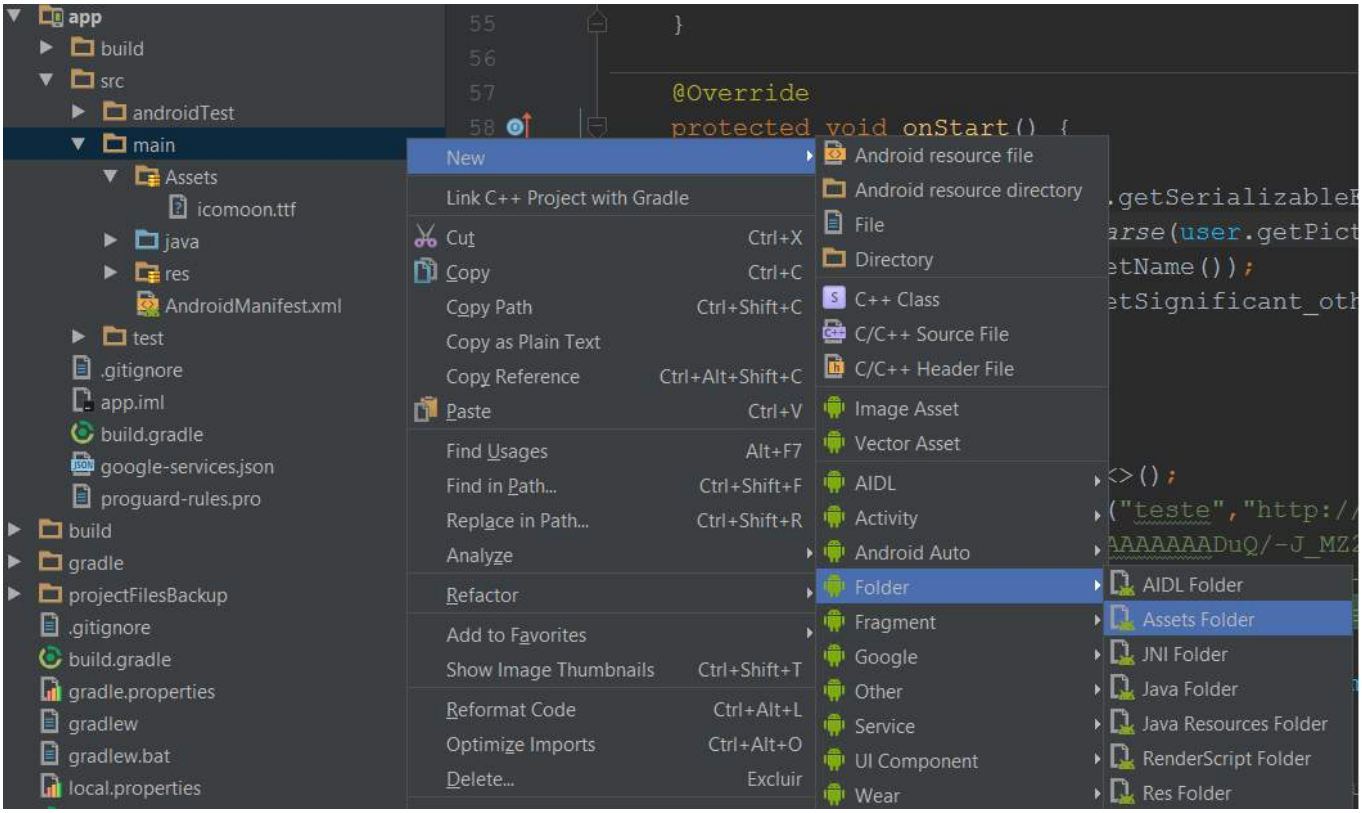


Without filter, there is some outputs



Section 2.10: Create assets folder

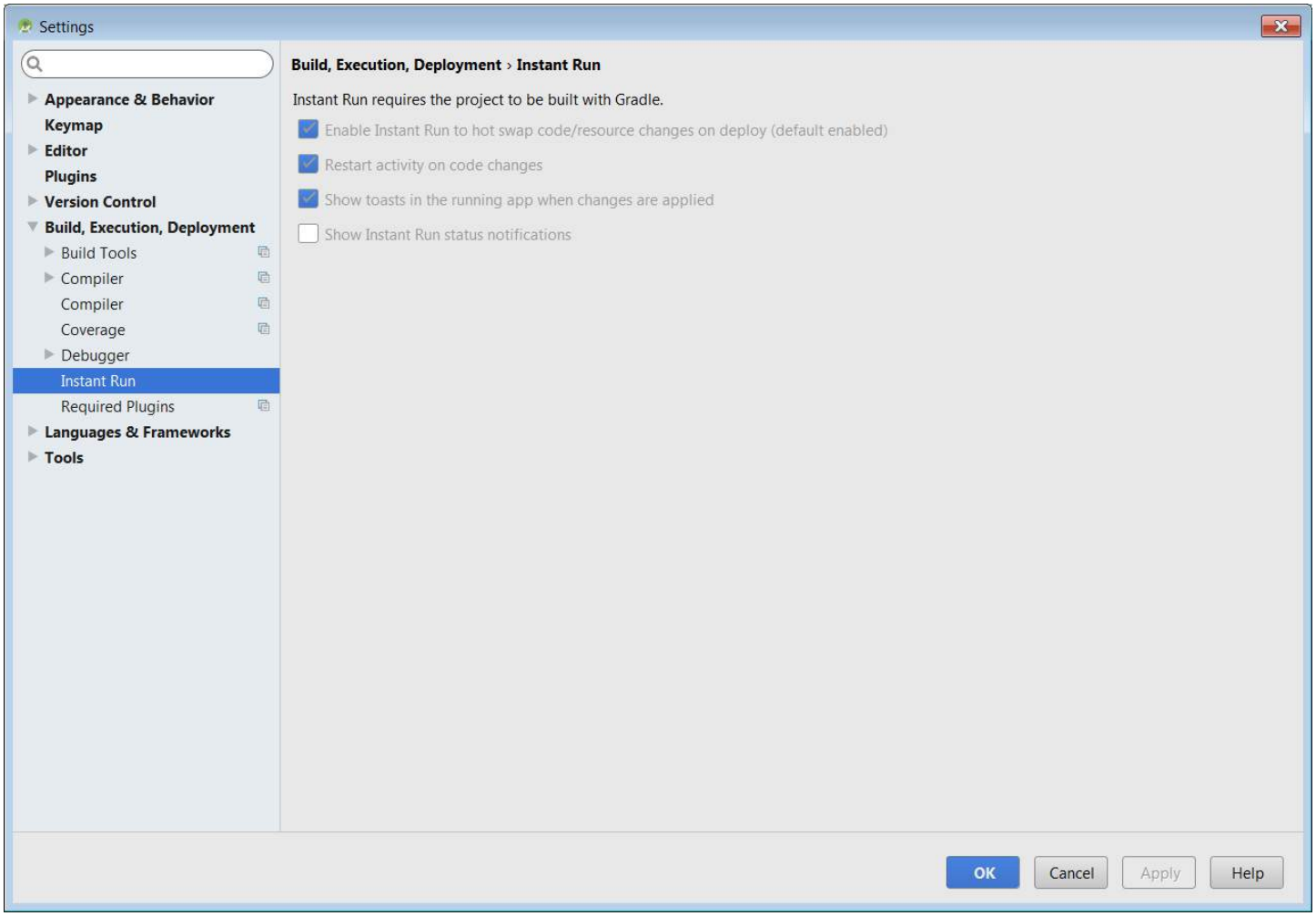
- Right click in MAIN folder > New > Folder > Assets Folder.
- Assets folder will be under MAIN folder with the same symbol as RES folder.
- In this example I put a font file.



Chapter 3: Instant Run in Android Studio

Section 3.1: Enabling or disabling Instant Run

1. Open the Settings or Preferences dialog:
    - On Windows or Linux, select `File` > `Settings` from the main menu.
    - On Mac OSX, select `Android Studio` > `Preferences` from the main menu.
2. Navigate to `Build, Execution, Deployment` > `Compiler`.
3. In the text field next to Command-line Options, enter your command-line options.
4. Click OK to save and exit.



The top option is Instant run. Check/uncheck that box.

Documentation

Section 3.2: Types of code Swaps in Instant Run

There are three types of code swaps that Instant run enables to support faster debugging and running app from your code in Android Studio.

- Hot Swap
- Warm Swap
- Cold Swap

When are each of these swaps triggered?

HOT SWAP is triggered when an existing method's implementation is changed.

WARM SWAP is triggered when an existing resource is changed or removed (anything in the res folder)

COLD SWAP whenever there is a structural code change in your app's code e.g.

1. Add, remove, or change:

- an annotation
- an instance field
- a static field
- a static method signature
- an instance method signature

2. Change which parent class the current class inherits from
3. Change the list of implemented interfaces
4. Change a class's static initializer
5. Reorder layout elements that use dynamic resource IDs

What happens when a code swap happens?

HOT SWAP changes are visible instantly - as soon as the next call to the method whose implementation is changed is made.

WARM SWAP restarts the current activity

COLD SWAP restarts the entire app (without reinstall)

Section 3.3: Unsupported code changes when using Instant Run

There are a few changes where instant won't do its trick and a full build and reinstall fo your app will happen just like it used to happen before Instant Run was born.

1. Change the app manifest
2. Change resources referenced by the app manifest
3. Change an Android widget UI element (requires a Clean and Rerun)

Documentation

Chapter 4: TextView

Everything related to TextView customization in Android SDK

Section 4.1: Spannable TextView

A spannable `TextView` can be used in Android to highlight a particular portion of text with a different color, style, size, and/or click event in a single `TextView` widget.

Consider that you have defined a `TextView` as follows:

```
TextView textview=findViewById(R.id.textview);
```

Then you can apply different highlighting to it as shown below:

- **Spannable color:** In order to set a different color to some portion of text, a `ForegroundColorSpan` can be used, as shown in the following example:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan(new ForegroundColorSpan(firstWordColor), 0, firstWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
spannable.setSpan(new ForegroundColorSpan(lastWordColor), firstWord.length(),
firstWord.length()+lastWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textview.setText( spannable );
```

  Output created by the code above:

- **Spannable font:** In order to set a different font size to some portion of text, a `RelativeSizeSpan` can be used, as shown in the following example:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
spannable.setSpan(new RelativeSizeSpan(1.1f),0, firstWord.length(),
 Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // set size
spannable.setSpan(new RelativeSizeSpan(0.8f), firstWord.length(), firstWord.length() +
lastWord.length(),  Spannable.SPAN_EXCLUSIVE_EXCLUSIVE); // set size
textview.setText( spannable );
```

  Output created by the code above:

- **Spannable typeface:** In order to set a different font typeface to some portion of text, a custom `TypefaceSpan` can be used, as shown in the following example:

```
Spannable spannable = new SpannableString(firstWord+lastWord);
```

```
spannable.setSpan( new CustomTypefaceSpan("SFUIText-Bold.otf",fontBold), 0,
firstWord.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
spannable.setSpan( new CustomTypefaceSpan("SFUIText-Regular.otf",fontRegular),
firstWord.length(), firstWord.length() + lastWord.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
text.setText( spannable );
```

However, in order to make the above code working, the class `CustomTypefaceSpan` has to be derived from the class `TypefaceSpan`. This can be done as follows:

```
public class CustomTypefaceSpan extends TypefaceSpan {
    private final Typeface newType;

    public CustomTypefaceSpan(String family, Typeface type) {
        super(family);
        newType = type;
    }

    @Override
    public void updateDrawState(TextPaint ds) {
        applyCustomTypeFace(ds, newType);
    }

    @Override
    public void updateMeasureState(TextPaint paint) {
        applyCustomTypeFace(paint, newType);
    }

    private static void applyCustomTypeFace(Paint paint, Typeface tf) {
        int oldStyle;
        Typeface old = paint.getTypeface();
        if (old == null) {
            oldStyle = 0;
        } else {
            oldStyle = old.getStyle();
        }
        int fake = oldStyle & ~tf.getStyle();
        if ((fake & Typeface.BOLD) != 0) {
            paint.setFakeBoldText(true);
        }

        if ((fake & Typeface.ITALIC) != 0) {
            paint.setTextSkewX(-0.25f);
        }

        paint.setTypeface(tf);
    }
}
```

Section 4.2: Strikethrough TextView

Strikethrough the entire text

```
String sampleText = "This is a test strike";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);
```

Output: This is a test strike

Strikethrough only parts of the text

```
String sampleText = "This is a test strike";
SpannableStringBuilder spanBuilder = new SpannableStringBuilder(sampleText);
StrikethroughSpan strikethroughSpan = new StrikethroughSpan();
spanBuilder.setSpan(
        strikethroughSpan, // Span to add
        0, // Start
        4, // End of the span (exclusive)
        Spanned.SPAN_EXCLUSIVE_EXCLUSIVE // Text changes will not reflect in the strike changing
);
textView.setText(spanBuilder);
```

Output: This is a test strike

Section 4.3: TextView with image

Android allows programmers to place images at all four corners of a `TextView`. For example, if you are creating a field with a `TextView` and at same time you want to show that the field is editable, then developers will usually place an edit icon near that field. Android provides us an interesting option called **compound drawable** for a `TextView`:

```
<TextView
       android:id="@+id/title"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_centerInParent="true"
       android:drawablePadding="4dp"
       android:drawableRight="@drawable/edit"
       android:text="Hello world"
       android:textSize="18dp" />
```

You can set the drawable to any side of your `TextView` as follows:

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

Setting the drawable can also be achieved programmatically in the following way:

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

Setting any of the parameters handed over to `setCompoundDrawables()` to `null` will remove the icon from the corresponding side of the `TextView`.

Section 4.4: Make RelativeSizeSpan align to top

In order to make a `RelativeSizeSpan` align to the top, a custom class can be derived from the class `SuperscriptSpan`. In the following example, the derived class is named `TopAlignSuperscriptSpan`:

activity_main.xml:

```
<TextView
   android:id="@+id/txtView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginTop="50dp"
```

```
    android:textSize="26sp" />
```

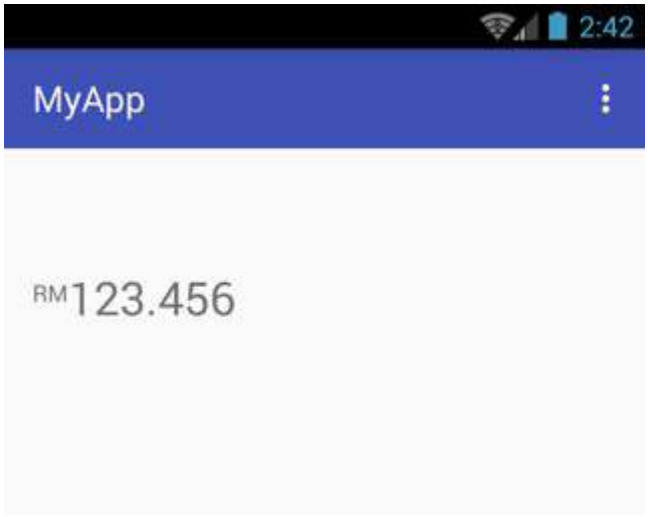
MainActivity.java:

```
TextView txtView = (TextView) findViewById(R.id.txtView);

SpannableString spannableString = new SpannableString("RM123.456");
spannableString.setSpan( new TopAlignSuperscriptSpan( (float)0.35 ), 0, 2,
Spanned.SPAN_EXCLUSIVE_EXCLUSIVE );
txtView.setText(spannableString);
```

TopAlignSuperscriptSpan.java:

```
private class TopAlignSuperscriptSpan extends SuperscriptSpan {
    //divide superscript by this number
    protected int fontScale = 2;

    //shift value, 0 to 1.0
    protected float shiftPercentage = 0;

    //doesn't shift
    TopAlignSuperscriptSpan() {}

    //sets the shift percentage
    TopAlignSuperscriptSpan( float shiftPercentage ) {
        if( shiftPercentage > 0.0 && shiftPercentage < 1.0 )
            this.shiftPercentage = shiftPercentage;
    }

    @Override
    public void updateDrawState( TextPaint tp ) {
        //original ascent
        float ascent = tp.ascent();

        //scale down the font
        tp.setTextSize( tp.getTextSize() / fontScale );

        //get the new font ascent
        float newAscent = tp.getFontMetrics().ascent;

        //move baseline to top of old font, then move down size of new font
        //adjust for errors with shift percentage
        tp.baselineShift += ( ascent - ascent * shiftPercentage )
                - (newAscent - newAscent * shiftPercentage );
    }

    @Override
    public void updateMeasureState( TextPaint tp ) {
        updateDrawState( tp );
    }
}
```

Reference screenshot:



Section 4.5: Pinchzoom on TextView

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

    <TextView
       android:id="@+id/mytv"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:layout_alignParentLeft="true"
       android:layout_alignParentTop="true"
       android:text="This is my sample text for pinch zoom demo, you can zoom in and out using
pinch zoom, thanks" />

</RelativeLayout>
```

MainActivity.java:

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.TextView;

public class MyTextViewPinchZoomClass extends Activity implements OnTouchListener {

    final static float STEP = 200;
    TextView mytv;
    float mRatio = 1.0f;
    int mBaseDist;
    float mBaseRatio;
    float fontsize = 13;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytv = (TextView) findViewById(R.id.mytv);
```

```
        mytv.setTextSize(mRatio + 13);
    }

    public boolean onTouchEvent(MotionEvent event) {
        if (event.getPointerCount() == 2) {
            int action = event.getAction();
            int pureaction = action & MotionEvent.ACTION_MASK;
            if (pureaction == MotionEvent.ACTION_POINTER_DOWN) {
                mBaseDist = getDistance(event);
                mBaseRatio = mRatio;
            } else {
                float delta = (getDistance(event) - mBaseDist) / STEP;
                float multi = (float) Math.pow(2, delta);
                mRatio = Math.min(1024.0f, Math.max(0.1f, mBaseRatio * multi));
                mytv.setTextSize(mRatio + 13);
            }
        }
        return true;
    }

    int getDistance(MotionEvent event) {
        int dx = (int) (event.getX(0) - event.getX(1));
        int dy = (int) (event.getY(0) - event.getY(1));
        return (int) (Math.sqrt(dx * dx + dy * dy));
    }

    public boolean onTouch(View v, MotionEvent event) {
        return false;
    }
}
```

Section 4.6: Textview with different Textsize

You can archive different Textsizes inside a Textview with a Span

```
TextView textView = (TextView) findViewById(R.id.textView);
Spannable span = new SpannableString(textView.getText());
span.setSpan(new RelativeSizeSpan(0.8f), start, end, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textView.setText(span)
```

Section 4.7: Theme and Style customization

MainActivity.java:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:custom="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

     <com.customthemeattributedemo.customview.CustomTextView
        style="?mediumTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />

     <com.customthemeattributedemo.customview.CustomTextView
        style="?largeTextStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="@string/message_hello"
        custom:font_family="@string/bold_font" />
</LinearLayout>
```

CustomTextView.java:

```
public class CustomTextView extends TextView {

    private static final String TAG = "TextViewPlus";
    private Context mContext;

    public CustomTextView(Context context) {
        super(context);
        mContext = context;
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        setCustomFont(context, attrs);
    }

    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mContext = context;
        setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray customFontNameTypedArray = ctx.obtainStyledAttributes(attrs,
R.styleable.CustomTextView);
        String customFont =
customFontNameTypedArray.getString(R.styleable.CustomTextView_font_family);
        Typeface typeface = null;
        typeface = Typeface.createFromAsset(ctx.getAssets(), customFont);
        setTypeface(typeface);
        customFontNameTypedArray.recycle();
    }
}
```

attrs.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <attr name="mediumTextStyle" format="reference" />
    <attr name="largeTextStyle" format="reference" />

    <declare-styleable name="CustomTextView">

        <attr name="font_family" format="string" />
        <!--- Your other attributes -->

    </declare-styleable>
</resources>
```

strings.xml:

```
<resources>
    <string name="app_name">Custom Style Theme Attribute Demo</string>
    <string name="message_hello">Hello Hiren!</string>

    <string name="bold_font">bold.ttf</string>
</resources>
```

styles.xml:

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>

        <item name="mediumTextStyle">@style/textMedium</item>
        <item name="largeTextStyle">@style/textLarge</item>
    </style>


    <style name="textMedium" parent="textParentStyle">
        <item name="android:textAppearance">@android:style/TextAppearance.Medium</item>
    </style>

    <style name="textLarge" parent="textParentStyle">
        <item name="android:textAppearance">@android:style/TextAppearance.Large</item>
    </style>

    <style name="textParentStyle">
        <item name="android:textColor">@android:color/white</item>
        <item name="android:background">@color/colorPrimary</item>
        <item name="android:padding">5dp</item>
    </style>

</resources>
```

Section 4.8: TextView customization

```
public class CustomTextView extends TextView {
```

```
    private float strokeWidth;
    private Integer strokeColor;
    private Paint.Join strokeJoin;
    private float strokeMiter;


    public CustomTextView(Context context) {
        super(context);
        init(null);
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs);
    }

    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(attrs);
    }

    public void init(AttributeSet attrs) {

        if (attrs != null) {
            TypedArray a = getContext().obtainStyledAttributes(attrs, R.styleable.CustomTextView);

            if (a.hasValue(R.styleable.CustomTextView_strokeColor)) {
                float strokeWidth = a.getDimensionPixelSize(R.styleable.CustomTextView_strokeWidth,
1);
                int strokeColor = a.getColor(R.styleable.CustomTextView_strokeColor, 0xff000000);
                float strokeMiter = a.getDimensionPixelSize(R.styleable.CustomTextView_strokeMiter,
10);
                Paint.Join strokeJoin = null;
                switch (a.getInt(R.styleable.CustomTextView_strokeJoinStyle, 0)) {
                    case (0):
                        strokeJoin = Paint.Join.MITER;
                        break;
                    case (1):
                        strokeJoin = Paint.Join.BEVEL;
                        break;
                    case (2):
                        strokeJoin = Paint.Join.ROUND;
                        break;
                }
                this.setStroke(strokeWidth, strokeColor, strokeJoin, strokeMiter);
            }
        }
    }

    public void setStroke(float width, int color, Paint.Join join, float miter) {
        strokeWidth = width;
        strokeColor = color;
        strokeJoin = join;
        strokeMiter = miter;
    }

    @Override
    public void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        int restoreColor = this.getCurrentTextColor();
        if (strokeColor != null) {
```

```
            TextPaint paint = this.getPaint();
            paint.setStyle(Paint.Style.STROKE);
            paint.setStrokeJoin(strokeJoin);
            paint.setStrokeMiter(strokeMiter);
            this.setTextColor(strokeColor);
            paint.setStrokeWidth(strokeWidth);
            super.onDraw(canvas);
            paint.setStyle(Paint.Style.FILL);
            this.setTextColor(restoreColor);
        }
    }
}
```

Usage:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        CustomTextView customTextView = (CustomTextView) findViewById(R.id.pager_title);
    }
}
```

Layout:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="@mipmap/background">


    <pk.sohail.gallerytest.activity.CustomTextView
       android:id="@+id/pager_title"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_centerHorizontal="true"
       android:layout_centerVertical="true"
       android:gravity="center"
       android:text="@string/txt_title_photo_gallery"
       android:textColor="@color/white"
       android:textSize="30dp"
       android:textStyle="bold"
       app:outerShadowRadius="10dp"
       app:strokeColor="@color/title_text_color"
       app:strokeJoinStyle="miter"
       app:strokeWidth="2dp" />

</RelativeLayout>
```

attars:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <declare-styleable name="CustomTextView">
```

```
        <attr name="outerShadowRadius" format="dimension" />
        <attr name="strokeWidth" format="dimension" />
        <attr name="strokeMiter" format="dimension" />
        <attr name="strokeColor" format="color" />
        <attr name="strokeJoinStyle">
            <enum name="miter" value="0" />
            <enum name="bevel" value="1" />
            <enum name="round" value="2" />
        </attr>
    </declare-styleable>

</resources>
```

Programmatically usage:

```
CustomTextView mtxt_name = (CustomTextView) findViewById(R.id.pager_title);
//then use
setStroke(float width, int color, Paint.Join join, float miter);
//method before setting
setText("Sample Text");
```

Section 4.9: Single TextView with two different colors

Colored text can be created by passing the text and a font color name to the following function:

```
private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}
```

The colored text can then be set to a `TextView` (or even to a `Button`, `EditText`, etc.) by using the example code below.

First, define a `TextView` as follows:

```
TextView txtView = (TextView)findViewById(R.id.txtView);
```

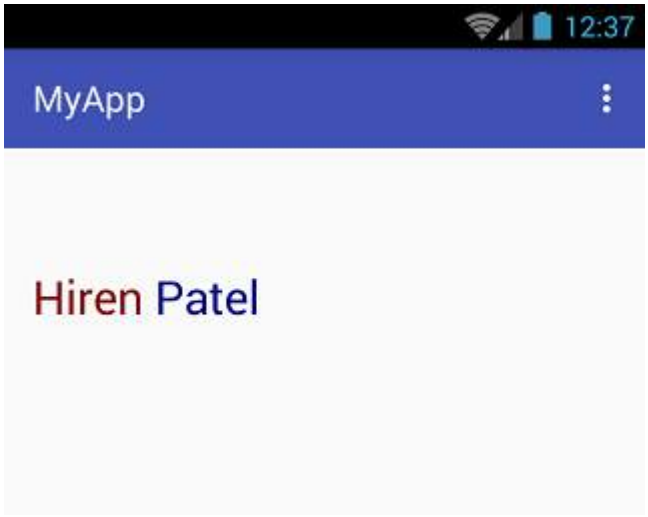
Then, create differently colored text and assign it to strings:

```
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel","#000080");
```

Finally, set the two differently colored strings to the `TextView`:

```
txtView.setText(Html.fromHtml(name+" "+surName));
```

Reference screenshot:



Chapter 5: AutoCompleteTextView

Section 5.1: AutoComplete with CustomAdapter, ClickListener and Filter

Main layout : activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

    <AutoCompleteTextView
       android:id="@+id/auto_name"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:completionThreshold="2"
       android:hint="@string/hint_enter_name" />
</LinearLayout>
```

Row layout row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/lbl_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="16dp"
        android:paddingLeft="8dp"
        android:paddingRight="8dp"
        android:paddingTop="16dp"
        android:text="Medium Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="hint_enter_name">Enter Name</string>
</resources>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    AutoCompleteTextView txtSearch;
    List<People> mList;
    PeopleAdapter adapter;
    private People selectedPerson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        mList = retrievePeople();
        txtSearch = (AutoCompleteTextView) findViewById(R.id.auto_name);
        adapter = new PeopleAdapter(this, R.layout.activity_main, R.id.lbl_name, mList);
        txtSearch.setAdapter(adapter);
        txtSearch.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int pos, long id) {
                //this is the way to find selected object/item
                selectedPerson = (People) adapterView.getItemAtPosition(pos);
            }
        });
    }

    private List<People> retrievePeople() {
        List<People> list = new ArrayList<People>();
        list.add(new People("James", "Bond", 1));
        list.add(new People("Jason", "Bourne", 2));
        list.add(new People("Ethan", "Hunt", 3));
        list.add(new People("Sherlock", "Holmes", 4));
        list.add(new People("David", "Beckham", 5));
        list.add(new People("Bryan", "Adams", 6));
        list.add(new People("Arjen", "Robben", 7));
        list.add(new People("Van", "Persie", 8));
        list.add(new People("Zinedine", "Zidane", 9));
        list.add(new People("Luis", "Figo", 10));
        list.add(new People("John", "Watson", 11));
        return list;
    }
}
```

Model class : People.java

```
public class People {

    private String name, lastName;
    private int id;

    public People(String name, String lastName, int id) {
        this.name = name;
        this.lastName = lastName;
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getlastName() {
```

```
        return lastName;
    }

    public void setlastName(String lastName) {
        this.lastName = lastName;
    }
}
```

Adapter class : PeopleAdapter.java

```
public class PeopleAdapter extends ArrayAdapter<People> {

    Context context;
    int resource, textViewResourceId;
    List<People> items, tempItems, suggestions;

    public PeopleAdapter(Context context, int resource, int textViewResourceId, List<People> items)
{
        super(context, resource, textViewResourceId, items);
        this.context = context;
        this.resource = resource;
        this.textViewResourceId = textViewResourceId;
        this.items = items;
        tempItems = new ArrayList<People>(items); // this makes the difference.
        suggestions = new ArrayList<People>();
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = convertView;
        if (convertView == null) {
            LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            view = inflater.inflate(R.layout.row, parent, false);
        }
        People people = items.get(position);
        if (people != null) {
            TextView lblName = (TextView) view.findViewById(R.id.lbl_name);
            if (lblName != null)
                lblName.setText(people.getName());
        }
        return view;
    }

    @Override
    public Filter getFilter() {
        return nameFilter;
    }

    /**
     * Custom Filter implementation for custom suggestions we provide.
     */
    Filter nameFilter = new Filter() {
        @Override
        public CharSequence convertResultToString(Object resultValue) {
            String str = ((People) resultValue).getName();
            return str;
        }

        @Override
        protected FilterResults performFiltering(CharSequence constraint) {
```

```
            if (constraint != null) {
                suggestions.clear();
                for (People people : tempItems) {
                    if
(people.getName().toLowerCase().contains(constraint.toString().toLowerCase())) {
                        suggestions.add(people);
                    }
                }
                FilterResults filterResults = new FilterResults();
                filterResults.values = suggestions;
                filterResults.count = suggestions.size();
                return filterResults;
            } else {
                return new FilterResults();
            }
        }

        @Override
        protected void publishResults(CharSequence constraint, FilterResults results) {
            List<People> filterList = (ArrayList<People>) results.values;
            if (results != null && results.count > 0) {
                clear();
                for (People people : filterList) {
                    add(people);
                    notifyDataSetChanged();
                }
            }
        }
    };
}
```

Section 5.2: Simple, hard-coded AutoCompleteTextView

Design (layout XML):

```
<AutoCompleteTextView
  android:id="@+id/autoCompleteTextView1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignParentTop="true"
  android:layout_centerHorizontal="true"
  android:layout_marginTop="65dp"
  android:ems="10" />
```

Find the view in code after `setContentView()` (or its fragment or custom view equivalent):

```
final AutoCompleteTextView myAutoCompleteTextView =
    (AutoCompleteTextView) findViewById(R.id.autoCompleteTextView1);
```

Provide hard-coded data via an adapter:

```
String[] countries = getResources().getStringArray(R.array.list_of_countries);
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,countries);
myAutoCompleteTextView.setAdapter(adapter);
```

Tip: Though the preferred way would be to provide data via a `Loader` of some kind instead of a hard-coded list like this.

Chapter 6: Autosizing TextViews

A TextView that automatically resizes text to fit perfectly within its bounds.

Android O allows you to instruct a TextView to let the size of the text expand or contract automatically to fill its layout based on the TextView's characteristics and boundaries.

You can set up the TextView autosizing in either code or XML.

There are two ways to set autosizing TextView: **Granularity** and **Preset Sizes**

Section 6.1: Granularity

In Java:

Call the `setAutoSizeTextTypeUniformWithConfiguration()` method:

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize, int
autoSizeStepGranularity, int unit)
```

In XML:

Use the `autoSizeMinTextSize`, `autoSizeMaxTextSize`, and `autoSizeStepGranularity` attributes to set the auto-sizing dimensions in the layout XML file:

```
<TextView android:id="@+id/autosizing_textview_presetsize"
        android:layout_width="wrap_content"
        android:layout_height="250dp"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="0dp"
        android:autoSizeMaxTextSize="100sp"
        android:autoSizeMinTextSize="12sp"
        android:autoSizeStepGranularity="2sp"
        android:autoSizeText="uniform"
        android:text="Hello World!"
        android:textSize="100sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Check out the AutosizingTextViews-Demo at GitHub for more details.

Section 6.2: Preset Sizes

In Java:

Call the `setAutoSizeTextTypeUniformWithPresetSizes()` method:

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

In XML:

Use the `autoSizePresetSizes` attribute in the layout XML file:

```
<TextView android:id="@+id/autosizing_textview_presetsize"
```

```
        android:layout_width="wrap_content"
        android:layout_height="250dp"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="0dp"
        android:autoSizeText="uniform"
        android:autoSizePresetSizes="@array/autosize_text_sizes"
        android:text="Hello World!"
        android:textSize="100sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

To access the array as a resource, define the array in the *res/values/arrays.xml* file:

```
<array name="autosize_text_sizes">
    <item>10sp</item>
    <item>12sp</item>
    <item>20sp</item>
    <item>40sp</item>
    <item>100sp</item>
</array>
```

Check out the AutosizingTextViews-Demo at GitHub for more details.

Chapter 7: ListView

ListView is a viewgroup which groups several items from a data source like array or database and displays them in a scroll-able list. Data are bound with listview using an Adapter class.

Section 7.1: Custom ArrayAdapter

By default the ArrayAdapter class creates a view for each array item by calling `toString()` on each item and placing the contents in a TextView.

To create a complex view for each item (for example, if you want an ImageView for each array item), extend the ArrayAdapter class and override the `getView()` method to return the type of View you want for each item.

For example:

```
public class MyAdapter extends ArrayAdapter<YourClassData>{

    private LayoutInflater inflater;

    public MyAdapter (Context context, List<YourClassData> data){
        super(context, 0, data);
        inflater = LayoutInflater.from(context);
    }

    @Override
    public long getItemId(int position)
    {
        //It is just an example
        YourClassData data = (YourClassData) getItem(position);
        return data.ID;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent)
    {
        ViewHolder viewHolder;
        if (view == null) {
            view = inflater.inflate(R.layout.custom_row_layout_design, null);
            // Do some initialization

            //Retrieve the view on the item layout and set the value.
            viewHolder = new ViewHolder(view);
            view.setTag(viewHolder);
         }
         else {
             viewHolder = (ViewHolder) view.getTag();
         }

        //Retrieve your object
        YourClassData data = (YourClassData) getItem(position);

        viewHolder.txt.setTypeface(m_Font);
        viewHolder.txt.setText(data.text);
        viewHolder.img.setImageBitmap(BitmapFactory.decodeFile(data.imageAddr));

        return view;

    }
```

```
    private class ViewHolder
    {
         private final TextView txt;
         private final ImageView img;

         private ViewHolder(View view)
         {
             txt = (TextView) view.findViewById(R.id.txt);
             img = (ImageView) view.findViewById(R.id.img);
         }
    }
}
```

Section 7.2: A basic ListView with an ArrayAdapter

By default the `ArrayAdapter` creates a view for each array item by calling `toString()` on each item and placing the contents in a `TextView`.

Example:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

where `android.R.layout.simple_list_item_1` is the layout that contains a `TextView` for each string in the array.

Then simply call `setAdapter()` on your `ListView`:

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

To use something other than TextViews for the array display, for instance, ImageViews, or to have some of data besides `toString()` results fill the views, override `getView(int, View, ViewGroup)` to return the type of view you want. Check this example.

Section 7.3: Filtering with CursorAdapter

```
// Get the reference to your ListView
ListView listResults = (ListView) findViewById(R.id.listResults);

// Set its adapter
listResults.setAdapter(adapter);

// Enable filtering in ListView
listResults.setTextFilterEnabled(true);

// Prepare your adapter for filtering
adapter.setFilterQueryProvider(new FilterQueryProvider() {
    @Override
    public Cursor runQuery(CharSequence constraint) {

        // in real life, do something more secure than concatenation
        // but it will depend on your schema
        // This is the query that will run on filtering
        String query = "SELECT _ID as _id, name FROM MYTABLE "
                       + "where name like '%" + constraint + "%' "
                       + "ORDER BY NAME ASC";
        return db.rawQuery(query, null);
     }
```

```
});
```

Let's say your query will run every time the user types in an `EditText`:

```
    EditText queryText = (EditText) findViewById(R.id.textQuery);
    queryText.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(final CharSequence s, final int start, final int count, final
int after) {

        }

        @Override
        public void onTextChanged(final CharSequence s, final int start, final int before, final
int count) {
            // This is the filter in action
            adapter.getFilter().filter(s.toString());
            // Don't forget to notify the adapter
            adapter.notifyDataSetChanged();
        }

        @Override
        public void afterTextChanged(final Editable s) {

        }
    });
```

Chapter 8: Layouts

A layout defines the visual structure for a user interface, such as an activity or widget.

A layout is declared in XML, including screen elements that will appear in it. Code can be added to the application to modify the state of screen objects at runtime, including those declared in XML.

Section 8.1: LayoutParams

Every single `ViewGroup` (e.g. `LinearLayout`, `RelativeLayout`, `CoordinatorLayout`, etc.) needs to store information about its children's properties. About the way its children are being laid out in the `ViewGroup`. This information is stored in objects of a wrapper class `ViewGroup.LayoutParams`.

To include parameters specific to a particular layout type, `ViewGroups` use subclasses of `ViewGroup.LayoutParams` class.

E.g. for

- `LinearLayout` it's `LinearLayout.LayoutParams`
- `RelativeLayout` it's `RelativeLayout.LayoutParams`
- `CoordinatorLayout` it's `CoordinatorLayout.LayoutParams`
- ...

Most of `ViewGroups` reutilize the ability to set `margins` for their children, so they do not subclass `ViewGroup.LayoutParams` directly, but they subclass `ViewGroup.MarginLayoutParams` instead (which itself is a subclass of `ViewGroup.LayoutParams`).

`LayoutParams` in xml

`LayoutParams` objects are created based on the inflated layout `xml` file.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">

    <TextView
       android:layout_width="wrap_content"
       android:layout_height="50dp"
       android:layout_gravity="right"
       android:gravity="bottom"
       android:text="Example text"
       android:textColor="@android:color/holo_green_dark"/>

    <ImageView
       android:layout_width="match_parent"
       android:layout_height="0dp"
       android:layout_weight="1"
       android:background="@android:color/holo_green_dark"
       android:scaleType="centerInside"
       android:src="@drawable/example"/>

</LinearLayout>
```

All parameters that begin with `layout_` specify how the **enclosing** layout should work. When the layout is inflated, those parameters are wrapped in a proper `LayoutParams` object, that later will be used by the `Layout` to properly

position a particular `View` within the `ViewGroup`. Other attributes of a `View` are directly `View`-related and are processed by the `View` itself.

For `TextView`:

- `layout_width`, `layout_height` and `layout_gravity` will be stored in a `LinearLayout.LayoutParams` object and used by the `LinearLayout`
- `gravity`, `text` and `textColor` will be used by the `TextView` itself

For `ImageView`:

- `layout_width`, `layout_height` and `layout_weight` will be stored in a `LinearLayout.LayoutParams` object and used by the `LinearLayout`
- `background`, `scaleType` and `src` will be used by the `ImageView` itself

Getting `LayoutParams` object

`getLayoutParams` is a `View's` method that allows to retrieve a current `LayoutParams` object.

Because the `LayoutParams` object is directly related to the **enclosing** `ViewGroup`, this method will return a non-null value only when `View` is attached to the `ViewGroup`. You need to bare in mind that this object might not be present at all times. Especially you should not depend on having it inside `View's` constructor.

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setupView(context);
    }

    private void setupView(Context context) {
        if (getLayoutParams().height == 50){  // DO NOT DO THIS!
                                              // This might produce NullPointerException
            doSomething();
        }
    }

    //...
}
```

If you want to depend on having `LayoutParams` object, you should use `onAttachedToWindow` method instead.

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
    }
```

```
    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        if (getLayoutParams().height == 50) { // getLayoutParams() will NOT return null here
            doSomething();
        }
    }

    //...
}
```

Casting `LayoutParams` object

You might need to use features that are specific to a particular `ViewGroup` (e.g. you might want to programmatically change rules of a `RelativeLayout`). For that purpose you will need to know how to properly cast the `ViewGroup.LayoutParams` object.

This might be a bit confusing when getting a `LayoutParams` object for a child `View` that actually is another `ViewGroup`.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/outer_layout"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical">

    <FrameLayout
       android:id="@+id/inner_layout"
       android:layout_width="match_parent"
       android:layout_height="50dp"
       android:layout_gravity="right"/>

</LinearLayout>
```

IMPORTANT: The type of `LayoutParams` object is directly related to the type of the **ENCLOSING** `ViewGroup`.

Incorrect *casting*:

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
                                        // INCORRECT! This will produce ClassCastException
```

Correct *casting*:

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
                                      // CORRECT! the enclosing layout is a LinearLayout
```

Section 8.2: Gravity and layout gravity

android:layout_gravity

- `android:layout_gravity` is used to set the position of an element in its parent (e.g. a child `View` inside a `Layout`).
- Supported by LinearLayout and FrameLayout

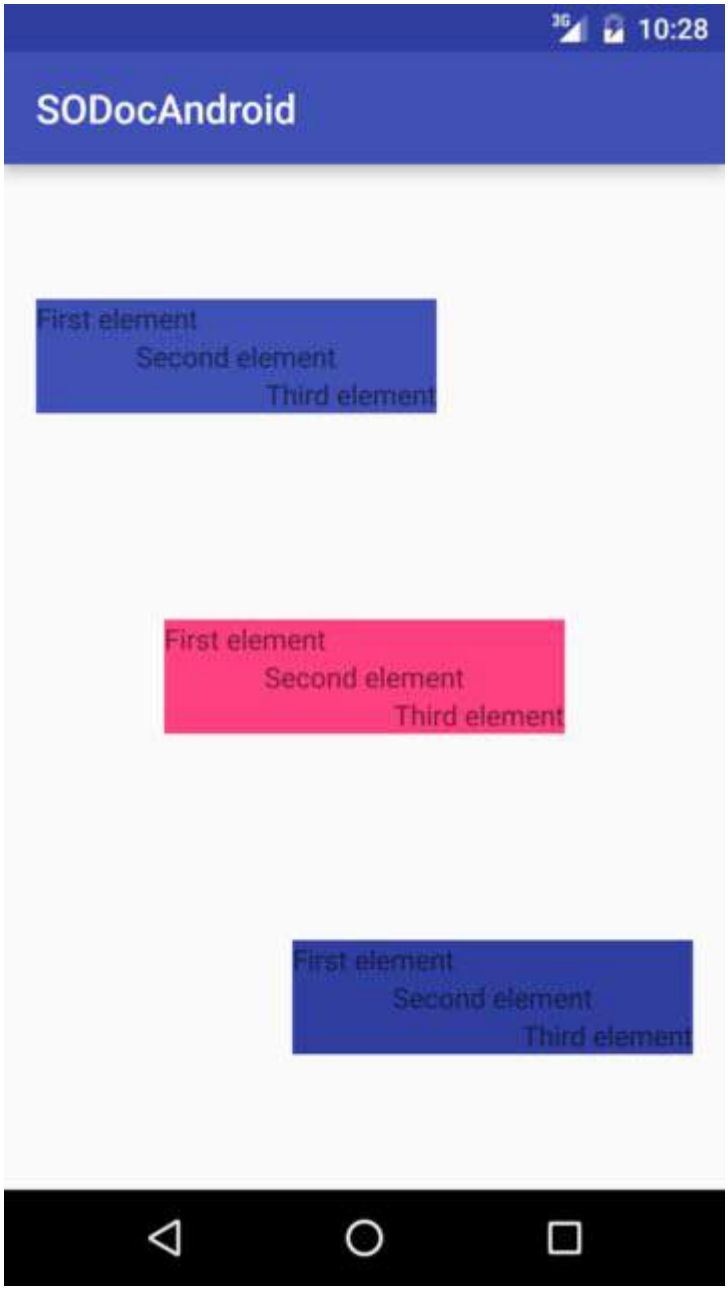
android:gravity

- `android:gravity` is used to set the position of content inside an element (e.g. a text inside a `TextView`).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:orientation="vertical">

    <LinearLayout
       android:layout_width="wrap_content"
       android:layout_height="0dp"
       android:layout_weight="1"
       android:orientation="vertical"
       android:layout_gravity="left"
       android:gravity="center_vertical">

        <TextView
           android:layout_width="@dimen/fixed"
           android:layout_height="wrap_content"
           android:text="@string/first"
           android:background="@color/colorPrimary"
           android:gravity="left"/>

        <TextView
           android:layout_width="@dimen/fixed"
           android:layout_height="wrap_content"
           android:text="@string/second"
           android:background="@color/colorPrimary"
           android:gravity="center"/>

        <TextView
           android:layout_width="@dimen/fixed"
           android:layout_height="wrap_content"
           android:text="@string/third"
           android:background="@color/colorPrimary"
           android:gravity="right"/>

    </LinearLayout>

    <LinearLayout
       android:layout_width="wrap_content"
       android:layout_height="0dp"
       android:layout_weight="1"
       android:orientation="vertical"
       android:layout_gravity="center"
       android:gravity="center_vertical">
```

```
        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorAccent"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorAccent"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorAccent"
            android:gravity="right"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="right"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorPrimaryDark"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorPrimaryDark"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorPrimaryDark"
            android:gravity="right"/>

    </LinearLayout>

</LinearLayout>
```

Which gets rendered as following:



Section 8.3: CoordinatorLayout Scrolling Behavior

`Version ≥ 2.3-2.3.2`

An enclosing `CoordinatorLayout` can be used to achieve Material Design Scrolling Effects when using inner layouts that support Nested Scrolling, such as `NestedScrollView` or `RecyclerView`.

For this example:

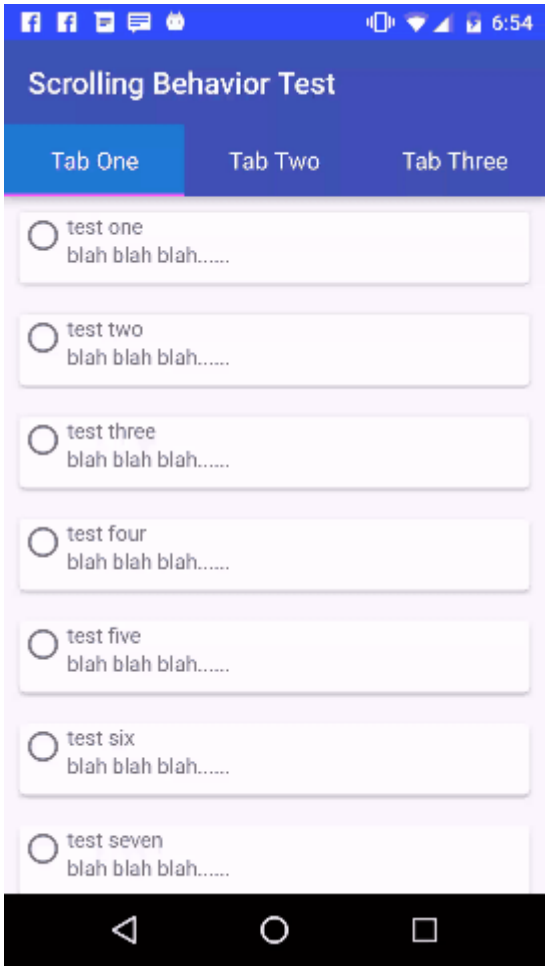
- `app:layout_scrollFlags="scroll|enterAlways"` is used in the Toolbar properties
- `app:layout_behavior="@string/appbar_scrolling_view_behavior"` is used in the ViewPager properties
- A RecyclerView is used in the ViewPager Fragments

Here is the layout xml file used in an Activity:

```
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/main_layout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


     <android.support.design.widget.AppBarLayout
        android:id="@+id/appBarLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:elevation="6dp">
     <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:background="?attr/colorPrimary"
        android:minHeight="?attr/actionBarSize"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
        app:elevation="0dp"
        app:layout_scrollFlags="scroll|enterAlways"
        />

     <android.support.design.widget.TabLayout
        android:id="@+id/tab_layout"
        app:tabMode="fixed"
        android:layout_below="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/colorPrimary"
        app:elevation="0dp"
        app:tabTextColor="#d3d3d3"
        android:minHeight="?attr/actionBarSize"
        />

     </android.support.design.widget.AppBarLayout>

     <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_below="@+id/tab_layout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        />

</android.support.design.widget.CoordinatorLayout>
```

Result:



Section 8.4: Percent Layouts

Version ≥ 2.3

The Percent Support Library provides `PercentFrameLayout` and `PercentRelativeLayout`, two ViewGroups that provide an easy way to specify View **dimensions and margins** in terms of a **percentage** of the overall size.

You can use the Percent Support Library by adding the following to your dependencies.

```
compile 'com.android.support:percent:25.3.1'
```

If you wanted to display a view that fills the screen horizontally but only half the screen vertically you would do thie following.

```
<android.support.percent.PercentFrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

    <FrameLayout
       app:layout_widthPercent="100%"
       app:layout_heightPercent="50%"
       android:background="@android:color/black" />

<android.support.percent.PercentFrameLayout>
```

You can also define the percentages in a separate XML file with code such as:

```
<fraction name="margin_start_percent">25%</fraction>
```

And refer to them in your layouts with `@fraction/margin_start_percent`.

They also contain the ability to set a custom **aspect ratio** via `app:layout_aspectRatio`.
This allows you to set only a single dimension, such as only the width, and the height will be automatically determined based on the aspect ratio you've defined, whether it is 4:3 or 16:9 or even a square 1:1 aspect ratio.

For example:

```
<ImageView
  app:layout_widthPercent="100%"
  app:layout_aspectRatio="178%"
  android:scaleType="centerCrop"
  android:src="@drawable/header_background"/>
```

Section 8.5: View Weight

One of the most used attribute for LinearLayout is the weight of its child views. Weight defines how much space a view will consume compared to other views within a LinearLayout.

Weight is used when you want to give specific screen space to one component compared to other.

Key Properties:

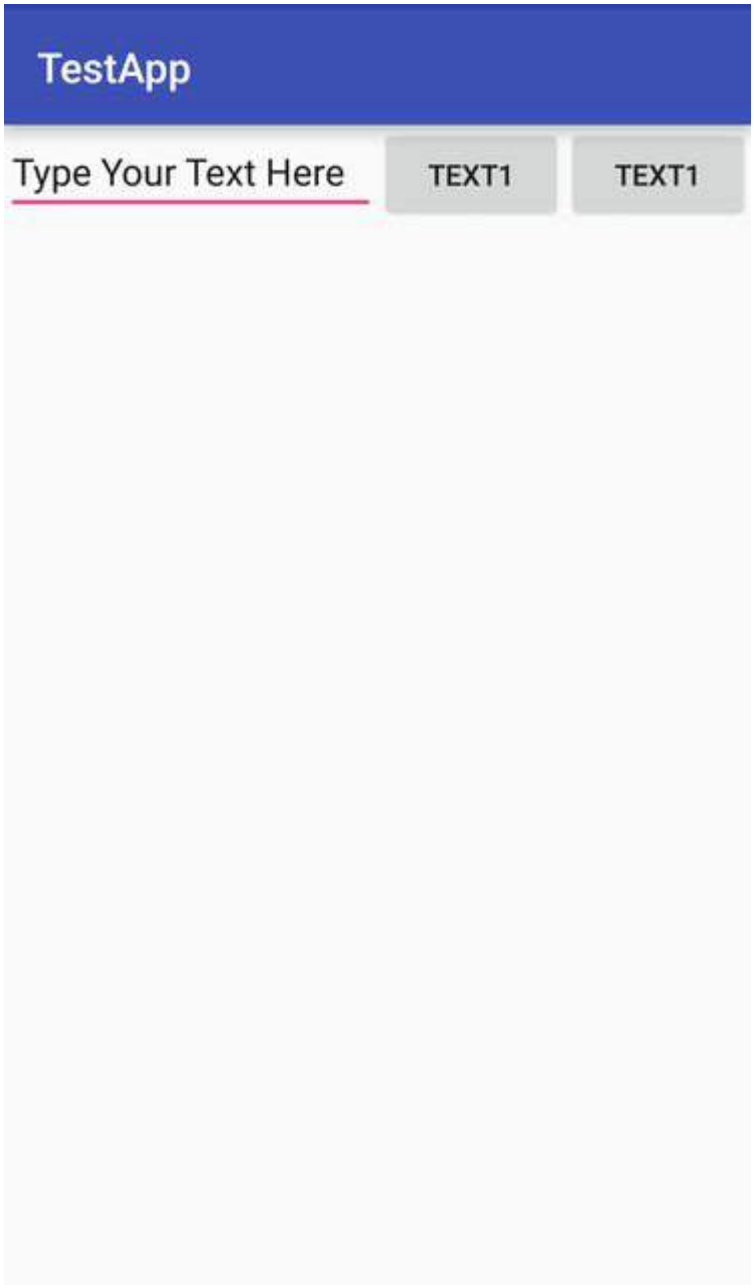
- `weightSum` is the overall sum of weights of all child views. If you don't specify the `weightSum`, the system will calculate the sum of all the weights on its own.
- `layout_weight` specifies the amount of space out of the total weight sum the widget will occupy.

Code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="horizontal"
   android:weightSum="4">

    <EditText
       android:layout_weight="2"
       android:layout_width="0dp"
       android:layout_height="wrap_content"
       android:text="Type Your Text Here" />

    <Button
       android:layout_weight="1"
       android:layout_width="0dp"
       android:layout_height="wrap_content"
       android:text="Text1" />

    <Button
       android:layout_weight="1"
       android:layout_width="0dp"
       android:layout_height="wrap_content"
```

```
        android:text="Text1" />

</LinearLayout>
```

The output is:



Now even if the size of the device is larger, the EditText will take 2/4 of the screen's space. Hence the look of your app is seen consistent across all screens.

Note: Here the `layout_width` is kept `0dp` as the widget space is divided horizontally. If the widgets are to be aligned vertically `layout_height` will be set to `0dp`. This is done to increase the efficiency of the code because at runtime the system won't attempt to calculate the width or height respectively as this is managed by the weight. If you instead used `wrap_content` the system would attempt to calculate the width/height first before applying the weight attribute which causes another calculation cycle.

Section 8.6: Creating LinearLayout programmatically

Hierarchy

```
- LinearLayout(horizontal)
   - ImageView
   - LinearLayout(vertical)
     - TextView
     - TextView
```

Code

```
LinearLayout rootView = new LinearLayout(context);
rootView.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
rootView.setOrientation(LinearLayout.HORIZONTAL);

// for imageview
ImageView imageView = new ImageView(context);
// for horizontal linearlayout
LinearLayout linearLayout2 = new LinearLayout(context);
linearLayout2.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
linearLayout2.setOrientation(LinearLayout.VERTICAL);

TextView tv1 = new TextView(context);
TextView tv2 = new TextView(context);
// add 2 textview to horizontal linearlayout
linearLayout2.addView(tv1);
linearLayout2.addView(tv2);

// finally, add imageview and horizontal linearlayout to vertical linearlayout (rootView)
rootView.addView(imageView);
rootView.addView(linearLayout2);
```

Section 8.7: LinearLayout

The LinearLayout is a `ViewGroup` that arranges its children in a single column or a single row. The orientation can be set by calling the method `setOrientation()` or using the xml attribute `android:orientation`.

1. **Vertical orientation** : `android:orientation="vertical"`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

    <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:text="@string/app_name" />

    <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:text="@android:string/cancel" />

</LinearLayout>
```

Here is a screenshot how this will look like:

2. **Horizontal orientation** : `android:orientation="horizontal"`

```
<TextView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="@string/app_name" />

<TextView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="@android:string/cancel" />
```

The `LinearLayout` also supports assigning a weight to individual children with the `android:layout_weight` attribute.

Section 8.8: RelativeLayout

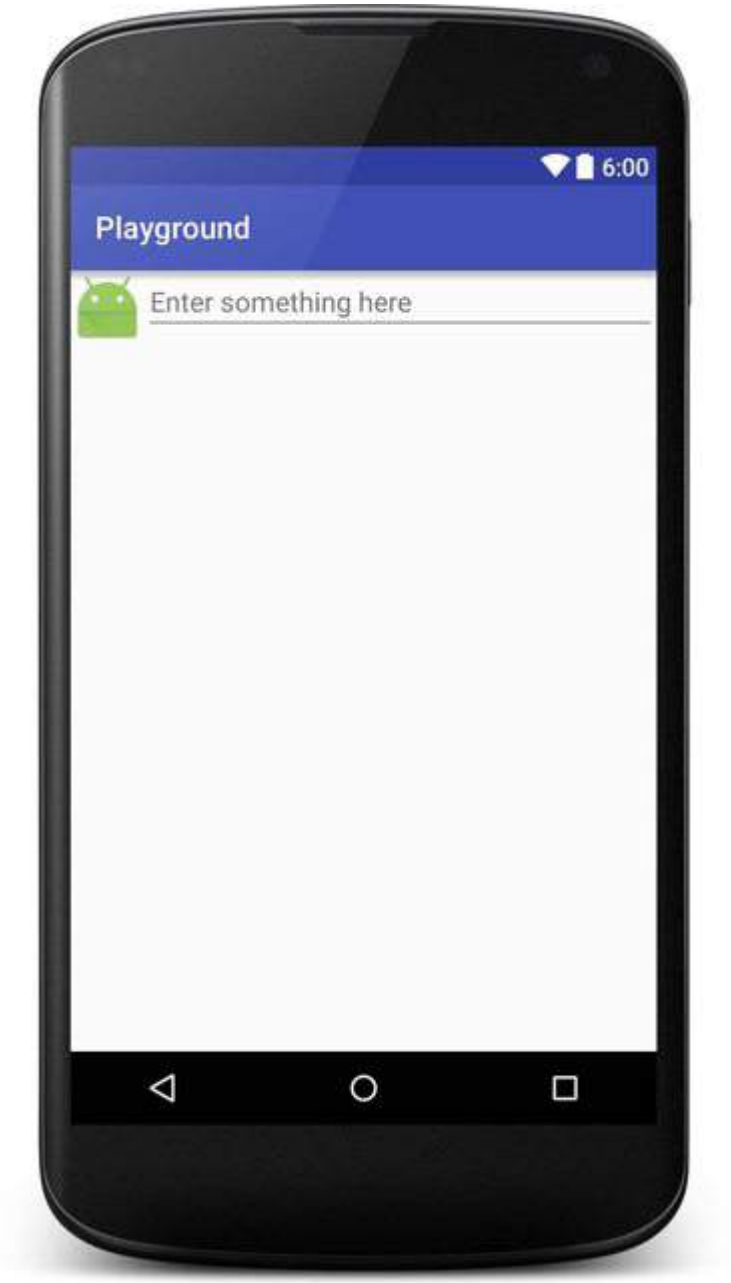
`RelativeLayout` is a `ViewGroup` that displays child views in relative positions. By default, all child views are drawn at

the top-left of the layout, so you must define the position of each view using the various layout properties available from `RelativeLayout.LayoutParams`. The value for each layout property is either a boolean to enable a layout position relative to the parent RelativeLayout or an ID that references another view in the layout against which the view should be positioned.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

    <ImageView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:id="@+id/imageView"
       android:src="@mipmap/ic_launcher" />

    <EditText
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:id="@+id/editText"
       android:layout_toRightOf="@+id/imageView"
       android:layout_toEndOf="@+id/imageView"
       android:hint="@string/hint" />

</RelativeLayout>
```

Here is a screenshot how this will look like:



Section 8.9: FrameLayout

`FrameLayout` is designed to block out an area on the screen to display a single item. You can, however, add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the android:layout_gravity attribute.

Generally, `FrameLayout` is used to hold a single child view. Common use cases are creating place holders for inflating `Fragments` in `Activity`, overlapping views or applying foreground to the views.

Example:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

    <ImageView
       android:src="@drawable/nougat"
       android:scaleType="fitCenter"
       android:layout_height="match_parent"
       android:layout_width="match_parent"/>
```

```
    <TextView
       android:text="FrameLayout Example"
       android:textSize="30sp"
       android:textStyle="bold"
       android:layout_height="match_parent"
       android:layout_width="match_parent"
       android:gravity="center"/>

</FrameLayout>
```

It will look like this:

Section 8.10: GridLayout

GridLayout, as the name suggests is a layout used to arrange Views in a grid. A GridLayout divides itself into columns and rows. As you can see in the example below, the amount of columns and/or rows is specified by the properties `columnCount` and `rowCount`. Adding Views to this layout will add the first view to the first column, the second view to the second column, and the third view to the first column of the second row.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
```

```
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:columnCount="2"
    android:rowCount="2">

     <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/first"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

     <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/second"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

     <TextView
        android:layout_width="@dimen/fixed"
        android:layout_height="wrap_content"
        android:text="@string/third"
        android:background="@color/colorPrimary"
        android:layout_margin="@dimen/default_margin" />

</GridLayout>
```

Section 8.11: CoordinatorLayout

Version ≥ 2.3

The `CoordinatorLayout` is a container somewhat similar to `FrameLayout` but with extra capabilities, it is called super-powered `FrameLayout` in the official documentation.

By attaching a `CoordinatorLayout.Behavior` to a direct child of CoordinatorLayout, you'll be able to intercept touch events, window insets, measurement, layout, and nested scrolling.

In order to use it, you will first have to add a dependency for the support library in your gradle file:

```
compile 'com.android.support:design:25.3.1'
```

The number of the latest version of the library may be found here

One practical use case of the `CoordinatorLayout` is creating a view with a `FloatingActionButton`. In this specific case, we will create a `RecyclerView` with a `SwipeRefreshLayout` and a `FloatingActionButton` on top of that. Here's how you can do that:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:id="@+id/coord_layout"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="horizontal">

    <android.support.v4.widget.SwipeRefreshLayout
       android:id="@+id/swipe_refresh_layout"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

        <android.support.v7.widget.RecyclerView
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:id="@+id/recycler_view"/>

    </android.support.v4.widget.SwipeRefreshLayout>

    <android.support.design.widget.FloatingActionButton
       android:id="@+id/fab"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_margin="16dp"
       android:clickable="true"
       android:color="@color/colorAccent"
       android:src="@mipmap/ic_add_white"
       android:layout_gravity="end|bottom"
       app:layout_anchorGravity="bottom|right|end"/>

</android.support.design.widget.CoordinatorLayout>
```

Notice how the FloatingActionButton is anchored to the CoordinatorLayout with

```
app:layout_anchor="@id/coord_layout"
```

Chapter 9: ConstraintLayout

Parameter	Details
child	The `View` to be added to the layout
index	The index of the `View` in the layout hierarchy
params	The `LayoutParams` of the `View`
attrs	The `AttributeSet` that defines the `LayoutParams`
view	The `View` that has been added or removed
changed	Indicates if this `View` has changed size or position
left	The left position, relative to the parent `View`
top	The top position, relative to the parent `View`
right	The right position, relative to the parent `View`
bottom	The bottom position, relative to the parent `View`
widthMeasureSpec	The horizontal space requirements imposed by the parent `View`
heightMeasureSpec	The vertical space requirements imposed by the parent `View`
layoutDirection	-
a	-
widthAttr	-
heightAttr	-

`ConstraintLayout` is a `ViewGroup` which allows you to position and size widgets in a flexible way. It is compatible with Android 2.3 (API level 9) and higher.

It allows you to create large and complex layouts with a flat view hierarchy. It is similar to `RelativeLayout` in that all views are laid out according to relationships between sibling views and the parent layout, but it's more flexible than `RelativeLayout` and easier to use with Android Studio's Layout Editor.

Section 9.1: Adding ConstraintLayout to your project

To work with ConstraintLayout, you need Android Studio Version 2.2 or newer and have at least version 32 (or higher) of Android Support Repository.

1. Add the Constraint Layout library as a dependency in your `build.gradle` file:

```
dependencies {
   compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

2. Sync project

To add a new constraint layout to your project:

1. **Right-click** on your module's layout directory, then click **`New`** > `XML` > `Layout XML`.
2. Enter a **name** for the layout and enter `"android.support.constraint.ConstraintLayout"` for the Root Tag.
3. Click **Finish**.

Otherwise just add in a layout file:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</android.support.constraint.ConstraintLayout>
```

Section 9.2: Chains

Since `ConstraintLayout` alpha 9, **Chains** are available. A **Chain** is a set of views inside a `ConstraintLayout` that are connected in a bi-directional way between them, i.e **A** connected to **B** with a constraint, and **B** connected to **A** with another constraint.

Example:

```
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- this view is linked to the bottomTextView -->
    <TextView
        android:id="@+id/topTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        app:layout_constraintBottom_toTopOf="@+id/bottomTextView"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainPacked="true" />

    <!-- this view is linked to the topTextView at the same time -->
    <TextView
        android:id="@+id/bottomTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bottom\nMkay"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/topTextView" />

</android.support.constraint.ConstraintLayout>
```

In this example, the two views are positioned one under another and both of them are centered vertically. You may change the vertical position of these views by adjusting the chain's **bias**. Add the following code to the first element of a chain:

```
app:layout_constraintVertical_bias="0.2"
```

In a vertical chain, the first element is a top-most view, and in a horizontal chain it is the left-most view. The first element defines the whole chain's behavior.

Chains are a new feature and are updated frequently. Here is an official Android Documentation on Chains.

Chapter 10: TextInputLayout

TextInputLayout was introduced to display the floating label on EditText. The EditText has to be wrapped by TextInputLayout in order to display the floating label.

Section 10.1: Basic usage

It is the basic usage of the `TextInputLayout`.
Make sure to add the dependency in the `build.gradle` file as described in the remarks section.

Example:

```
<android.support.design.widget.TextInputLayout
       android:layout_width="match_parent"
       android:layout_height="wrap_content">

    <EditText
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:hint="@string/username"/>

</android.support.design.widget.TextInputLayout>
```

Section 10.2: Password Visibility Toggles

With an input password type, you can also enable an icon that can show or hide the entire text using the `passwordToggleEnabled` attribute.

You can also customize same default using these attributes:

- `passwordToggleDrawable`: to change the default eye icon
- `passwordToggleTint`: to apply a tint to the password visibility toggle drawable.
- `passwordToggleTintMode`: to specify the blending mode used to apply the background tint.

Example:

```
<android.support.design.widget.TextInputLayout
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       app:passwordToggleContentDescription="@string/description"
       app:passwordToggleDrawable="@drawable/another_toggle_drawable"
       app:passwordToggleEnabled="true">

            <EditText/>

</android.support.design.widget.TextInputLayout>
```

Section 10.3: Adding Character Counting

The TextInputLayout has a character counter for an EditText defined within it.
The counter will be rendered below the EditText.

Just use the `setCounterEnabled()` and `setCounterMaxLength` methods:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
```

```
til.setCounterEnabled(true);
til.setCounterMaxLength(15);
```

or the `app:counterEnabled` and `app:counterMaxLength` attributes in the xml.

```
<android.support.design.widget.TextInputLayout
   app:counterEnabled="true"
   app:counterMaxLength="15">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

Section 10.4: Handling Errors

You can use the `TextInputLayout` to display error messages according to the material design guidelines using the `setError` and `setErrorEnabled`methods.

In order to show the error below the EditText use:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setErrorEnabled(true);
til.setError("You need to enter a name");
```

To enable error in the `TextInputLayout` you can eithr use `app:errorEnabled="true"` in xml or `til.setErrorEnabled(true);` as shown above.

You will obtain:

Section 10.5: Customizing the appearance of the TextInputLayout

You can customize the appearance of the `TextInputLayout` and its embedded `EditText`by defining custom styles in your `styles.xml`. The defined styles can either be added as styles or themes to your `TextInputLayout`.

Example for customizing the hint appearance:

`styles.xml`:

```
<!--Floating label text style-->
<style name="MyHintStyle" parent="TextAppearance.AppCompat.Small">
    <item name="android:textColor">@color/black</item>
</style>

<!--Input field style-->
<style name="MyEditText" parent="Theme.AppCompat.Light">
    <item name="colorControlNormal">@color/indigo</item>
    <item name="colorControlActivated">@color/pink</item>
</style>
```

To Apply Style update your TextInputLayout And EditText as follows

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:hintTextAppearance="@style/MyHintStyle">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/Title"
        android:theme="@style/MyEditText" />

</android.support.design.widget.TextInputLayout>
```

Example to customize the accent color of the `TextInputLayout`. The accent color affects the color of the baseline of the `EditText` and the text color for the floating hint text:

`styles.xml`:

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

layout file:

```
<android.support.design.widget.TextInputLayout
            android:id="@+id/textInputLayout_password"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/TextInputLayoutWithPrimaryColor">

            <android.support.design.widget.TextInputEditText
                android:id="@+id/textInputEditText_password"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="@string/login_hint_password"
                android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>
```

Section 10.6: TextInputEditText

The `TextInputEditText` is an `EditText` with an extra fix to display a hint in the IME when in 'extract' mode.

The **Extract mode** is the mode that the keyboard editor switches to when you click on an EditText when the space is too small (for example landscape on a smartphone).
In this case, using an `EditText` while you are editing the text you can see that the IME doesn't give you a hint of what you're editing

The `TextInputEditText` fixes this issue providing hint text while the user's device's IME is in Extract mode.

Example:

```
<android.support.design.widget.TextInputLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:hint="Description"
```

```
    >
     <android.support.design.widget.TextInputEditText
        android:id="@+id/description"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</android.support.design.widget.TextInputLayout>
```

Chapter 11: CoordinatorLayout and Behaviors

The CoordinatorLayout is a super-powered FrameLayout and goal of this ViewGroup is to coordinate the views that are inside it.

The main appeal of the CoordinatorLayout is its ability to coordinate the animations and transitions of the views within the XML file itself.

CoordinatorLayout is intended for two primary use cases:

:As a top-level application decor or chrome layout

:As a container for a specific interaction with one or more child views

Section 11.1: Creating a simple Behavior

To create a `Behavior` just extend the `CoordinatorLayout.Behavior` class.

Extend the CoordinatorLayout.Behavior

Example:

```
public class MyBehavior<V extends View> extends CoordinatorLayout.Behavior<V> {

      /**
       * Default constructor.
       */
      public MyBehavior() {
      }

      /**
       * Default constructor for inflating a MyBehavior from layout.
       *
       * @param context The {@link Context}.
       * @param attrs The {@link AttributeSet}.
       */
      public MyBehavior(Context context, AttributeSet attrs) {
        super(context, attrs);
      }
}
```

This behavior need to be attached to a child View of a `CoordinatorLayout` to be called.

Attach a Behavior programmatically

```
MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) view.getLayoutParams();
params.setBehavior(myBehavior);
```

Attach a Behavior in XML

You can use the `layout_behavior` attribute to attach the behavior in XML:

```
<View
 android:layout_height="...."
 android:layout_width="...."
```

```
app:layout_behavior=".MyBehavior" />
```

Attach a Behavior automatically

If you are working with a custom view you can attach the behavior using the `@CoordinatorLayout.DefaultBehavior` annotation:

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {

}
```

Section 11.2: Using the SwipeDismissBehavior

The `SwipeDismissBehavior` works on any View and implements the functionality of swipe to dismiss in our layouts with a `CoordinatorLayout`.

Just use:

```
        final SwipeDismissBehavior<MyView> swipe = new SwipeDismissBehavior();

        //Sets the swipe direction for this behavior.
        swipe.setSwipeDirection(
            SwipeDismissBehavior.SWIPE_DIRECTION_ANY);

        //Set the listener to be used when a dismiss event occurs
        swipe.setListener(
            new SwipeDismissBehavior.OnDismissListener() {
            @Override public void onDismiss(View view) {
                //......
            }

            @Override
            public void onDragStateChanged(int state) {
                //......
            }
        });

        //Attach the SwipeDismissBehavior to a view
        LayoutParams coordinatorParams =
            (LayoutParams) mView.getLayoutParams();
        coordinatorParams.setBehavior(swipe);
```

Section 11.3: Create dependencies between Views

You can use the `CoordinatorLayout.Behavior` to create dependencies between views. You can anchor a `View` to another `View` by:

- using the `layout_anchor` attribute.
- creating a custom `Behavior` and implementing the `layoutDependsOn` method returning **`true`**.

For example, in order to create a `Behavior` for moving an `ImageView` when another one is moved (example Toolbar), perform the following steps:

- Create the custom Behavior:

```
public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
```

- Override the `layoutDependsOn` method returning **`true`**. This method is called every time a change occurs to the layout:

```
@Override
public boolean layoutDependsOn(CoordinatorLayout parent,
        ImageView child, View dependency) {
    // Returns true to add a dependency.
    return dependency instanceof Toolbar;
}
```

- Whenever the method `layoutDependsOn` returns **`true`** the method `onDependentViewChanged` is called:

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, ImageView child, View
dependency) {
    // Implement here animations, translations, or movements; always related to the provided
dependency.
    float translationY = Math.min(0, dependency.getTranslationY() - dependency.getHeight());
    child.setTranslationY(translationY);
}
```

Chapter 12: TabLayout

Section 12.1: Using a TabLayout without a ViewPager

Most of the time a `TabLayout` is used together with a ViewPager, in order to get the swipe functionality that comes with it.

It is possible to use a `TabLayout` without a `ViewPager` by using a `TabLayout.OnTabSelectedListener`.

First, add a `TabLayout` to your activity's XML file:

```
<android.support.design.widget.TabLayout
   android:layout_height="wrap_content"
   android:layout_width="match_parent"
   android:id="@+id/tabLayout" />
```

For navigation within an `Activity`, manually populate the UI based on the tab selected.

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        int position = tab.getPosition();
        switch (tab.getPosition()) {
            case 1:
                getSupportFragmentManager().beginTransaction()
                    .replace(R.id.fragment_container, new ChildFragment()).commit();
                break;
            // Continue for each tab in TabLayout
    }

    @Override
    public void onTabUnselected(TabLayout.Tab tab) {

    }

    @Override
    public void onTabReselected(TabLayout.Tab tab) {

    }
});
```

Chapter 13: ViewPager

ViewPager is a Layout manager that allows the user to flip left and right through pages of data. It is most often used in conjunction with Fragment, which is a convenient way to supply and manage the lifecycle of each page.

Section 13.1: ViewPager with a dots indicator

All we need are: ViewPager, TabLayout and 2 drawables for selected and default dots.

Firstly, we have to add `TabLayout` to our screen layout, and connect it with `ViewPager`. We can do this in two ways:

Nested TabLayout in ViewPager

```
<android.support.v4.view.ViewPager
   android:id="@+id/photos_viewpager"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

    <android.support.design.widget.TabLayout
       android:layout_width="match_parent"
       android:layout_height="wrap_content"/>
</android.support.v4.view.ViewPager>
```

> In this case `TabLayout` will be automatically connected with `ViewPager`, but `TabLayout` will be next to `ViewPager`, not over him.

Separate TabLayout

```
<android.support.v4.view.ViewPager
   android:id="@+id/photos_viewpager"
   android:layout_width="match_parent"
   android:layout_height="match_parent"/>
```

```
<android.support.design.widget.TabLayout
   android:id="@+id/tab_layout"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"/>
```

> In this case, we can put `TabLayout` anywhere, but we have to connect `TabLayout` with `ViewPager` programmatically

```
ViewPager pager = (ViewPager) view.findViewById(R.id.photos_viewpager);
PagerAdapter adapter = new PhotosAdapter(getChildFragmentManager(), photosUrl);
pager.setAdapter(adapter);

TabLayout tabLayout = (TabLayout) view.findViewById(R.id.tab_layout);
tabLayout.setupWithViewPager(pager, true);
```

Once we created our layout, we have to prepare our dots. So we create three files: `selected_dot.xml`, `default_dot.xml` and `tab_selector.xml`.

selected_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape
           android:innerRadius="0dp"
           android:shape="ring"
           android:thickness="8dp"
           android:useLevel="false">
            <solid android:color="@color/colorAccent"/>
        </shape>
    </item>
</layer-list>
```

default_dot.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape
            android:innerRadius="0dp"
            android:shape="ring"
            android:thickness="8dp"
            android:useLevel="false">
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
</layer-list>
```

tab_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@drawable/selected_dot"
          android:state_selected="true"/>

    <item android:drawable="@drawable/default_dot"/>
```

```
</selector>
```

Now we need to add only 3 lines of code to `TabLayout` in our xml layout and you're done.

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

Section 13.2: Basic ViewPager usage with fragments

A `ViewPager` allows to show multiple fragments in an activity that can be navigated by either fliping left or right. A `ViewPager` needs to be feed of either Views or Fragments by using a `PagerAdapter`.

There are however two more specific implementations that you will find most useful in case of using Fragments which are `FragmentPagerAdapter` and `FragmentStatePagerAdapter`. When a Fragment needs to be instantiated for the first time, `getItem(position)` will be called for each position that needs instantiating. The `getCount()` method will return the total number of pages so the `ViewPager` knows how many Fragments need to be shown.

Both `FragmentPagerAdapter` and `FragmentStatePagerAdapter` keep a cache of the Fragments that the `ViewPager` will need to show. By default the `ViewPager` will try to store a maximum of 3 Fragments that correspond to the currently visible Fragment, and the ones next to the right and left. Also `FragmentStatePagerAdapter` will keep the state of each of your fragments.

Be aware that both implementations assume your fragments will keep their positions, so if you keep a list of the fragments instead of having a static number of them as you can see in the `getItem()` method, you will need to create a subclass of `PagerAdapter` and override at least `instantiateItem()`,`destroyItem()` and `getItemPosition()`methods.

Just add a ViewPager in your layout as described in the basic example:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
       android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Then define the adapter that will determine how many pages exist and which fragment to display for each page of the adapter.

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myActivityLayout);

        //Apply the Adapter
        mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.view_pager);
        mViewPager.setAdapter(mFragmentAdapter);
    }
```

```
    private class MyPagerAdapter extends FragmentPagerAdapter{

        public MyPagerAdapter(FragmentManager supportFragmentManager) {
            super(supportFragmentManager);
        }

        // Returns the fragment to display for that page
        @Override
        public Fragment getItem(int position) {
            switch(position) {
                case 0:
                    return new Fragment1();

                case 1:
                    return new Fragment2();

                case 2:
                    return new Fragment3();

                default:
                    return null;
            }
        }

        // Returns total number of pages
        @Override
        public int getCount() {
            return 3;
        }

    }
}
```

Version ≥ 3.2.x

If you are using `android.app.Fragment` you have to add this dependency:

```
compile 'com.android.support:support-v13:25.3.1'
```

If you are using `android.support.v4.app.Fragment` you have to add this dependency:

```
compile 'com.android.support:support-fragment:25.3.1'
```

Section 13.3: ViewPager with PreferenceFragment

Until recently, using `android.support.v4.app.FragmentPagerAdapter` would prevent the usage of a `PreferenceFragment` as one of the Fragments used in the FragmentPagerAdapter.

The latest versions of the support v7 library now include the `PreferenceFragmentCompat` class, which will work with a ViewPager and the v4 version of FragmentPagerAdapter.

Example Fragment that extends `PreferenceFragmentCompat`:

```
import android.os.Bundle;
import android.support.v7.preference.PreferenceFragmentCompat;
import android.view.View;

public class MySettingsPrefFragment extends PreferenceFragmentCompat {

    public MySettingsPrefFragment() {
```

```
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.fragment_settings_pref);
    }

    @Override
    public void onCreatePreferences(Bundle bundle, String s) {

    }
}
```

You can now use this Fragment in a `android.support.v4.app.FragmentPagerAdapter` subclass:

```
private class PagerAdapterWithSettings extends FragmentPagerAdapter {

    public PagerAdapterWithSettings(FragmentManager supportFragmentManager) {
        super(supportFragmentManager);
    }

    @Override
    public Fragment getItem(int position) {
        switch(position) {
            case 0:
                return new FragmentOne();

            case 1:
                return new FragmentTwo();

            case 2:
                return new MySettingsPrefFragment();

            default:
                return null;
        }
    }

    // .......

}
```

Section 13.4: Adding a ViewPager

Make sure the following dependency is added to your app's `build.gradle` file under dependencies:

```
compile 'com.android.support:support-core-ui:25.3.0'
```

Then add the `ViewPager` to your activity layout:

```
<android.support.v4.view.ViewPager
   android:id="@+id/viewpager"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   />
```

Then define your `PagerAdapter`:

```
public class MyPagerAdapter extends PagerAdapter {

    private Context mContext;

    public CustomPagerAdapter(Context context) {
        mContext = context;
    }

    @Override
    public Object instantiateItem(ViewGroup collection, int position) {

        // Create the page for the given position. For example:
        LayoutInflater inflater = LayoutInflater.from(mContext);
        ViewGroup layout = (ViewGroup) inflater.inflate(R.layout.xxxx, collection, false);
        collection.addView(layout);
        return layout;
    }

    @Override
    public void destroyItem(ViewGroup collection, int position, Object view) {
        // Remove a page for the given position. For example:
        collection.removeView((View) view);
    }

    @Override
    public int getCount() {
        //Return the number of views available.
        return numberOfPages;
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        // Determines whether a page View is associated with a specific key object
        // as returned by instantiateItem(ViewGroup, int). For example:
        return view == object;
    }
}
```

Finally setup the `ViewPager` in your Activity:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
        viewPager.setAdapter(new MyPagerAdapter(this));
    }
}
```

Section 13.5: Setup OnPageChangeListener

If you need to listen for changes to the page selected you can implement the `ViewPager.OnPageChangeListener` listener on the ViewPager:

```
viewPager.addOnPageChangeListener(new OnPageChangeListener() {

    // This method will be invoked when a new page becomes selected. Animation is not necessarily
```

```
complete.
    @Override
    public void onPageSelected(int position) {
         // Your code
    }

    // This method will be invoked when the current page is scrolled, either as part of
    // a programmatically initiated smooth scroll or a user initiated touch scroll.
    @Override
    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
         // Your code
    }

    // Called when the scroll state changes. Useful for discovering when the user begins
    // dragging, when the pager is automatically settling to the current page,
    // or when it is fully stopped/idle.
    @Override
    public void onPageScrollStateChanged(int state) {
         // Your code
    }
});
```

Section 13.6: ViewPager with TabLayout

A `TabLayout` can be used for easier navigation.
You can set the tabs for each fragment in your adapter by using `TabLayout.newTab()` method but there is another more convenient and easier method for this task which is `TabLayout.setupWithViewPager()`.

This method will sync by creating and removing tabs according to the contents of the adapter associated with your `ViewPager` each time you call it.
Also, it will set a callback so each time the user flips the page, the corresponding tab will be selected.

Just define a layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>

    <android.support.design.widget.TabLayout
       android:id="@+id/tabs"
       app:tabMode="scrollable" />

    <android.support.v4.view.ViewPager
       android:id="@+id/viewpager"
       android:layout_width="match_parent"
       android:layout_height="0px"
       android:layout_weight="1" />

</LinearLayout>
```

Then implement the `FragmentPagerAdapter` and apply it to the `ViewPager`:

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPager;
    private TabLayout mTabLayout;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.myActivityLayout);

        // Get the ViewPager and apply the PagerAdapter
        mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.view_pager);
        mViewPager.setAdapter(mFragmentAdapter);

        // link the tabLayout and the viewpager together
        mTabLayout = (TabLayout) findViewById(R.id.tab_layout);
        mTabLayout.setupWithViewPager(mViewPager);
    }

    private class MyPagerAdapter extends FragmentPagerAdapter{

        public MyPagerAdapter(FragmentManager supportFragmentManager) {
            super(supportFragmentManager);
        }

         // Returns the fragment to display for that page
        @Override
        public Fragment getItem(int position) {
            switch(position) {
                case 0:
                    return new Fragment1();

                case 1:
                    return new Fragment2();

                case 2:
                    return new Fragment3();

                default:
                    return null;
            }
        }

        // Will be displayed as the tab's label
        @Override
        public CharSequence getPageTitle(int position) {
            switch(position) {
                case 0:
                    return "Fragment 1 title";

                case 1:
                    return "Fragment 2 title";

                case 2:
                    return "Fragment 3 title";

                default:
                    return null;
            }
        }

        // Returns total number of pages
        @Override
        public int getCount() {
            return 3;
        }
```

```
    }
}
```

Chapter 14: CardView

Parameter	Details
cardBackgroundColor	Background color for CardView.
cardCornerRadius	Corner radius for CardView.
cardElevation	Elevation for CardView.
cardMaxElevation	Maximum Elevation for CardView.
cardPreventCornerOverlap	Add padding to CardView on v20 and before to prevent intersections between the Card content and rounded corners.
cardUseCompatPadding	Add padding in API v21+ as well to have the same measurements with previous versions. May be a boolean value, such as "true" or "false".
contentPadding	Inner padding between the edges of the Card and children of the CardView.
contentPaddingBottom	Inner padding between the bottom edge of the Card and children of the CardView.
contentPaddingLeft	Inner padding between the left edge of the Card and children of the CardView.
contentPaddingRight	Elevation for CardView.
cardElevation	Inner padding between the right edge of the Card and children of the CardView.
contentPaddingTop	Inner padding between the top edge of the Card and children of the CardView.

A FrameLayout with a rounded corner background and shadow.

CardView uses elevation property on Lollipop for shadows and falls back to a custom emulated shadow implementation on older platforms.

Due to expensive nature of rounded corner clipping, on platforms before Lollipop, CardView does not clip its children that intersect with rounded corners. Instead, it adds padding to avoid such intersection (See setPreventCornerOverlap(boolean) to change this behavior).

Section 14.1: Getting Started with CardView

`CardView` is a member of the Android Support Library, and provides a layout for cards.

To add `CardView` to your project, add the following line to your `build.gradle` dependencies.

```
compile 'com.android.support:cardview-v7:25.1.1'
```

A number of the latest version may be found here

In your layout you can then add the following to get a card.

```
<android.support.v7.widget.CardView
       xmlns:card_view="http://schemas.android.com/apk/res-auto"
       android:layout_width="match_parent"
       android:layout_height="wrap_content">

        <!-- one child layout containing other layouts or views -->

</android.support.v7.widget.CardView>
```

You can then add other layouts inside this and they will be encompassed in a card.

Also, CardView can be populated with any UI element and manipulated from code.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/card_view"
    android:layout_margin="5dp"
    card_view:cardBackgroundColor="#81C784"
    card_view:cardCornerRadius="12dp"
    card_view:cardElevation="3dp"
    card_view:contentPadding="4dp" >

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp" >

        <ImageView
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:id="@+id/item_image"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginRight="16dp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/item_title"
            android:layout_toRightOf="@+id/item_image"
            android:layout_alignParentTop="true"
            android:textSize="30sp"
            />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/item_detail"
            android:layout_toRightOf="@+id/item_image"
            android:layout_below="@+id/item_title"
            />

    </RelativeLayout>
</android.support.v7.widget.CardView>
```

Section 14.2: Adding Ripple animation

To enable the ripple animation in a CardView, add the following attributes:

```
<android.support.v7.widget.CardView
  ...
  android:clickable="true"
  android:foreground="?android:attr/selectableItemBackground">
  ...
</android.support.v7.widget.CardView>
```

Section 14.3: Customizing the CardView

CardView provides a default elevation and corner radius so that cards have a consistent appearance across the

platforms.

You can customize these default values using these attributes in the xml file:

1. `card_view:cardElevation` attribute add elevation in CardView.
2. `card_view:cardBackgroundColor` attribute is used to customize background color of CardView's background(you can give any color).
3. `card_view:cardCornerRadius` attribute is used to curve 4 edges of CardView
4. `card_view:contentPadding` attribute add padding between card and children of card

Note: card_view is a namespace defined in topmost parent layout view.
xmlns:card_view="http://schemas.android.com/apk/res-auto"

Here an example:

```
<android.support.v7.widget.CardView
       xmlns:card_view="http://schemas.android.com/apk/res-auto"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       card_view:cardElevation="4dp"
       card_view:cardBackgroundColor="@android:color/white"
       card_view:cardCornerRadius="8dp"
       card_view:contentPadding="16dp">

        <!-- one child layout containing other layouts or views -->

</android.support.v7.widget.CardView>
```

You can also do it programmatically using:

```
card.setCardBackgroundColor(....);
card.setCardElevation(...);
card.setRadius(....);
card.setContentPadding();
```

Check the official javadoc for additional properties.

Section 14.4: Using Images as Background in CardView (Pre-Lollipop device issues)

While using Image/Colour as an background in a CardView, You might end up with slight white paddings (If default Card colour is white) on the edges. This occurs due to the default rounded corners in the Card View. Here is how to avoid those margins in Pre-lollipop devices.

We need to use an attribute `card_view:cardPreventCornerOverlap="false"` in the CardView. 1). In XML use the following snippet.

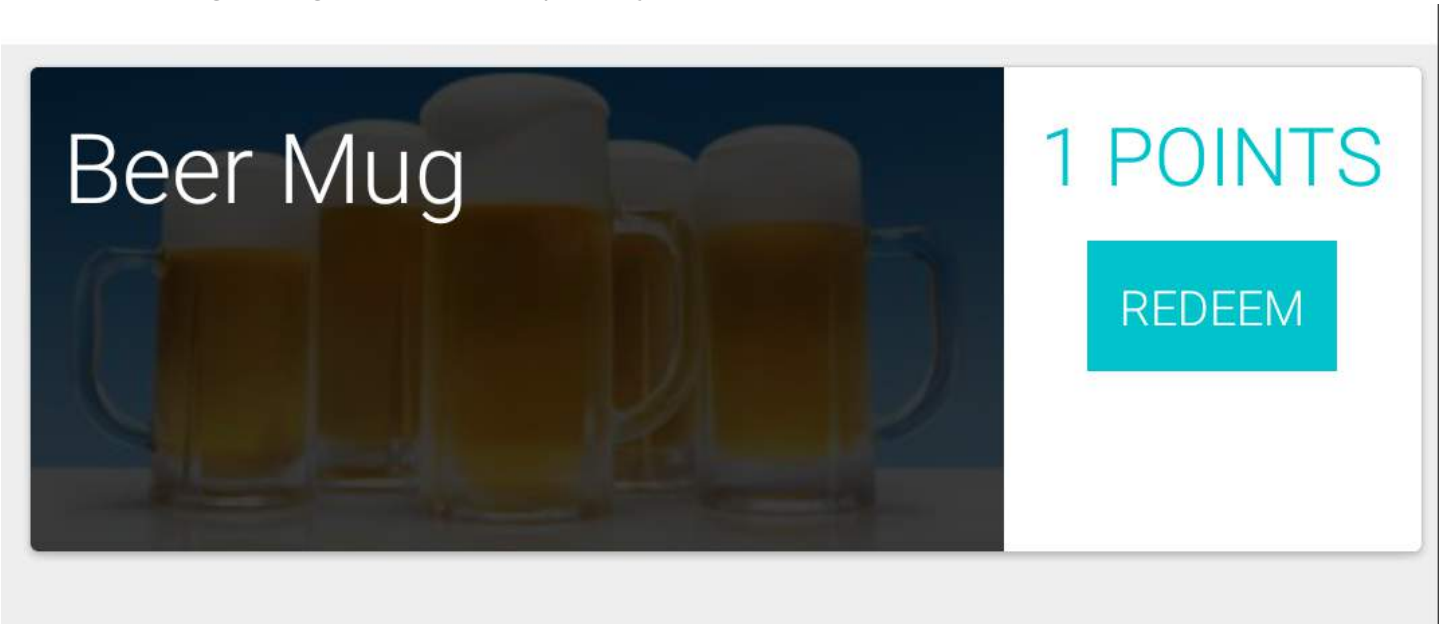
```
<android.support.v7.widget.CardView
   xmlns:card_view="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   card_view:cardPreventCornerOverlap="false"
   android:layout_height="wrap_content">
      <ImageView
           android:id="@+id/row_wallet_redeem_img"
           android:layout_width="match_parent"
           android:layout_height="match_parent"
           android:adjustViewBounds="true"
```

```
            android:scaleType="centerCrop"
            android:src="@drawable/bg_image" />

</android.support.v7.widget.CardView>
```

2. In Java like this `cardView.setPreventCornerOverlap(false)`.

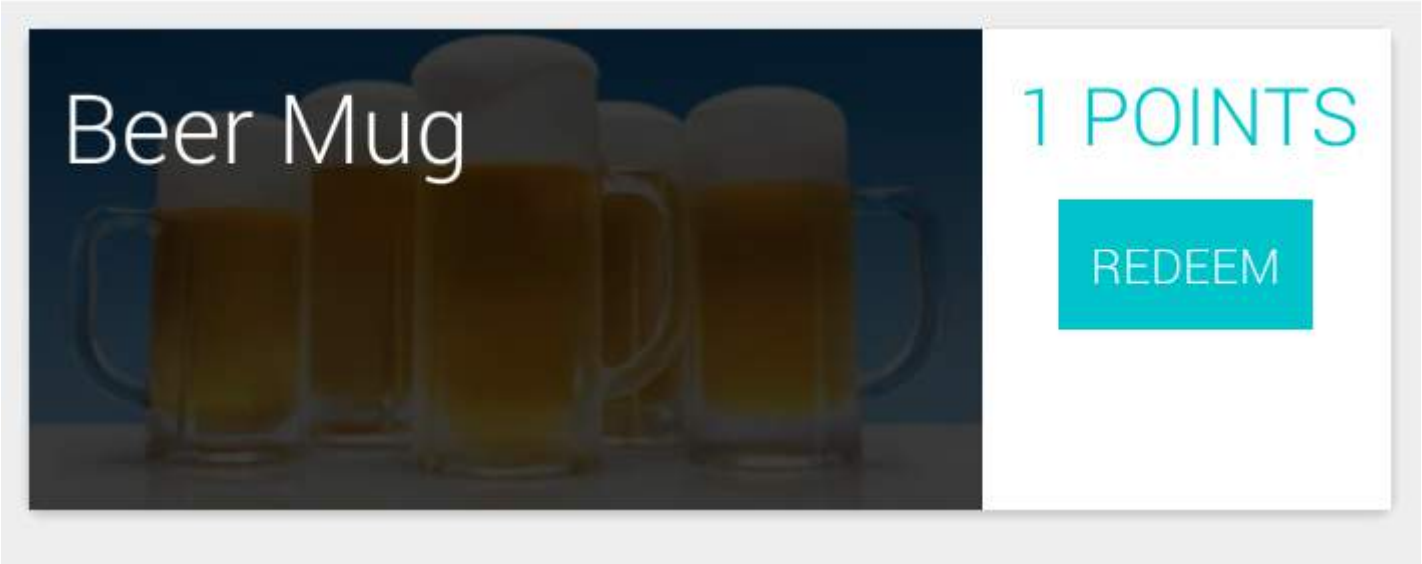
Doing so removes an unwanted padding on the Card's edges. Here are some visual examples related to this implementation.

1 Card with image background in API 21 (perfectly fine)



2 Card with image background in API 19 without attribute (notice the paddings around image)

3 FIXED Card with image background in API 19 with attribute `cardView.setPreventCornerOverlap(false)` (Issue now fixed)



Also read about this on Documentation here
Original SOF post here

Section 14.5: Animate CardView background color with TransitionDrawable

```
public void setCardColorTran(CardView card) {
    ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};
    TransitionDrawable trans = new TransitionDrawable(color);
    if(Build.VERSION.SDK_INT > Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1) {
        card.setBackground(trans);
    } else {
        card.setBackgroundDrawable(trans);
    }
    trans.startTransition(5000);
}
```

Chapter 15: NavigationView

Section 15.1: How to add the NavigationView

To use a NavigationView just add the dependency in the `build.gradle` file as described in the remarks section

Then add the `NavigationView` in the layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/drawer_layout"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:fitsSystemWindows="true"
   tools:openDrawer="start">

    <include
       layout="@layout/app_bar_main"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
       android:id="@+id/nav_view"
       android:layout_width="wrap_content"
       android:layout_height="match_parent"
       android:layout_gravity="start"
       app:headerLayout="@layout/nav_header_main"
       app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

res/layout/nav_header_main.xml: The view which will been displayed on the top of the drawer

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="@dimen/nav_header_height"
   android:background="@drawable/side_nav_bar"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:theme="@style/ThemeOverlay.AppCompat.Dark"
   android:orientation="vertical"
   android:gravity="bottom">

    <ImageView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:paddingTop="@dimen/nav_header_vertical_spacing"
       android:src="@android:drawable/sym_def_app_icon"
       android:id="@+id/imageView" />

    <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
```

```
        android:paddingTop="@dimen/nav_header_vertical_spacing"
        android:text="Android Studio"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="android.studio@android.com"
        android:id="@+id/textView" />

</LinearLayout>
```

res/layout/app_bar_main.xml An abstraction layer for the toolbar to separate it from the content:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="eu.rekisoft.playground.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

res/layout/content_main.xml The real content of the activity just for demo, here you would put your normal layout xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:showIn="@layout/app_bar_main"
    tools:context="eu.rekisoft.playground.MainActivity">

     <TextView
        android:text="Hello World!"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RelativeLayout>
```

Define your menu file as *res/menu/activity_main_drawer.xml*:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
           android:id="@+id/nav_camera"
           android:icon="@drawable/ic_menu_camera"
           android:title="Import" />
        <item
           android:id="@+id/nav_gallery"
           android:icon="@drawable/ic_menu_gallery"
           android:title="Gallery" />
        <item
           android:id="@+id/nav_slideshow"
           android:icon="@drawable/ic_menu_slideshow"
           android:title="Slideshow" />
        <item
           android:id="@+id/nav_manage"
           android:icon="@drawable/ic_menu_manage"
           android:title="Tools" />
    </group>

    <item android:title="Communicate">
        <menu>
            <item
               android:id="@+id/nav_share"
               android:icon="@drawable/ic_menu_share"
               android:title="Share" />
            <item
               android:id="@+id/nav_send"
               android:icon="@drawable/ic_menu_send"
               android:title="Send" />
        </menu>
    </item>

</menu>
```

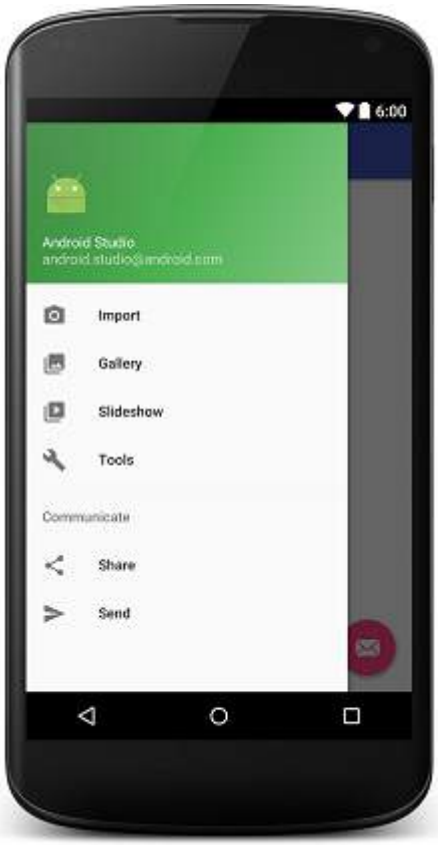
And finally the `java/main/eu/rekisoft/playground/MainActivity.java`:

```
public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                        .setAction("Action", null).show();
            }
        });

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.setDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        switch(item.getItemId()) {/*...*/}

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
```

```
            drawer.closeDrawer(GravityCompat.START);
            return true;
        }
    }
```

It will look like this:

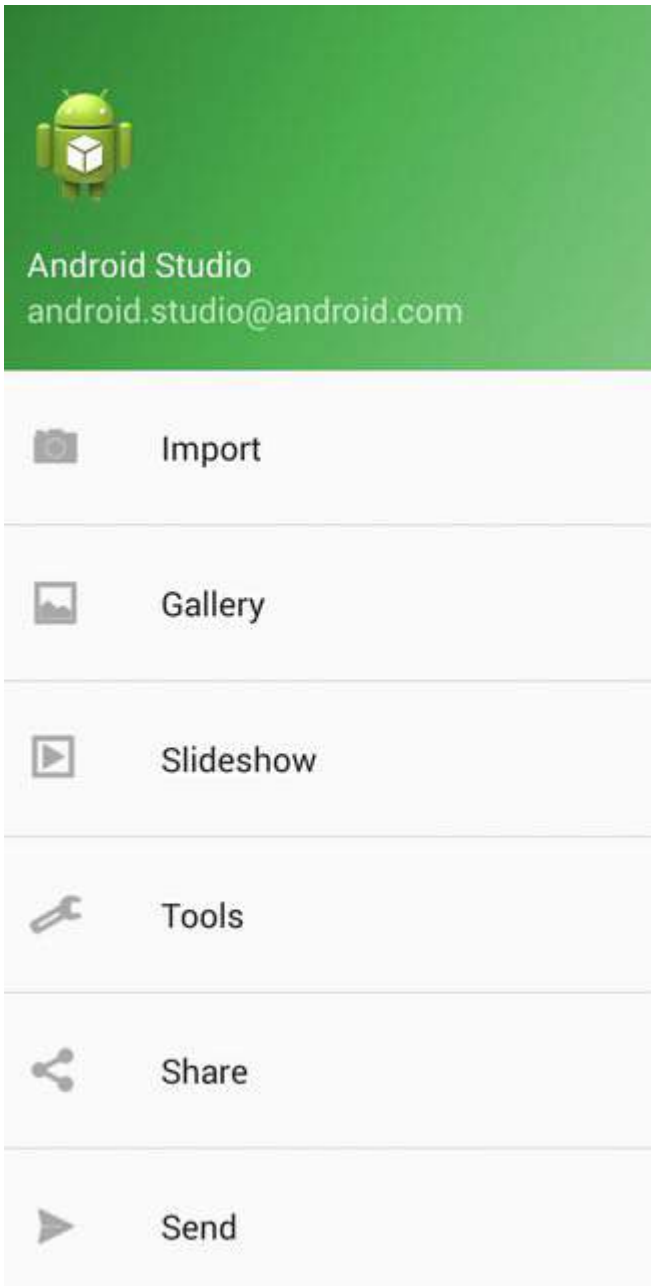


Section 15.2: Add underline in menu elements

Each group ends with a line separator. If each item in your menu has its own group you will achieve the desired graphical output. It will work only if your different groups have different `android:id`. Also, in `menu.xml` remember to mention `android:checkable="true"` for single item and `android:checkableBehavior="single"` for a group of items.

```
<?xml version="1.0" encoding="utf-8"?>
    <menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
         android:id="@+id/pos_item_help"
         android:checkable="true"
         android:title="Help" />
     <item
       android:id="@+id/pos_item_pos"
       android:checkable="true"
       android:title="POS" />

    <item
       android:id="@+id/pos_item_orders"
       android:checkable="true"
       android:title="Orders" />

       <group
           android:id="@+id/group"
           android:checkableBehavior="single">
```

```
            <item
            android:id="@+id/menu_nav_home"
            android:icon="@drawable/ic_home_black_24dp"
            android:title="@string/menu_nav_home" />
    </group>

    ......
    </menu>
```



Section 15.3: Add seperators to menu

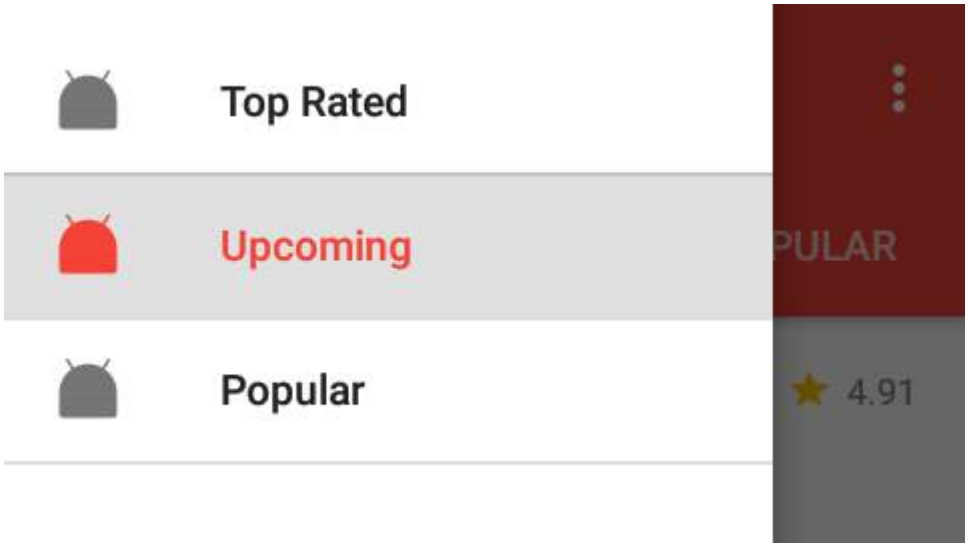
Access the RecyclerView in the NavigationView and add ItemDecoration to it.

```
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));
```

Code for DividerItemDecoration

```
public class DividerItemDecoration extends RecyclerView.ItemDecoration {

    private static final int[] ATTRS = new int[]{android.R.attr.listDivider};

    private Drawable mDivider;

    public DividerItemDecoration(Context context) {
        final TypedArray styledAttributes = context.obtainStyledAttributes(ATTRS);
        mDivider = styledAttributes.getDrawable(0);
        styledAttributes.recycle();
    }


    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        int left = parent.getPaddingLeft();
        int right = parent.getWidth() - parent.getPaddingRight();

        int childCount = parent.getChildCount();
        for (int i = 1; i < childCount; i++) {
            View child = parent.getChildAt(i);

            RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child.getLayoutParams();

            int top = child.getBottom() + params.bottomMargin;
            int bottom = top + mDivider.getIntrinsicHeight();

            mDivider.setBounds(left, top, right, bottom);
            mDivider.draw(c);
        }
    }
}
```

Preview:



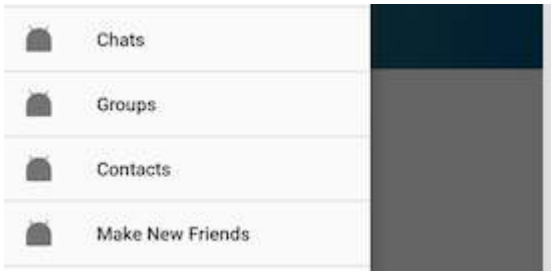
Section 15.4: Add menu Divider using default DividerItemDecoration

Just use default DividerItemDecoration class :

```
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
```

```
navMenuView.addItemDecoration(new DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

Preview :



Chapter 16: RecyclerView

Parameter	Detail
Adapter	A subclass of RecyclerView.Adapter responsible for providing views that represent items in a data set
Position	The position of a data item within an Adapter
Index	The index of an attached child view as used in a call to getChildAt(int). Contrast with Position
Binding	The process of preparing a child view to display data corresponding to a position within the adapter
Recycle (view)	A view previously used to display data for a specific adapter position may be placed in a cache for later reuse to display the same type of data again later. This can drastically improve performance by skipping initial layout inflation or construction
Scrap (view)	A child view that has entered into a temporarily detached state during layout. Scrap views may be reused without becoming fully detached from the parent RecyclerView, either unmodified if no rebinding is required or modified by the adapter if the view was considered dirty
Dirty (view)	A child view that must be rebound by the adapter before being displayed

RecyclerView is a more advanced version of List View with improved performance and additional features.

Section 16.1: Adding a RecyclerView

Add the dependency as described in the Remark section, then add a `RecyclerView` to your layout:

```
<android.support.v7.widget.RecyclerView
   android:id="@+id/my_recycler_view"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"/>
```

Once you have added a `RecyclerView` widget to your layout, obtain a handle to the object, connect it to a layout manager and attach an adapter for the data to be displayed:

```
mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);

// set a layout manager (LinearLayoutManager in this example)

mLayoutManager = new LinearLayoutManager(getApplicationContext());
mRecyclerView.setLayoutManager(mLayoutManager);

// specify an adapter
mAdapter = new MyAdapter(myDataset);
mRecyclerView.setAdapter(mAdapter);
```

Or simply setup layout manager from xml by adding this lines:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

If you know that changes in content of the `RecyclerView` won't change the layout size of the `RecyclerView`, use the following code to improve the performance of the component. If RecyclerView has a fixed size, it knows that RecyclerView itself will not resize due to its children, so it doesn't call request layout at all. It just handles the change itself. If invalidating whatever the parent is, the coordinator, layout, or whatever. (you can use this method even before setting `LayoutManager` and `Adapter`):

```
mRecyclerView.setHasFixedSize(true);
```

`RecyclerView` provides these built-in layout managers to use. So you can create a list, a grid and a staggered grid using `RecyclerView`:

1. LinearLayoutManager shows items in a vertical or horizontal scrolling list.
2. GridLayoutManager shows items in a grid.
3. StaggeredGridLayoutManager shows items in a staggered grid.

Section 16.2: Smoother loading of items

If the items in your `RecyclerView` load data from the network (commonly images) or carry out other processing, that can take a significant amount of time and you may end up with items on-screen but not fully loaded. To avoid this you can extend the existing `LinearLayoutManager` to preload a number of items before they become visible on-screen:

```
package com.example;

import android.content.Context;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.OrientationHelper;
import android.support.v7.widget.RecyclerView;

/**
 * A LinearLayoutManager that preloads items off-screen.
 * <p>
 * Preloading is useful in situations where items might take some time to load
 * fully, commonly because they have maps, images or other items that require
 * network requests to complete before they can be displayed.
 * <p>
 * By default, this layout will load a single additional page's worth of items,
 * a page being a pixel measure equivalent to the on-screen size of the
 * recycler view.  This can be altered using the relevant constructor, or
 * through the {@link #setPages(int)} method.
 */
public class PreLoadingLinearLayoutManager extends LinearLayoutManager {
  private int mPages = 1;
  private OrientationHelper mOrientationHelper;

  public PreLoadingLinearLayoutManager(final Context context) {
    super(context);
  }

  public PreLoadingLinearLayoutManager(final Context context, final int pages) {
    super(context);
    this.mPages = pages;
  }

  public PreLoadingLinearLayoutManager(final Context context, final int orientation, final boolean
reverseLayout) {
    super(context, orientation, reverseLayout);
  }

  @Override
  public void setOrientation(final int orientation) {
    super.setOrientation(orientation);
    mOrientationHelper = null;
  }

  /**
   * Set the number of pages of layout that will be preloaded off-screen,
```

```
     * a page being a pixel measure equivalent to the on-screen size of the
     * recycler view.
     * @param pages the number of pages; can be {@code 0} to disable preloading
     */
    public void setPages(final int pages) {
      this.mPages = pages;
    }

    @Override
    protected int getExtraLayoutSpace(final RecyclerView.State state) {
      if (mOrientationHelper == null) {
        mOrientationHelper = OrientationHelper.createOrientationHelper(this, getOrientation());
      }
      return mOrientationHelper.getTotalSpace() * mPages;
    }
}
```

Section 16.3: RecyclerView with DataBinding

Here is a generic ViewHolder class that you can use with any DataBinding layout. Here an instance of particular `ViewDataBinding` class is created using the inflated `View` object and `DataBindingUtil` utility class.

```
import android.databinding.DataBindingUtil;
import android.support.v7.widget.RecyclerView;
import android.view.View;

public class BindingViewHolder<T> extends RecyclerView.ViewHolder{

    private final T binding;

    public BindingViewHolder(View itemView) {
        super(itemView);
        binding = (T)DataBindingUtil.bind(itemView);
    }

    public T getBinding() {
        return binding;
    }
}
```

After creating this class you can use the `<layout>` in your layout file to enable databinding for that layout like this:

`file name: my_item.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
           name="item"
           type="ItemModel" />
    </data>

    <LinearLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent">

        <TextView
           android:layout_width="wrap_content"
           android:layout_height="match_parent"
```

```
            android:text="@{item.itemLabel}" />
    </LinearLayout>
</layout>
```

and here is your sample dataModel:

```
public class ItemModel {
    public String itemLabel;
}
```

By default, Android Data Binding library generates a `ViewDataBinding` class based on the layout file name, converting it to Pascal case and suffixing "Binding" to it. For this example it would be `MyItemBinding` for the layout file `my_item.xml`. That Binding class would also have a setter method to set the object defined as data in the layout file(`ItemModel` for this example).

Now that we have all the pieces we can implement our adapter like this:

```
class MyAdapter extends RecyclerView.Adapter<BindingViewHolder<MyItemBinding>>{
    ArrayList<ItemModel> items = new ArrayList<>();

    public MyAdapter(ArrayList<ItemModel> items) {
        this.items = items;
    }

    @Override public BindingViewHolder<MyItemBinding> onCreateViewHolder(ViewGroup parent, int
viewType) {
        return new
BindingViewHolder<>(LayoutInflater.from(parent.getContext()).inflate(R.layout.my_item, parent,
false));
    }

    @Override public void onBindViewHolder(BindingViewHolder<ItemModel> holder, int position) {
        holder.getBinding().setItemModel(items.get(position));
        holder.getBinding().executePendingBindings();
    }

    @Override public int getItemCount() {
        return items.size();
    }
}
```

Section 16.4: Animate data change

`RecyclerView` will perform a relevant animation if any of the "notify" methods are used except for `notifyDataSetChanged`; this includes `notifyItemChanged`, `notifyItemInserted`, `notifyItemMoved`, `notifyItemRemoved`, etc.

The adapter should extend this class instead of `RecyclerView.Adapter`.

```
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;

import java.util.List;

public abstract class AnimatedRecyclerAdapter<T, VH extends RecyclerView.ViewHolder>
        extends RecyclerView.Adapter<VH> {
    protected List<T> models;
```

```
    protected AnimatedRecyclerAdapter(@NonNull List<T> models) {
        this.models = models;
    }

    //Set new models.
    public void setModels(@NonNull final List<T> models) {
        applyAndAnimateRemovals(models);
        applyAndAnimateAdditions(models);
        applyAndAnimateMovedItems(models);
    }

    //Remove an item at position and notify changes.
    private T removeItem(int position) {
        final T model = models.remove(position);
        notifyItemRemoved(position);
        return model;
    }

    //Add an item at position and notify changes.
    private void addItem(int position, T model) {
        models.add(position, model);
        notifyItemInserted(position);
    }

    //Move an item at fromPosition to toPosition and notify changes.
    private void moveItem(int fromPosition, int toPosition) {
        final T model = models.remove(fromPosition);
        models.add(toPosition, model);
        notifyItemMoved(fromPosition, toPosition);
    }

    //Remove items that no longer exist in the new models.
    private void applyAndAnimateRemovals(@NonNull final List<T> newTs) {
        for (int i = models.size() - 1; i >= 0; i--) {
            final T model = models.get(i);
            if (!newTs.contains(model)) {
                removeItem(i);
            }
        }
    }

    //Add items that do not exist in the old models.
    private void applyAndAnimateAdditions(@NonNull final List<T> newTs) {
        for (int i = 0, count = newTs.size(); i < count; i++) {
            final T model = newTs.get(i);
            if (!models.contains(model)) {
                addItem(i, model);
            }
        }
    }

    //Move items that have changed their position.
    private void applyAndAnimateMovedItems(@NonNull final List<T> newTs) {
        for (int toPosition = newTs.size() - 1; toPosition >= 0; toPosition--) {
            final T model = newTs.get(toPosition);
            final int fromPosition = models.indexOf(model);
            if (fromPosition >= 0 && fromPosition != toPosition) {
                moveItem(fromPosition, toPosition);
            }
        }
    }
}
```

> You should **NOT** use the same `List` for `setModels` and `List` in the adapter.

You declare `models` as global variables. `DataModel` is a dummy class only.

```
private List<DataModel> models;
private YourAdapter adapter;
```

Initialize `models` before pass it to adapter. `YourAdapter` is the implementation of `AnimatedRecyclerAdapter`.

```
models = new ArrayList<>();
//Add models
models.add(new DataModel());
//Do NOT pass the models directly. Otherwise, when you modify global models,
//you will also modify models in adapter.
//adapter = new YourAdapter(models); <- This is wrong.
adapter = new YourAdapter(new ArrayList(models));
```

Call this after you have updated your global `models`.

```
adapter.setModels(new ArrayList(models));
```

> If you do not override `equals`, all the comparison is compared by reference.

Example using SortedList

Android introduced the `SortedList` class soon after `RecyclerView` was introduced. This class handles all 'notify' method calls to the `RecyclerView.Adapter` to ensure proper animation, and even allows batching multiple changes, so the animations don't jitter.

```
import android.support.v7.util.SortedList;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.util.SortedListAdapterCallback;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.List;

public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {

    private SortedList<DataModel> mSortedList;

    class ViewHolder extends RecyclerView.ViewHolder {

        TextView text;
        CheckBox checkBox;

        ViewHolder(View itemView){
            super(itemView);

            //Initiate your code here...

        }

        void setDataModel(DataModel model) {
            //Update your UI with the data model passed here...
```

```
            text.setText(modle.getText());
            checkBox.setChecked(model.isChecked());
        }
    }

    public MyAdapter() {
        mSortedList = new SortedList<>(DataModel.class, new
SortedListAdapterCallback<DataModel>(this) {
            @Override
            public int compare(DataModel o1, DataModel o2) {
                //This gets called to find the ordering between objects in the array.
                if (o1.someValue() < o2.someValue()) {
                    return -1;
                } else if (o1.someValue() > o2.someValue()) {
                    return 1;
                } else {
                    return 0;
                }
            }

            @Override
            public boolean areContentsTheSame(DataModel oldItem, DataModel newItem) {
                //This is to see of the content of this object has changed. These items are only
considered equal if areItemsTheSame() returned true.

                //If this returns false, onBindViewHolder() is called with the holder containing the
item, and the item's position.
                return oldItem.getText().equals(newItem.getText()) && oldItem.isChecked() ==
newItem.isChecked();
            }

            @Override
            public boolean areItemsTheSame(DataModel item1, DataModel item2) {
                //Checks to see if these two items are the same. If not, it is added to the list,
otherwise, check if content has changed.
                return item1.equals(item2);
            }
        });
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = //Initiate your item view here.
        return new ViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        //Just update the holder with the object in the sorted list from the given position
        DataModel model = mSortedList.get(position);
        if (model != null) {
            holder.setDataModel(model);
        }
    }

    @Override
    public int getItemCount() {
        return mSortedList.size();
    }

    public void resetList(List<DataModel> models) {
        //If you are performing multiple changes, use the batching methods to ensure proper
```

```
animation.
        mSortedList.beginBatchedUpdates();
        mSortedList.clear();
        mSortedList.addAll(models);
        mSortedList.endBatchedUpdates();
    }

    //The following methods each modify the data set and automatically handles calling the
appropriate 'notify' method on the adapter.
    public void addModel(DataModel model) {
        mSortedList.add(model);
    }

    public void addModels(List<DataModel> models) {
        mSortedList.addAll(models);
    }

    public void clear() {
        mSortedList.clear();
    }

    public void removeModel(DataModel model) {
        mSortedList.remove(model);
    }

    public void removeModelAt(int i) {
        mSortedList.removeItemAt(i);
    }
}
```

Section 16.5: Popup menu with recyclerView

put this code inside your ViewHolder

note: In this code I am using `btnExpand` click-event, for whole `recyclerview` click event you can set listener to itemView object.

```
public class MyViewHolder extends RecyclerView.ViewHolder{
        CardView cv;
        TextView recordName, visibleFile, date, time;
        Button btnIn, btnExpand;

        public MyViewHolder(final View itemView) {
            super(itemView);

            cv = (CardView)itemView.findViewById(R.id.cardview);
            recordName = (TextView)itemView.findViewById(R.id.tv_record);
            visibleFile = (TextView)itemView.findViewById(R.id.visible_file);
            date = (TextView)itemView.findViewById(R.id.date);
            time = (TextView)itemView.findViewById(R.id.time);
            btnIn = (Button)itemView.findViewById(R.id.btn_in_out);

            btnExpand = (Button) itemView.findViewById(R.id.btn_expand);

            btnExpand.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    PopupMenu popup = new PopupMenu(btnExpand.getContext(), itemView);

                    popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                        @Override
```

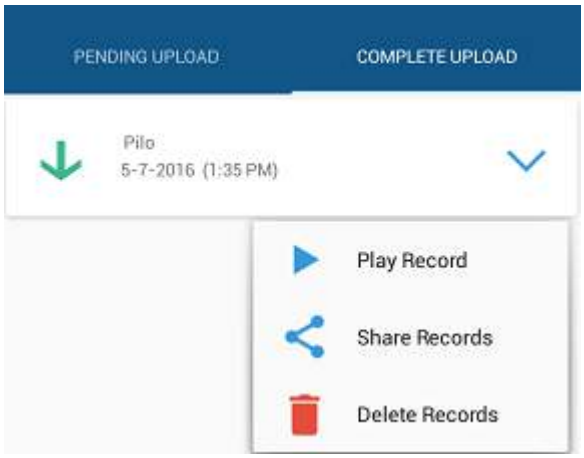
```
                        public boolean onMenuItemClick(MenuItem item) {
                            switch (item.getItemId()) {
                                case R.id.action_delete:
                                    moveFile(recordName.getText().toString(),
getAdapterPosition());
                                    return true;
                                case R.id.action_play:
                                    String valueOfPath = recordName.getText().toString();
                                    Intent intent = new Intent();
                                    intent.setAction(android.content.Intent.ACTION_VIEW);
                                    File file = new File(valueOfPath);
                                    intent.setDataAndType(Uri.fromFile(file), "audio/*");
                                    context.startActivity(intent);
                                    return true;
                                case R.id.action_share:
                                    String valueOfPath = recordName.getText().toString();
                                    File filee = new File(valueOfPath);
                                    try {
                                        Intent sendIntent = new Intent();
                                        sendIntent.setAction(Intent.ACTION_SEND);
                                        sendIntent.setType("audio/*");
                                        sendIntent.putExtra(Intent.EXTRA_STREAM,
Uri.fromFile(filee));
                                        context.startActivity(sendIntent);
                                    } catch (NoSuchMethodError | IllegalArgumentException |
NullPointerException e) {
                                        e.printStackTrace();
                                    } catch (Exception e) {
                                        e.printStackTrace();
                                    }
                                    return true;
                                default:
                                    return false;
                            }
                        }
                    });
                    // here you can inflate your menu
                    popup.inflate(R.menu.my_menu_item);
                    popup.setGravity(Gravity.RIGHT);

                    // if you want icon with menu items then write this try-catch block.
                    try {
                        Field mFieldPopup=popup.getClass().getDeclaredField("mPopup");
                        mFieldPopup.setAccessible(true);
                        MenuPopupHelper mPopup = (MenuPopupHelper) mFieldPopup.get(popup);
                        mPopup.setForceShowIcon(true);
                    } catch (Exception e) {

                    }
                    popup.show();
                }
            });

        }
    }
```

alternative way to show icons in menu

```
try {
    Field[] fields = popup.getClass().getDeclaredFields();
    for (Field field : fields) {
```

```
        if ("mPopup".equals(field.getName())) {
            field.setAccessible(true);
            Object menuPopupHelper = field.get(popup);
            Class<?> classPopupHelper = Class.forName(menuPopupHelper
                .getClass().getName());
            Method setForceIcons = classPopupHelper.getMethod(
                "setForceShowIcon", boolean.class);
            setForceIcons.invoke(menuPopupHelper, true);
            break;
        }
    }
} catch (Exception e) {

}
```

Here is the output:



Section 16.6: Using several ViewHolders with ItemViewType

Sometimes a RecyclerView will need to use several types of Views to be displayed in the list shown in the UI, and each View needs a different layout xml to be inflated.

For this issue, you may use different ViewHolders in single Adapter, by using a special method in RecyclerView - `getItemViewType(int position)`.

Below is example of using two ViewHolders:

1. A ViewHolder for displaying list entries
2. A ViewHolder for displaying multiple header views

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = LayoutInflater.from(context).inflate(viewType, parent, false);
    return ViewHolder.create(itemView, viewType);
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
    final Item model = this.items.get(position);
    ((ViewHolder) holder).bind(model);
}

@Override
public int getItemViewType(int position) {
```

```
        return inSearchState ? R.layout.item_header : R.layout.item_entry;
    }

    abstract class ViewHolder {
        abstract void bind(Item model);

        public static ViewHolder create(View v, int viewType) {
            return viewType == R.layout.item_header ? new HeaderViewHolder(v) :new
    EntryViewHolder(v);
        }
    }

    static class EntryViewHolder extends ViewHolder {
        private View v;

        public EntryViewHolder(View v) {
            this.v = v;
        }

        @Override public void bind(Item model) {
            // Bind item data to entry view.
        }
    }

    static class HeaderViewHolder extends ViewHolder {
        private View v;

        public HeaderViewHolder(View v) {
            this.v = v;
        }

        @Override public void bind(Item model) {
            // Bind item data to header view.
        }
    }
```

Section 16.7: Filter items inside RecyclerView with a SearchView

add `filter` method in `RecyclerView.Adapter`:

```
public void filter(String text) {
        if(text.isEmpty()){
            items.clear();
            items.addAll(itemsCopy);
        } else{
            ArrayList<PhoneBookItem> result = new ArrayList<>();
            text = text.toLowerCase();
            for(PhoneBookItem item: itemsCopy){
                //match by name or phone
                if(item.name.toLowerCase().contains(text) ||
item.phone.toLowerCase().contains(text)){
                    result.add(item);
                }
            }
            items.clear();
            items.addAll(result);
        }
        notifyDataSetChanged();
```

```
    }
```

`itemsCopy` is initialized in adapter's constructor like `itemsCopy.addAll(items)`.

If you do so, just call `filter` from `OnQueryTextListener` from `SearchView`:

```
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        adapter.filter(query);
        return true;
    }

    @Override
    public boolean onQueryTextChange(String newText) {
        adapter.filter(newText);
        return true;
    }
});
```

Section 16.8: Drag&Drop and Swipe with RecyclerView

You can implement the swipe-to-dismiss and drag-and-drop features with the RecyclerView without using 3rd party libraries.
Just use the `ItemTouchHelper` class included in the RecyclerView support library.

Instantiate the `ItemTouchHelper` with the `SimpleCallback` callback and depending on which functionality you support, you should override `onMove(RecyclerView, ViewHolder, ViewHolder)` and / or `onSwiped(ViewHolder, int)`and and finally attach to your `RecyclerView`.

```
ItemTouchHelper.SimpleCallback simpleItemTouchCallback = new ItemTouchHelper.SimpleCallback(0,
ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {

    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder, int swipeDir) {
        // remove item from adapter
    }

    @Override
    public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
RecyclerView.ViewHolder target) {
            final int fromPos = viewHolder.getAdapterPosition();
            final int toPos = target.getAdapterPosition();
            // move item in `fromPos` to `toPos` in adapter.
            return true;// true if moved, false otherwise
        }

};

ItemTouchHelper itemTouchHelper = new ItemTouchHelper(simpleItemTouchCallback);
itemTouchHelper.attachToRecyclerView(recyclerView);
```

It's worth mentioning that `SimpleCallback` constructor applies the same swiping strategy to all items in the `RecyclerView`. It's possible in any case to update the default swiping direction for specific items by simply overriding method `getSwipeDirs(RecyclerView, ViewHolder)`.

Let's suppose for example that our `RecyclerView` includes a `HeaderViewHolder` and that we obviously don't want to apply swiping to it. It will be enough to override `getSwipeDirs` as follows:

```
@Override
public int getSwipeDirs(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder) {
    if (viewHolder instanceof HeaderViewHolder) {
        // no swipe for header
        return 0;
    }
    // default swipe for all other items
    return super.getSwipeDirs(recyclerView, viewHolder);
}
```

Section 16.9: Show default view till items load or when data is not available

Screenshot

Adapter Class

```
private class MyAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

final int EMPTY_VIEW = 77777;
List<CustomData> datalist = new ArrayList<>();

MyAdapter() {
    super();
}

@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

    LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());
```

```
    if (viewType == EMPTY_VIEW) {
        return new EmptyView(layoutInflater.inflate(R.layout.nothing_yet, parent, false));
    } else {
        return new ItemView(layoutInflater.inflate(R.layout.my_item, parent, false));
    }
}

@SuppressLint("SetTextI18n")
@Override
public void onBindViewHolder(final RecyclerView.ViewHolder holder, int position) {
    if (getItemViewType(position) == EMPTY_VIEW) {
        EmptyView emptyView = (EmptyView) holder;
        emptyView.primaryText.setText("No data yet");
        emptyView.secondaryText.setText("You're doing good !");
        emptyView.primaryText.setCompoundDrawablesWithIntrinsicBounds(null, new
IconicsDrawable(getActivity()).icon(FontAwesome.Icon.faw_ticket).sizeDp(48).color(Color.DKGRAY),
null, null);

    } else {
        ItemView itemView = (ItemView) holder;
        // Bind data to itemView
    }
}

@Override
public int getItemCount() {
    return datalist.size() > 0 ? datalist.size() : 1;
}

@Override
public int getItemViewType(int position) {
    if  datalist.size() == 0) {
        return EMPTY_VIEW;
    }
    return super.getItemViewType(position);
}

}
```

nothing_yet.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_gravity="center"
   android:orientation="vertical"
   android:paddingBottom="100dp"
   android:paddingTop="100dp">

    <TextView
       android:id="@+id/nothingPrimary"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="center"
       android:drawableTint="@android:color/secondary_text_light"
       android:drawableTop="@drawable/ic_folder_open_black_24dp"
       android:enabled="false"
       android:fontFamily="sans-serif-light"
       android:text="No Item's Yet"
```

```
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="@android:color/secondary_text_light"
        android:textSize="40sp"
        tools:targetApi="m" />

    <TextView
        android:id="@+id/nothingSecondary"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:enabled="false"
        android:fontFamily="sans-serif-condensed"
        android:text="You're doing good !"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textColor="@android:color/tertiary_text_light" />
</LinearLayout>
```

I'm using FontAwesome with Iconics Library for the images. Add this to your app level build.gradle file.

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

Section 16.10: Add header/footer to a RecyclerView

This is a sample adapter code.

```
public class SampleAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

private static final int FOOTER_VIEW = 1;

// Define a view holder for Footer view

public class FooterViewHolder extends ViewHolder {
    public FooterViewHolder(View itemView) {
        super(itemView);
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Do whatever you want on clicking the item
            }
        });
    }
}

// Now define the viewholder for Normal list item
public class NormalViewHolder extends ViewHolder {
    public NormalViewHolder(View itemView) {
        super(itemView);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Do whatever you want on clicking the normal items
            }
        });
    }
}

// And now in onCreateViewHolder you have to pass the correct view
// while populating the list item.
```

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

    View v;

    if (viewType == FOOTER_VIEW) {
        v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_footer, parent,
false);

        FooterViewHolder vh = new FooterViewHolder(v);

        return vh;
    }

    v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_item_normal, parent, false);

    NormalViewHolder vh = new NormalViewHolder(v);

    return vh;
}

// Now bind the viewholders in onBindViewHolder
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {

    try {
        if (holder instanceof NormalViewHolder) {
            NormalViewHolder vh = (NormalViewHolder) holder;

            vh.bindView(position);
        } else if (holder instanceof FooterViewHolder) {
            FooterViewHolder vh = (FooterViewHolder) holder;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// Now the critical part. You have return the exact item count of your list
// I've only one footer. So I returned data.size() + 1
// If you've multiple headers and footers, you've to return total count
// like, headers.size() + data.size() + footers.size()

@Override
public int getItemCount() {
    if (data == null) {
        return 0;
    }

    if (data.size() == 0) {
        //Return 1 here to show nothing
        return 1;
    }

    // Add extra view to show the footer view
    return data.size() + 1;
}

// Now define getItemViewType of your own.

@Override
public int getItemViewType(int position) {
```

```
    if (position == data.size()) {
        // This is where we'll add footer.
        return FOOTER_VIEW;
    }

    return super.getItemViewType(position);
}

// So you're done with adding a footer and its action on onClick.
// Now set the default ViewHolder for NormalViewHolder

public class ViewHolder extends RecyclerView.ViewHolder {
    // Define elements of a row here
    public ViewHolder(View itemView) {
        super(itemView);
        // Find view by ID and initialize here
    }

    public void bindView(int position) {
        // bindView() method to implement actions
    }
}
}
```

Here's a good read about the implementation of `RecyclerView` with header and footer.

Alternate method:

While the above answer will work you can use this approach as well using a recycler view using a `NestedScrollView` .You can add a layout for header using the following approach:

```
<android.support.v4.widget.NestedScrollView
       android:layout_width="match_parent"
       android:layout_height="match_parent">

     <RelativeLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent">

        <include
           layout="@layout/drawer_view_header"
           android:id="@+id/navigation_header" />

        <android.support.v7.widget.RecyclerView
           android:layout_below="@id/navigation_header"
           android:id="@+id/followers_list"
           android:layout_width="match_parent"
           android:layout_height="wrap_content" />

    </RelativeLayout>
</android.support.v4.widget.NestedScrollView>
```

Or you may also use a `LinearLayout` with vertical alignment in your `NestedScrollView`.

Note: This will only work with `RecyclerView` above **23.2.0**

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

Section 16.11: Endless Scrolling in Recycleview

Here I have shared a code snippet for implementing endless scrolling in recycle view.

Step 1: First make a one abstract method in Recycleview adapter like below.

```
public abstract class ViewAllCategoryAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder>
{
    public abstract void load();
}
```

Step 2: Now override onBindViewHolder and `getItemCount()` method of ViewAllCategoryAdapter class and call `Load()` method like below.

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
}
```

Step 3: Now every backend logic is complete now it's time to execute this logic.It's simple you can override load method where you create object of your adapter.this method is automatically call while user reach at end of the listing.

```
  adapter = new ViewAllCategoryAdapter(CONTEXT, YOURLIST) {
        @Override
        public void load() {

            /* do your stuff here */
            /* This method is automatically call while user reach at end of your list. */
        }
    };
    recycleCategory.setAdapter(adapter);
```

Now `load()` method automatically call while user scroll at end of list.

Best Luck

Section 16.12: Add divider lines to RecyclerView items

Just add these lines to the initialization

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
DividerItemDecoration.VERTICAL));
```

Add an `adapter` and call `.notifyDataSetChanged();` as usual !
This is not an inbuilt feature of Recyclerview but added in the support libraries. So don't forget to include this in your app level build.gradle file

```
compile "com.android.support:appcompat-v7:25.3.1"
compile "com.android.support:recyclerview-v7:25.3.1"
```

> Multiple ItemDecorations can be added to a single RecyclerView.

Changing divider color :

It's pretty easy to set an color for a itemDecoration.

1. step is: creating a `divider.xml` file which is located on `drawable` folder

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
      android:shape="line">
    <size
       android:width="1px"
       android:height="1px"/>
    <solid android:color="@color/divider_color"/>
</shape>
```

2. step is: setting drawable

```
    // Get drawable object
    Drawable mDivider = ContextCompat.getDrawable(m_jContext, R.drawable.divider);
    // Create a DividerItemDecoration whose orientation is Horizontal
    DividerItemDecoration hItemDecoration = new DividerItemDecoration(m_jContext,
            DividerItemDecoration.HORIZONTAL);
    // Set the drawable on it
    hItemDecoration.setDrawable(mDivider);
```


```
    // Create a DividerItemDecoration whose orientation is vertical
    DividerItemDecoration vItemDecoration = new DividerItemDecoration(m_jContext,
            DividerItemDecoration.VERTICAL);
    // Set the drawable on it
    vItemDecoration.setDrawable(mDivider);
```

row 7

row 8

row 9

row 10

row 11

row 12

row 13

Chapter 17: RecyclerView Decorations

Parameter	Details
decoration	the item decoration to add to the `RecyclerView`
index	the index in the list of decorations for this `RecyclerView`. This is the order in which `getItemOffset` and `onDraw` are called. Later calls might overdraw previous ones.

Section 17.1: Add divider to RecyclerView

First of all you need to create a class which extends `RecyclerView.ItemDecoration` :

```
public class SimpleBlueDivider extends RecyclerView.ItemDecoration {
private Drawable mDivider;

public SimpleBlueDivider(Context context) {
    mDivider = context.getResources().getDrawable(R.drawable.divider_blue);
}

@Override
public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
    //divider padding give some padding whatever u want or disable
    int left =parent.getPaddingLeft()+80;
    int right = parent.getWidth() - parent.getPaddingRight()-30;

    int childCount = parent.getChildCount();
    for (int i = 0; i < childCount; i++) {
        View child = parent.getChildAt(i);

        RecyclerView.LayoutParams params = (RecyclerView.LayoutParams) child.getLayoutParams();

        int top = child.getBottom() + params.bottomMargin;
        int bottom = top + mDivider.getIntrinsicHeight();

        mDivider.setBounds(left, top, right, bottom);
        mDivider.draw(c);
    }
}

}
```

Add `divider_blue.xml` to your drawable folder :

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">
<size android:width="1dp" android:height="4dp" />
<solid android:color="#AA123456" />
</shape>
```

Then use it like :

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

The result will be like :

This image is just an example how dividers working , if you want to follow Material Design specs when adding dividers please take a look at this link : dividers and thanks @Brenden Kromhout by providing link .

Section 17.2: Drawing a Separator

This will draw a line at the bottom of every view but the last to act as a separator between items.

```
public class SeparatorDecoration extends RecyclerView.ItemDecoration {

    private final Paint mPaint;
    private final int mAlpha;

    public SeparatorDecoration(@ColorInt int color, float width) {
        mPaint = new Paint();
        mPaint.setColor(color);
        mPaint.setStrokeWidth(width);
        mAlpha = mPaint.getAlpha();
    }

    @Override
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent, RecyclerView.State
state) {
        final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
view.getLayoutParams();

        // we retrieve the position in the list
        final int position = params.getViewAdapterPosition();

        // add space for the separator to the bottom of every view but the last one
        if (position < state.getItemCount()) {
            outRect.set(0, 0, 0, (int) mPaint.getStrokeWidth()); // left, top, right, bottom
        } else {
            outRect.setEmpty(); // 0, 0, 0, 0
        }
    }

    @Override
    public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
        // a line will draw half its size to top and bottom,
        // hence the offset to place it correctly
        final int offset = (int) (mPaint.getStrokeWidth() / 2);

        // this will iterate over every visible view
        for (int i = 0; i < parent.getChildCount(); i++) {
            final View view = parent.getChildAt(i);
            final RecyclerView.LayoutParams params = (RecyclerView.LayoutParams)
view.getLayoutParams();

            // get the position
            final int position = params.getViewAdapterPosition();

            // and finally draw the separator
            if (position < state.getItemCount()) {
                // apply alpha to support animations
                mPaint.setAlpha((int) (view.getAlpha() * mAlpha));

                float positionY = view.getBottom() + offset + view.getTranslationY();
                // do the drawing
                c.drawLine(view.getLeft() + view.getTranslationX(),
                        positionY,
                        view.getRight() + view.getTranslationX(),
                        positionY,
                        mPaint);
            }
```

```
        }
    }
}
```

Section 17.3: How to add dividers using and DividerItemDecoration

The `DividerItemDecoration` is a `RecyclerView.ItemDecoration` that can be used as a divider between items.

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,
             mLayoutManager.getOrientation());
recyclerView.addItemDecoration(mDividerItemDecoration);
```

It supports both orientation using `DividerItemDecoration.VERTICAL` and `DividerItemDecoration.HORIZONTAL`.

Section 17.4: Per-item margins with ItemDecoration

You can use a `RecyclerView.ItemDecoration` to put extra margins around each item in a RecyclerView. This can in some cases clean up both your adapter implementation and your item view XML.

```
public class MyItemDecoration
    extends RecyclerView.ItemDecoration {

    private final int extraMargin;

    @Override
    public void getItemOffsets(Rect outRect, View view,
            RecyclerView parent, RecyclerView.State state) {

        int position = parent.getChildAdapterPosition(view);

        // It's easy to put extra margin on the last item...
        if (position + 1 == parent.getAdapter().getItemCount()) {
            outRect.bottom = extraMargin; // unit is px
        }

        // ...or you could give each item in the RecyclerView different
        // margins based on its position...
        if (position % 2 == 0) {
            outRect.right = extraMargin;
        } else {
            outRect.left = extraMargin;
        }

        // ...or based on some property of the item itself
        MyListItem item = parent.getAdapter().getItem(position);
        if (item.isFirstItemInSection()) {
            outRect.top = extraMargin;
        }
    }

    public MyItemDecoration(Context context) {
        extraMargin = context.getResources()
                .getDimensionPixelOffset(R.dimen.extra_margin);
    }
}
```

To enable the decoration, simply add it to your RecyclerView:

```
// in your onCreate()
RecyclerView rv = (RecyclerView) findItemById(R.id.myList);
rv.addItemDecoration(new MyItemDecoration(context));
```

Section 17.5: ItemOffsetDecoration for GridLayoutManager in RecycleView

Following example will help to give equal space to an item in GridLayout.

ItemOffsetDecoration.java

```
public class ItemOffsetDecoration extends RecyclerView.ItemDecoration {

    private int mItemOffset;

    private int spanCount = 2;

    public ItemOffsetDecoration(int itemOffset) {
        mItemOffset = itemOffset;
    }

    public ItemOffsetDecoration(@NonNull Context context, @DimenRes int itemOffsetId) {
        this(context.getResources().getDimensionPixelSize(itemOffsetId));
    }

    @Override
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent,
                               RecyclerView.State state) {
        super.getItemOffsets(outRect, view, parent, state);

        int position = parent.getChildLayoutPosition(view);

        GridLayoutManager manager = (GridLayoutManager) parent.getLayoutManager();

        if (position < manager.getSpanCount())
            outRect.top = mItemOffset;

        if (position % 2 != 0) {
            outRect.right = mItemOffset;
        }

        outRect.left = mItemOffset;
        outRect.bottom = mItemOffset;
    }
}
```

You can call ItemDecoration like below code.

```
recyclerView = (RecyclerView) view.findViewById(R.id.recycler_view);

GridLayoutManager lLayout = new GridLayoutManager(getActivity(), 2);

ItemOffsetDecoration itemDecoration = new ItemOffsetDecoration(mActivity, R.dimen.item_offset);
recyclerView.addItemDecoration(itemDecoration);

recyclerView.setLayoutManager(lLayout);
```

> and example item offset

```
<dimen name="item_offset">5dp</dimen>
```

Chapter 18: RecyclerView onClickListeners

Section 18.1: Kotlin and RxJava example

First example reimplemented in Kotlin and using RxJava for cleaner interaction.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.support.v7.widget.RecyclerView
import rx.subjects.PublishSubject

public class SampleAdapter(private val items: Array<String>) :
RecyclerView.Adapter<SampleAdapter.ViewHolder>() {

    // change to different subjects from rx.subjects to get different behavior
    // BehaviorSubject for example allows to receive last event on subscribe
    // PublishSubject sends events only after subscribing on the other hand which is desirable for
clicks
    public val itemClickStream: PublishSubject<View> = PublishSubject.create()

    override fun getItemCount(): Int {
        return items.size
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder? {
        val v = LayoutInflater.from(parent.getContext()).inflate(R.layout.text_row_item, parent,
false);
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.bind(items[position])
    }

    public inner class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        private val textView: TextView by lazy { view.findViewById(R.id.textView) as TextView }

        init {
            view.setOnClickListener { v -> itemClickStream.onNext(v) }
        }

        fun bind(text: String) {
            textView.text = text
        }
    }
}
```

Usage is quite simple then. It's possible to subscribe on separate thread using RxJava facilities.

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // do something
    }
}
```

Section 18.2: RecyclerView Click listener

```
public  class RecyclerTouchListener implements RecyclerView.OnItemTouchListener {

    private GestureDetector gestureDetector;
    private RecyclerTouchListener.ClickListener clickListener;

    public RecyclerTouchListener(Context context, final RecyclerView recyclerView, final
RecyclerTouchListener.ClickListener clickListener) {
        this.clickListener = clickListener;

        gestureDetector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }
            @Override
            public void onLongPress(MotionEvent e) {
                View child = recyclerView.findChildViewUnder(e.getX(), e.getY());
                if (child != null && clickListener != null) {
                    clickListener.onLongClick(child, recyclerView.getChildPosition(child));
                }
            }
        });
    }


    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {
        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child, rv.getChildPosition(child));
        }
        return false;
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {

    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {

    }

    public interface ClickListener {
        void onLongClick(View child, int childPosition);

        void onClick(View child, int childPosition);
    }
}
```

In MainActivity

```
RecyclerView recyclerView =(RecyclerView) findViewById(R.id.recyclerview);
 recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getActivity(),recyclerView, new
RecyclerTouchListener.ClickListener() {
      @Override
      public void onLongClick(View child, int childPosition) {
```

```
      }

      @Override
      public void onClick(View child, int childPosition) {


     }
    }));
```

Section 18.3: Another way to implement Item Click Listener

Another way to implement item click listener is to use interface with several methods, the number of which is equal to the number of clickable views, and use overrided click listeners as you can see below. This method is more flexible, because you can set click listeners to different views and quite easy control the click logic separately for each.

```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.CustomHolder> {

    private ArrayList<Object> mObjects;
    private ClickInterface mClickInterface;

    public interface ClickInterface {
        void clickEventOne(Object obj);
        void clickEventTwo(Object obj1, Object obj2);
    }

    public void setClickInterface(ClickInterface clickInterface) {
        mClickInterface = clickInterface;
    }

    public CustomAdapter(){
        mList = new ArrayList<>();
    }

    public void addItems(ArrayList<Object> objects) {
        mObjects.clear();
        mObjects.addAll(objects);
        notifyDataSetChanged();
    }

    @Override
    public CustomHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext())
                               .inflate(R.layout.list_item, parent, false);
        return new CustomHolder(v);
    }

    @Override
    public void onBindViewHolder(CustomHolder holder, int position) {
        //make all even positions not clickable
        holder.firstClickListener.setClickable(position%2==0);
        holder.firstClickListener.setPosition(position);
        holder.secondClickListener.setPosition(position);
    }


    private class FirstClickListener implements View.OnClickListener {
        private int mPosition;
        private boolean mClickable;
```

```
        void setPosition(int position) {
            mPosition = position;
        }

        void setClickable(boolean clickable) {
            mPosition = position;
        }

        @Override
        public void onClick(View v) {
            if(mClickable) {
                mClickInterface.clickEventOne(mObjects.get(mPosition));
            }
        }
    }

    private class SecondClickListener implements View.OnClickListener {
        private int mPosition;

        void setPosition(int position) {
            mPosition = position;
        }

        @Override
        public void onClick(View v) {
            mClickInterface.clickEventTwo(mObjects.get(mPosition), v);
        }
    }

    @Override
    public int getItemCount() {
        return mObjects.size();
    }

    protected class CustomHolder extends RecyclerView.ViewHolder {
        FirstClickListener firstClickListener;
        SecondClickListener secondClickListener;
        View v1, v2;

        public DialogHolder(View itemView) {
            super(itemView);
            v1 = itemView.findViewById(R.id.v1);
            v2 = itemView.findViewById(R.id.v2);
            firstClickListener = new FirstClickListener();
            secondClickListener = new SecondClickListener();

            v1.setOnClickListener(firstClickListener);
            v2.setOnClickListener(secondClickListener);
        }
    }
}
```

And when you have an instance of adapter, you can set your click listener which listens to clicking on each of the views:

```
customAdapter.setClickInterface(new CustomAdapter.ClickInterface {
    @Override
    public void clickEventOne(Object obj) {
        // Your implementation here
    }
    @Override
```

```
    public void clickEventTwo(Object obj1, Object obj2) {
        // Your implementation here
    }
});
```

Section 18.4: New Example

```
public class SampleAdapter extends RecyclerView.Adapter<SampleAdapter.ViewHolder> {

    private String[] mDataSet;
    private OnRVItemClickListener mListener;

    /**
     * Provide a reference to the type of views that you are using (custom ViewHolder)
     */
    public static class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView textView;

        public ViewHolder(View v) {
            super(v);
            // Define click listener for the ViewHolder's View.
            v.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) { // handle click events here
                    Log.d(TAG, "Element " + getPosition() + " clicked.");
                    mListener.onRVItemClicked(getPosition(),v); //set callback
                }
            });
            textView = (TextView) v.findViewById(R.id.textView);
        }

        public TextView getTextView() {
            return textView;
        }
    }

    /**
     * Initialize the dataset of the Adapter.
     *
     * @param dataSet String[] containing the data to populate views to be used by RecyclerView.
     */
    public SampleAdapter(String[] dataSet) {
        mDataSet = dataSet;
    }

    // Create new views (invoked by the layout manager)
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        // Create a new view.
        View v = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.text_row_item, viewGroup, false);

        return new ViewHolder(v);
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(ViewHolder viewHolder, final int position) {
        // Get element from your dataset at this position and replace the contents of the view
        // with that element
        viewHolder.getTextView().setText(mDataSet[position]);
```

```
    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return mDataSet.length;
    }

    public void setOnRVClickListener(OnRVItemClickListener) {
        mListener = OnRVItemClickListener;
    }

    public interface OnRVItemClickListener {
       void onRVItemClicked(int position, View v);
    }
}
```

Section 18.5: Easy OnLongClick and OnClick Example

First of all, implement your view holder:

```
implements View.OnClickListener, View.OnLongClickListener
```

Then, register the listeners as follows:

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

Next, override the listeners as follows:

```
@Override
public void onClick(View v) {
    oncclicklistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    oncclicklistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}
```

And finally, add the following code:

```
public void setOnItemClickListener(onClickListner oncclicklistner) {
    SampleAdapter.oncclicklistner = oncclicklistner;
}

public void setHeader(View v) {
    this.headerView = v;
}

public interface onClickListner {
    void onItemClick(int position, View v);
    void onItemLongClick(int position, View v);
}
```

Adaptor demo

```
package adaptor;
```

```
import android.annotation.SuppressLint;
import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.wings.example.recycleview.MainActivity;
import com.wings.example.recycleview.R;

import java.util.ArrayList;

public class SampleAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
    Context context;
    private ArrayList<String> arrayList;
    private static onClickListner onclicklistner;
    private static final int VIEW_HEADER = 0;
    private static final int VIEW_NORMAL = 1;
    private View headerView;

    public SampleAdapter(Context context) {
        this.context = context;
        arrayList = MainActivity.arrayList;
    }

    public class HeaderViewHolder extends RecyclerView.ViewHolder {
        public HeaderViewHolder(View itemView) {
            super(itemView);
        }
    }

    public class ItemViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener,
View.OnLongClickListener {
        TextView txt_pos;
        SampleAdapter sampleAdapter;

        public ItemViewHolder(View itemView, SampleAdapter sampleAdapter) {
            super(itemView);

            itemView.setOnClickListener(this);
            itemView.setOnLongClickListener(this);

            txt_pos = (TextView) itemView.findViewById(R.id.txt_pos);
            this.sampleAdapter = sampleAdapter;

            itemView.setOnClickListener(this);
        }

        @Override
        public void onClick(View v) {
            oncliclistner.onItemClick(getAdapterPosition(), v);
        }

        @Override
        public boolean onLongClick(View v) {
            oncliclistner.onItemLongClick(getAdapterPosition(), v);
            return true;
        }
    }

    public void setOnItemClickListener(onClickListner oncliclistner) {
```

```
        SampleAdapter.onclicklistner = onclicklistner;
    }

    public void setHeader(View v) {
        this.headerView = v;
    }

    public interface onClickListner {
        void onItemClick(int position, View v);
        void onItemLongClick(int position, View v);
    }

    @Override
    public int getItemCount() {
        return arrayList.size()+1;
    }

    @Override
    public int getItemViewType(int position) {
        return position == 0 ? VIEW_HEADER : VIEW_NORMAL;
    }

    @SuppressLint("InflateParams")
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        if (viewType == VIEW_HEADER) {
            return new HeaderViewHolder(headerView);
        } else {
            View view =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.custom_recycler_row_sample_item,
viewGroup, false);
            return new ItemViewHolder(view, this);
        }
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
        if (viewHolder.getItemViewType() == VIEW_HEADER) {
            return;
        } else {
            ItemViewHolder itemViewHolder = (ItemViewHolder) viewHolder;
            itemViewHolder.txt_pos.setText(arrayList.get(position-1));
        }
    }
}
```

The example code above can be called by the following code:

```
sampleAdapter.setOnItemClickListener(new SampleAdapter.onClickListner() {
    @Override
    public void onItemClick(int position, View v) {
        position = position+1;//As we are adding header
        Log.e(TAG + "ON ITEM CLICK", position + "");
        Snackbar.make(v, "On item click "+position, Snackbar.LENGTH_LONG).show();
    }

    @Override
    public void onItemLongClick(int position, View v) {
        position = position+1;//As we are adding header
        Log.e(TAG + "ON ITEM LONG CLICK", position + "");
        Snackbar.make(v, "On item longclick  "+position, Snackbar.LENGTH_LONG).show();
```

```
    }
});
```

Section 18.6: Item Click Listeners

To implement an item click listener and/or an item long click listener, you can create an interface in your adapter:

```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {

    public interface OnItemClickListener {

        void onItemSeleted(int position, View view, CustomObject object);
    }

    public interface OnItemLongClickListener {

        boolean onItemSelected(int position, View view, CustomObject object);
    }

    public final class ViewHolder extends RecyclerView.ViewHolder {

        public ViewHolder(View itemView) {
            super(itemView);
            final int position = getAdapterPosition();

            itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    if(mOnItemClickListener != null) {
                        mOnItemClickListener.onItemSeleted(position, view, mDataSet.get(position));
                    }
                }
            });

            itemView.setOnLongClickListener(new View.OnLongClickListener() {
                @Override
                public boolean onLongClick(View view) {
                    if(mOnItemLongClickListener != null) {
                        return mOnItemLongClickListener.onItemSelected(position, view,
mDataSet.get(position));
                    }
                }
            });

        }
    }

    private List<CustomObject> mDataSet;

    private OnItemClickListener mOnItemClickListener;
    private OnItemLongClickListener mOnItemLongClickListener;

    public CustomAdapter(List<CustomObject> dataSet) {
        mDataSet = dataSet;
    }

    @Override
    public CustomAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.view_item_custom, parent, false);
        return new ViewHolder(view);
```

```
    }

    @Override
    public void onBindViewHolder(CustomAdapter.ViewHolder holder, int position) {
        // Bind views
    }

    @Override
    public int getItemCount() {
        return mDataSet.size();
    }

    public void setOnItemClickListener(OnItemClickListener listener) {
        mOnItemClickListener = listener;
    }

    public void setOnItemLongClickListener(OnItemLongClickListener listener) {
        mOnItemLongClickListener = listener;
    }

}
```

Then you can set your click listeners after you create an instance of the adapter:

```
customAdapter.setOnItemClickListener(new CustomAdapter.OnItemClickListener {
    @Override
    public void onItemSelected(int position, View view, CustomObject object) {
        // Your implementation here
    }
});

customAdapter.setOnItemLongClickListener(new CustomAdapter.OnItemLongClickListener {
    @Override
    public boolean onItemSelected(int position, View view, CustomObject object) {
        // Your implementation here
        return true;
    }
});
```

Chapter 19: RecyclerView and LayoutManagers

Section 19.1: Adding header view to recyclerview with gridlayout manager

To add a header to a recyclerview with a gridlayout, first the adapter needs to be told that the header view is the first position - rather than the standard cell used for the content. Next, the layout manager must be told that the first position should have a span equal to the *span count of the entire list. *

Take a regular RecyclerView.Adapter class and configure it as follows:

```
public class HeaderAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private static final int ITEM_VIEW_TYPE_HEADER = 0;
    private static final int ITEM_VIEW_TYPE_ITEM = 1;

    private List<YourModel> mModelList;

    public HeaderAdapter (List<YourModel> modelList) {
        mModelList = modelList;
    }

    public boolean isHeader(int position) {
        return position == 0;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());

        if (viewType == ITEM_VIEW_TYPE_HEADER) {
            View headerView = inflater.inflate(R.layout.header, parent, false);
            return new HeaderHolder(headerView);
        }

        View cellView = inflater.inflate(R.layout.gridcell, parent, false);
        return new ModelHolder(cellView);
    }

    @Override
    public int getItemViewType(int position) {
        return isHeader(position) ? ITEM_VIEW_TYPE_HEADER : ITEM_VIEW_TYPE_ITEM;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder h, int position) {
        if (isHeader(position)) {
            return;
        }

        final YourModel model = mModelList.get(position -1 ); // Subtract 1 for header

        ModelHolder holder = (ModelHolder) h;
        // populate your holder with data from your model as usual
    }

    @Override
```

```
    public int getItemCount() {
        return _categories.size() + 1; // add one for the header
    }
}
```

Then in the activity/fragment:

```
final HeaderAdapter adapter = new HeaderAdapter (mModelList);
final GridLayoutManager manager = new GridLayoutManager();
manager.setSpanSizeLookup(new GridLayoutManager.SpanSizeLookup() {
    @Override
    public int getSpanSize(int position) {
        return adapter.isHeader(position) ? manager.getSpanCount() : 1;
    }
});
mRecyclerView.setLayoutManager(manager);
mRecyclerView.setAdapter(adapter);
```

The same approach can be used add a footer in addition to or instead of a header.

Source: Chiu-Ki Chan's Square Island blog

Section 19.2: GridLayoutManager with dynamic span count

When creating a recyclerview with a gridlayout layout manager you have to specify the span count in the constructor. Span count refers to the number of columns. This is fairly clunky and doesn't take into account larger screen sizes or screen orientation. One approach is to create multiple layouts for the various screen sizes. Another more dynamic approach can be seen below.

First we create a custom RecyclerView class as follows:

```
public class AutofitRecyclerView extends RecyclerView {
    private GridLayoutManager manager;
    private int columnWidth = -1;

    public AutofitRecyclerView(Context context) {
        super(context);
        init(context, null);
    }

    public AutofitRecyclerView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(context, attrs);
    }

    public AutofitRecyclerView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(context, attrs);
    }

    private void init(Context context, AttributeSet attrs) {
        if (attrs != null) {
            int[] attrsArray = {
                    android.R.attr.columnWidth
            };
            TypedArray array = context.obtainStyledAttributes(attrs, attrsArray);
            columnWidth = array.getDimensionPixelSize(0, -1);
            array.recycle();
        }
```

```
        manager = new GridLayoutManager(getContext(), 1);
        setLayoutManager(manager);
    }

    @Override
    protected void onMeasure(int widthSpec, int heightSpec) {
        super.onMeasure(widthSpec, heightSpec);
        if (columnWidth > 0) {
            int spanCount = Math.max(1, getMeasuredWidth() / columnWidth);
            manager.setSpanCount(spanCount);
        }
    }
}
```

This class determines how many columns can fit into the recyclerview. To use it you will need to put it into your layout.xml as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<com.path.to.your.class.autofitRecyclerView.AutofitRecyclerView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/auto_fit_recycler_view"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:columnWidth="200dp"
   android:clipToPadding="false"
   />
```

Notice that we use the columnWidth attribute. The recyclerview will need it to determine how many columns will fit into the available space.

In your activity/fragment you just get a reference to the recylerview and set an adapter to it (and any item decorations or animations that you want to add). **DO NOT SET A LAYOUT MANAGER**

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());
```

(where MyAdapter is your adapter class)

You now have a recyclerview that will adjust the spancount (ie columns) to fit the screen size. As a final addition you might want to center the columns in the recyclerview (by default they are aligned to layout_start). You can do that by modifying the AutofitRecyclerView class a little. Start by creating an inner class in the recyclerview. This will be a class that extends from GridLayoutManager. It will add enough padding to the left and right in order to center the rows:

```
public class AutofitRecyclerView extends RecyclerView {

    // etc see above

    private class CenteredGridLayoutManager extends GridLayoutManager {

        public CenteredGridLayoutManager(Context context, AttributeSet attrs, int defStyleAttr, int
defStyleRes) {
            super(context, attrs, defStyleAttr, defStyleRes);
        }

        public CenteredGridLayoutManager(Context context, int spanCount) {
            super(context, spanCount);
        }
```

```
        public CenteredGridLayoutManager(Context context, int spanCount, int orientation, boolean
reverseLayout) {
            super(context, spanCount, orientation, reverseLayout);
        }

        @Override
        public int getPaddingLeft() {
            final int totalItemWidth = columnWidth * getSpanCount();
            if (totalItemWidth >= AutofitRecyclerView.this.getMeasuredWidth()) {
                return super.getPaddingLeft(); // do nothing
            } else {
                return Math.round((AutofitRecyclerView.this.getMeasuredWidth() / (1f +
getSpanCount())) - (totalItemWidth / (1f + getSpanCount())));
            }
        }

        @Override
        public int getPaddingRight() {
            return getPaddingLeft();
        }
    }
}
```

Then when you set the LayoutManager in the AutofitRecyclerView use the CenteredGridLayoutManager as follows:

```
private void init(Context context, AttributeSet attrs) {
    if (attrs != null) {
        int[] attrsArray = {
                android.R.attr.columnWidth
        };
        TypedArray array = context.obtainStyledAttributes(attrs, attrsArray);
        columnWidth = array.getDimensionPixelSize(0, -1);
        array.recycle();
    }

    manager = new CenteredGridLayoutManager(getContext(), 1);
    setLayoutManager(manager);
}
```

And that's it! You have a dynamic spancount, center aligned gridlayoutmanager based recyclerview.

Sources:

- Chiu-Ki Chan's Square Island blog
- StackOverflow

Section 19.3: Simple list with LinearLayoutManager

This example adds a list of places with image and name by using an `ArrayList` of custom `Place` objects as dataset.

Activity layout

The layout of the activity / fragment or where the RecyclerView is used only has to contain the RecyclerView. There is no ScrollView or a specific layout needed.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
```

```
    <android.support.v7.widget.RecyclerView
       android:id="@+id/my_recycler_view"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />

</RelativeLayout>
```

Define the data model

You could use any class or primitive data type as a model, like `int`, `String`, `float[]` or `CustomObject`. The RecyclerView will refer to a `List` of this objects / primitives.

When a list item refers to different data types like text, numbers, images (as in this example with places), it is often a good idea to use a custom object.

```
public class Place {
    // these fields will be shown in a list item
    private Bitmap image;
    private String name;

    // typical constructor
    public Place(Bitmap image, String name) {
        this.image = image;
        this.name = name;
    }

    // getters
    public Bitmap getImage() {
        return image;
    }
    public String getName() {
        return name;
    }
}
```

List item layout

You have to specify a xml layout file that will be used for each list item. In this example, an `ImageView` is used for the image and a `TextView` for the name. The `LinearLayout` positions the `ImageView` at the left and the `TextView` right to the image.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:gravity="center_vertical"
   android:orientation="horizontal"
   android:padding="8dp">

    <ImageView
       android:id="@+id/image"
       android:layout_width="36dp"
       android:layout_height="36dp"
       android:layout_marginEnd="8dp"
       android:layout_marginRight="8dp" />

    <TextView
       android:id="@+id/name"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

Create a RecyclerView adapter and ViewHolder

Next, you have to inherit the `RecyclerView.Adapter` and the `RecyclerView.ViewHolder`. A usual class structure would be:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    public class ViewHolder extends RecyclerView.ViewHolder {
        // ...
    }
}
```

First, we implement the `ViewHolder`. It only inherits the default constructor and saves the needed views into some fields:

```
public class ViewHolder extends RecyclerView.ViewHolder {
    private ImageView imageView;
    private TextView nameView;

    public ViewHolder(View itemView) {
        super(itemView);

        imageView = (ImageView) itemView.findViewById(R.id.image);
        nameView = (TextView) itemView.findViewById(R.id.name);
    }
}
```

The adapter's constructor sets the used dataset:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    private List<Place> mPlaces;

    public PlaceListAdapter(List<Place> contacts) {
        mPlaces = contacts;
    }

    // ...
}
```

To use our custom list item layout, we override the method `onCreateViewHolder(...)`. In this example, the layout file is called `place_list_item.xml`.

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(
                R.layout.place_list_item,
                parent,
                false
        );
        return new ViewHolder(view);
    }

    // ...
```

```
}
```

In the `onBindViewHolder(...)`, we actually set the views' contents. We get the used model by finding it in the `List` at the given position and then set image and name on the `ViewHolder`'s views.

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public void onBindViewHolder(PlaceListAdapter.ViewHolder viewHolder, int position) {
        Place place = mPlaces.get(position);

        viewHolder.nameView.setText(place.getName());
        viewHolder.imageView.setImageBitmap(place.getImage());
    }

    // ...
}
```

We also need to implement `getItemCount()`, which simply return the `List`'s size.

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

(Generate random data)

For this example, we'll generate some random places.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...

    List<Place> places = randomPlaces(5);

    // ...
}

private List<Place> randomPlaces(int amount) {
    List<Place> places = new ArrayList<>();
    for (int i = 0; i < amount; i++) {
        places.add(new Place(
                BitmapFactory.decodeResource(getResources(), Math.random() > 0.5 ?
                        R.drawable.ic_account_grey600_36dp :
                        R.drawable.ic_android_grey600_36dp
                ),
                "Place #" + (int) (Math.random() * 1000)
        ));
    }
    return places;
}
```

Connect the RecyclerView with the PlaceListAdapter and the dataset

Connecting a `RecyclerView` with an adapter is very easy. You have to set the `LinearLayoutManager` as layout manager to achieve the list layout.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...

    RecyclerView recyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
    recyclerView.setAdapter(new PlaceListAdapter(places));
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
}
```

Done!

Section 19.4: StaggeredGridLayoutManager

1. Create your RecyclerView in your layout xml file:

```
<android.support.v7.widget.RecyclerView
           android:id="@+id/recycleView"
           android:layout_width="match_parent"
           android:layout_height="match_parent" />
```

2. Create your Model class for holding your data:

```
     public class PintrestItem {
     String url;
     public PintrestItem(String url,String name){
         this.url=url;
         this.name=name;
     }
     public String getUrl() {
         return url;
     }

    public String getName(){
        return name;
    }
     String name;
}
```

3. Create a layout file to hold RecyclerView items:

```
<ImageView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:adjustViewBounds="true"
   android:scaleType="centerCrop"
   android:id="@+id/imageView"/>
    <TextView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:gravity="center"
       android:id="@+id/name"
       android:layout_gravity="center"
       android:textColor="@android:color/white"/>
```

4. Create the adapter class for the RecyclerView:

```
public class PintrestAdapter extends
RecyclerView.Adapter<PintrestAdapter.PintrestViewHolder>{
    private ArrayList<PintrestItem>images;
    Picasso picasso;
    Context context;
    public PintrestAdapter(ArrayList<PintrestItem>images,Context context){
        this.images=images;
        picasso=Picasso.with(context);
        this.context=context;

    }

    @Override
    public PintrestViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view=
LayoutInflater.from(parent.getContext()).inflate(R.layout.pintrest_layout_item,parent,false);
        return new PintrestViewHolder(view);
    }

    @Override
    public void onBindViewHolder(PintrestViewHolder holder, int position) {
          picasso.load(images.get(position).getUrl()).into(holder.imageView);
          holder.tv.setText(images.get(position).getName());
    }

    @Override
    public int getItemCount() {
        return images.size();
    }

    public class PintrestViewHolder extends RecyclerView.ViewHolder{
        ImageView imageView;
        TextView tv;
        public PintrestViewHolder(View itemView) {
            super(itemView);
            imageView=(ImageView)itemView.findViewById(R.id.imageView);
            tv=(TextView)itemView.findViewById(R.id.name);

        }
    }
}
```

5. Instantiate the RecyclerView in your activity or fragment:

```
RecyclerView recyclerView = (RecyclerView)findViewById(R.id.recyclerView);
//Create the instance of StaggeredGridLayoutManager with 2 rows i.e the span count and provide
the orientation
StaggeredGridLayoutManager layoutManager=new new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.VERTICAL);
recyclerView.setLayoutManager(layoutManager);
// Create Dummy Data and Add to your List<PintrestItem>
List<PintrestItem>items=new ArrayList<PintrestItem>
items.add(new PintrestItem("url of image you want to show","imagename"));
items.add(new PintrestItem("url of image you want to show","imagename"));
items.add(new PintrestItem("url of image you want to show","imagename"));
recyclerView.setAdapter(new PintrestAdapter(items,getContext() );
```

Don't forgot to add the Picasso dependency in your build.gradle file:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

Chapter 20: Pagination in RecyclerView

Pagination is a common issue with for a lot of mobile apps that need to deal with lists of data. Most of the mobile apps are now starting to take up the "endless page" model, where scrolling automatically loads in new content. CWAC Endless Adapter makes it really easy to use this pattern in Android applications

Section 20.1: MainActivity.java

```
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";
    private Toolbar toolbar;

    private TextView tvEmptyView;
    private RecyclerView mRecyclerView;
    private DataAdapter mAdapter;
    private LinearLayoutManager mLayoutManager;
    private int mStart=0,mEnd=20;
    private List<Student> studentList;
    private List<Student> mTempCheck;
    public static int pageNumber;
    public int total_size=0;


    protected Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pageNumber = 1;
        toolbar = (Toolbar) findViewById(R.id.toolbar);
        tvEmptyView = (TextView) findViewById(R.id.empty_view);
        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
        studentList = new ArrayList<>();
        mTempCheck=new ArrayList<>();
```

```
        handler = new Handler();
        if (toolbar != null) {
            setSupportActionBar(toolbar);
            getSupportActionBar().setTitle("Android Students");

        }

        mRecyclerView.setHasFixedSize(true);
        mLayoutManager = new LinearLayoutManager(this);
        mRecyclerView.setLayoutManager(mLayoutManager);
        mAdapter = new DataAdapter(studentList, mRecyclerView);
        mRecyclerView.setAdapter(mAdapter);
        GetGroupData("" + mStart, "" + mEnd);
        mAdapter.setOnLoadMoreListener(new OnLoadMoreListener() {
            @Override
            public void onLoadMore() {
                if( mTempCheck.size()> 0) {
                    studentList.add(null);
                    mAdapter.notifyItemInserted(studentList.size() - 1);
                    int start = pageNumber * 20;
                    start = start + 1;
                    ++ pageNumber;
                    mTempCheck.clear();
                    GetData("" + start,""+ mEnd);
                }
            }
        });
    }

    public void GetData(final String LimitStart, final String LimitEnd) {
        Map<String, String> params = new HashMap<>();
        params.put("LimitStart", LimitStart);
        params.put("Limit", LimitEnd);
        Custom_Volly_Request jsonObjReq = new Custom_Volly_Request(Request.Method.POST,
                "Your php file link", params,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        Log.d("ResponseSuccess",response.toString());
                        // handle the data from the servoce
                    }
                }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {
                VolleyLog.d("ResponseErrorVolly: " + error.getMessage());

            }});

    }
    // load initial data
    private void loadData(int start,int end,boolean notifyadapter) {
        for (int i = start; i <= end; i++) {
            studentList.add(new Student("Student " + i, "androidstudent" + i + "@gmail.com"));
            if(notifyadapter)
                mAdapter.notifyItemInserted(studentList.size());
        }
    }
}
```

OnLoadMoreListener.java

public interface OnLoadMoreListener { void onLoadMore(); }

DataAdapter.java

```
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

public class DataAdapter extends RecyclerView.Adapter {
    private final int VIEW_ITEM = 1;
    private final int VIEW_PROG = 0;

    private List<Student> studentList;

    // The minimum amount of items to have below your current scroll position
    // before loading more.
    private int visibleThreshold = 5;
    private int lastVisibleItem, totalItemCount;
    private boolean loading;
    private OnLoadMoreListener onLoadMoreListener;



    public DataAdapter(List<Student> students, RecyclerView recyclerView) {
        studentList = students;
        if (recyclerView.getLayoutManager() instanceof LinearLayoutManager) {
            final LinearLayoutManager linearLayoutManager = (LinearLayoutManager)
recyclerView.getLayoutManager();
                    recyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {
                        @Override
                        public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
                            super.onScrolled(recyclerView, dx, dy);
                            totalItemCount = linearLayoutManager.getItemCount();
                            lastVisibleItem = linearLayoutManager.findLastVisibleItemPosition();
                            if (! loading && totalItemCount <= (lastVisibleItem +
visibleThreshold)) {
                                if (onLoadMoreListener != null) {
                                    onLoadMoreListener.onLoadMore();
                                }
                                loading = true;
                            }
                        }
                    });
        }
    }

    @Override
    public int getItemViewType(int position) {

            return studentList.get(position) != null ? VIEW_ITEM : VIEW_PROG;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
```

```
        RecyclerView.ViewHolder vh;
        if (viewType == VIEW_ITEM) {
            View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.list_row, parent,
false);
            vh = new StudentViewHolder(v);
        } else {
            View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.progress_item,
parent, false);
            vh = new ProgressViewHolder(v);
        }
        return vh;
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        if (holder instanceof StudentViewHolder) {
            Student singleStudent=studentList.get(position);
            ((StudentViewHolder) holder).tvName.setText(singleStudent.getName());
            ((StudentViewHolder) holder).tvEmailId.setText(singleStudent.getEmailId());
            ((StudentViewHolder) holder).student= singleStudent;
        } else {
            ((ProgressViewHolder) holder).progressBar.setIndeterminate(true);
        }
    }

    public void setLoaded(boolean state) {
        loading = state;
    }

    @Override
    public int getItemCount() {
        return studentList.size();
    }

    public void setOnLoadMoreListener(OnLoadMoreListener onLoadMoreListener) {
        this.onLoadMoreListener = onLoadMoreListener;
    }


    //
    public static class StudentViewHolder extends RecyclerView.ViewHolder {
        public TextView tvName;

        public TextView tvEmailId;

        public Student student;

        public StudentViewHolder(View v) {
            super(v);
            tvName = (TextView) v.findViewById(R.id.tvName);
            tvEmailId = (TextView) v.findViewById(R.id.tvEmailId);


        }
    }

    public static class ProgressViewHolder extends RecyclerView.ViewHolder {
        public ProgressBar progressBar;

        public ProgressViewHolder(View v) {
            super(v);
            progressBar = (ProgressBar) v.findViewById(R.id.progressBar1);
```

```
            }
        }

}
```

Chapter 21: ImageView

Parameter	Description
`resId`	your Image file name in the `res` folder (usually in `drawable` folder)

ImageView (`android.widget.ImageView`) is a View for displaying and manipulating image resources, such as Drawables and Bitmaps.

Some effects, discussed in this topic, can be applied to the image. The image source can be set in XML file (`layout` folder) or by programmatically in Java code.

Section 21.1: Set tint

Set a tinting color for the image. By default, the tint will blend using `SRC_ATOP` mode.

set tint using XML attribute:

```
android:tint="#009c38"
```

Note: Must be a color value, in the form of `"#rgb"`, `"#argb"`, `"#rrggbb"`, or `"#aarrggbb"`.

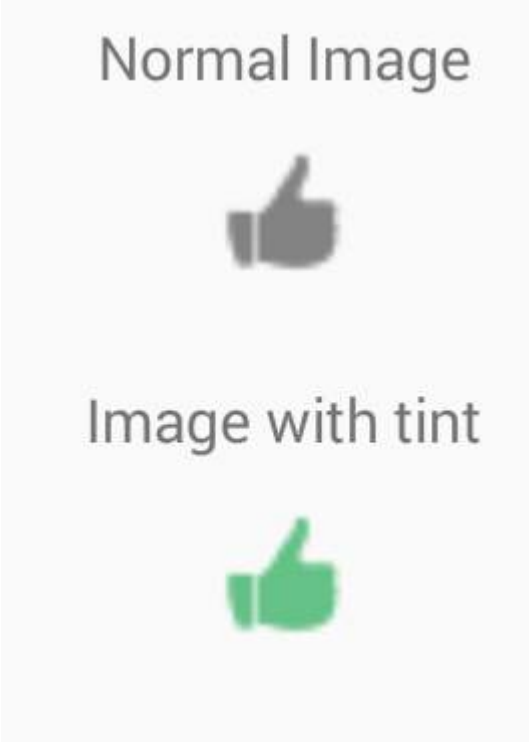
set tint programmatically:

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

and you can clear this color filter:

```
imgExample.clearColorFilter();
```

Example:



Section 21.2: Set alpha

"alpha" is used to specify the opacity for an image.

set alpha using XML attribute:

```
android:alpha="0.5"
```

Note: takes float value from 0 (transparent) to 1 (fully visible)

set alpha programmatically:

```
imgExample.setAlpha(0.5f);
```

Section 21.3: Set Scale Type

Controls how the image should be resized or moved to match the size of `ImageView`.

XML attribute:

```
android:scaleType="..."
```

i will illustrate different scale types with a square `ImageView` which has a black background and we want to display a rectangular drawable in white background in `ImageView`.

```
<ImageView
android:id="@+id/imgExample"
android:layout_width="200dp"
android:layout_height="200dp"
android:background="#000"
android:src="@drawable/android2"
android:scaleType="..."/>
```

scaleType must be one of the following values:

1. `center`:Center the image in the view, but perform no scaling.

2. `centerCrop`: Scale the image uniformly (maintain the image's aspect ratio) so both dimensions (width and height) of the image will be equal to or larger than the corresponding dimension of the view (minus padding). The image is then centered in the view.

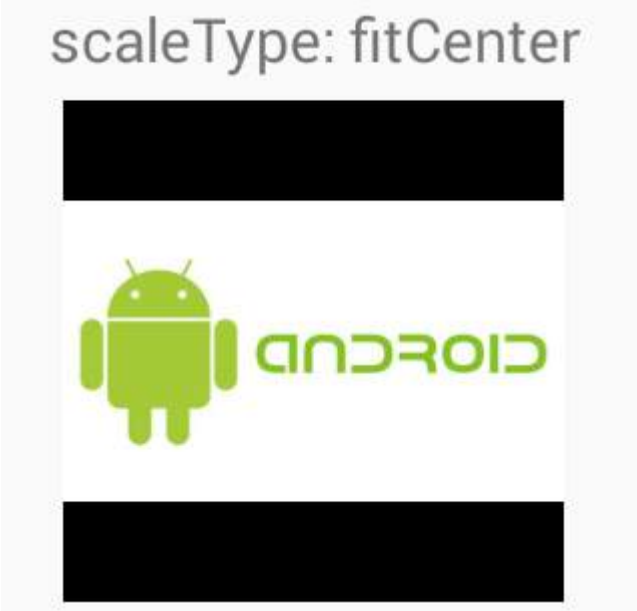
3. `centerInside`: Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or less than the corresponding dimension of the view (minus padding). The image is then centered in the view.

4. `matrix` : Scale using the image matrix when drawing.

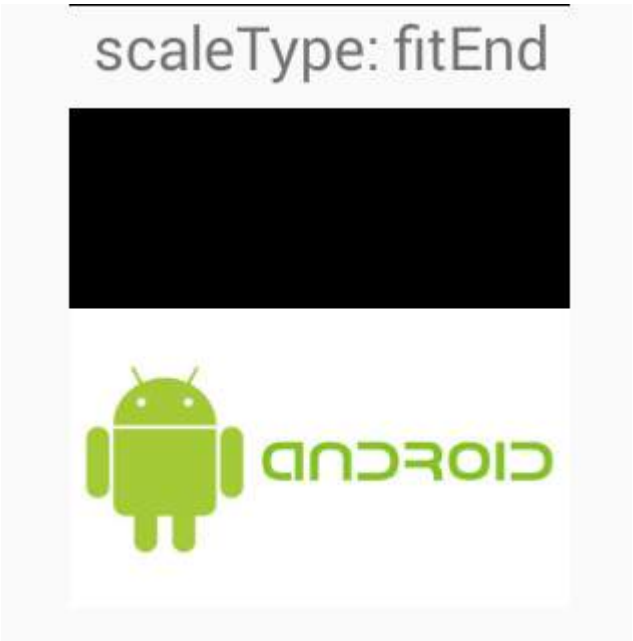
5. `fitXY`: Scale the image using FILL.

6. `fitStart`: Scale the image using START.

7. `fitCenter`: Scale the image using CENTER.



8. `fitEnd`: Scale the image using END.



Section 21.4: ImageView ScaleType - Center

The image contained in the ImageView may not fit the exact size given to the container. In that case, the framework allows you to resize the image in a number of ways.

Center

```
<ImageView android:layout_width="20dp"
      android:layout_height="20dp"
      android:src="@mipmap/ic_launcher"
      android:id="@+id/imageView"
      android:scaleType="center"
      android:background="@android:color/holo_orange_light"/>
```

This will not resize the image, and it will center it inside the container *(Orange = container)*

In case that the ImageView is smaller than the image, the image will not be resized and you will only be able to see a part of it

strong text

Section 21.5: ImageView ScaleType - CenterCrop

Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or larger than the corresponding dimension of the view (minus padding).

Official Docs

When the image matches the proportions of the container:

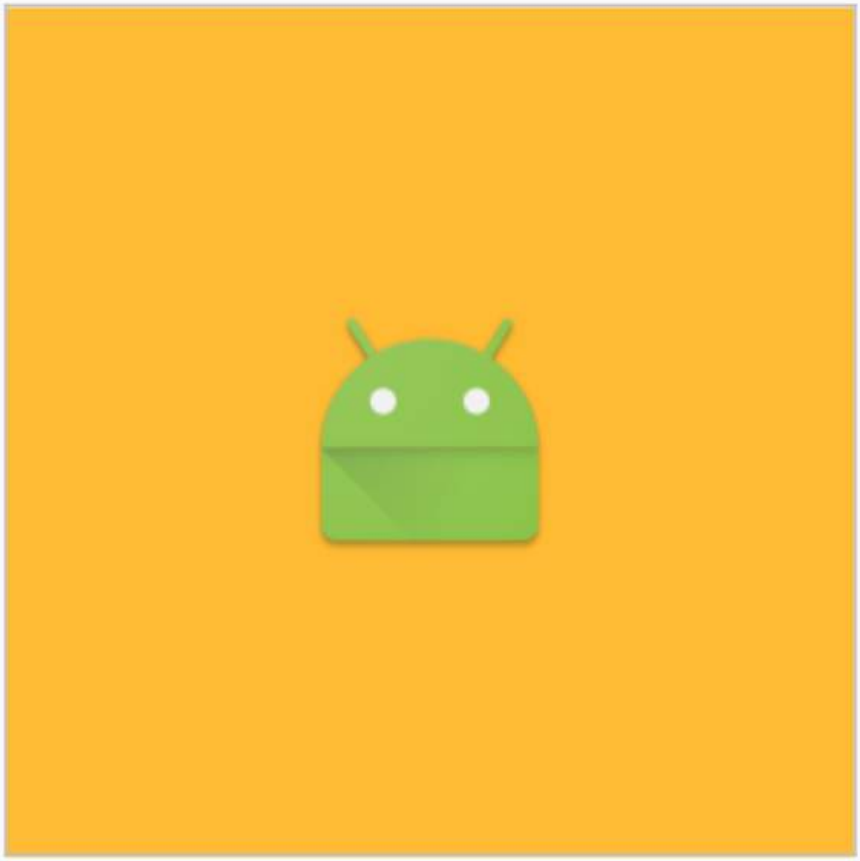
When the image is wider than the container it will expand it to the bigger size (in this case height) and adjust the width of the image without changing it's proportions, causing it to crop.

Section 21.6: ImageView ScaleType - CenterInside

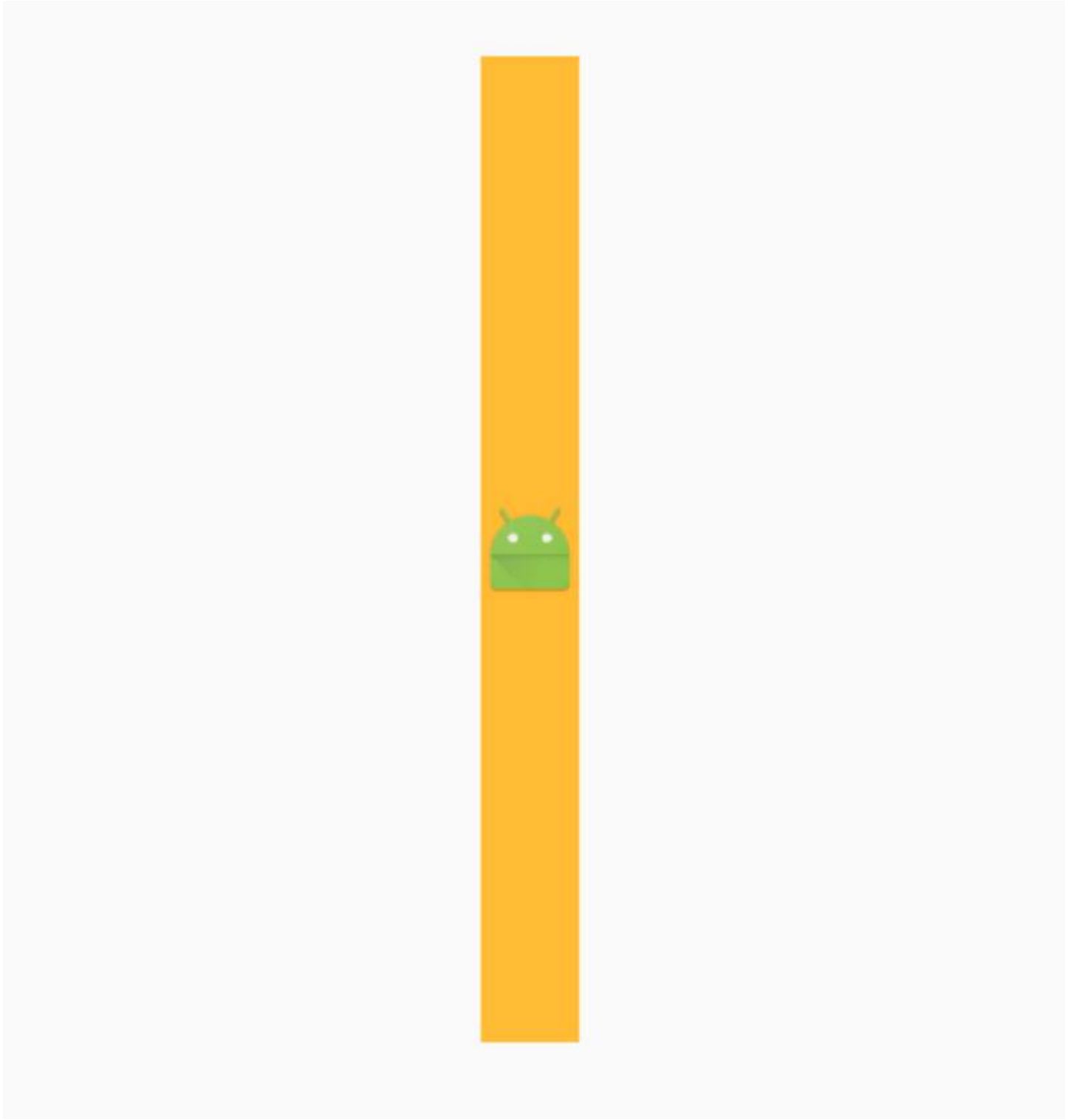
Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or less than the corresponding dimension of the view (minus padding).

Official Docs

It will center the image and resize it to the smaller size, if both container sizes are bigger it will act the same as center.



But if one of the sizes are small, it will fit to that size.



Section 21.7: ImageView ScaleType - FitStart and FitEnd

Scale the image using START.

Scale the image using END.

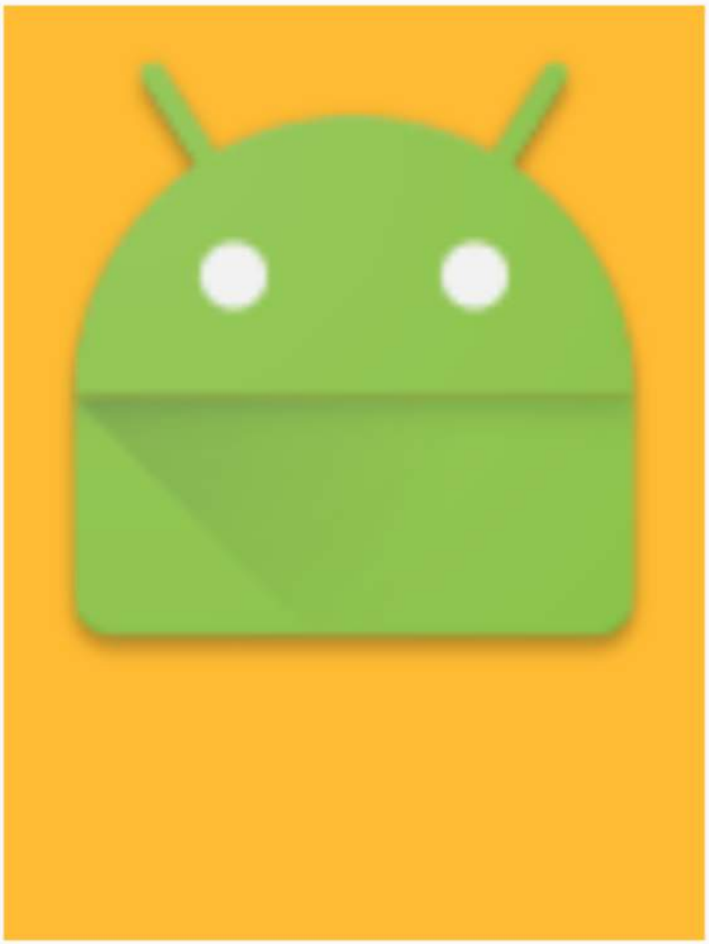
Official Docs

FitStart

This will fit to the smallest size of the container, and it will align it to the start.

```
<ImageView android:layout_width="200dp"
           android:layout_height="200dp"
```

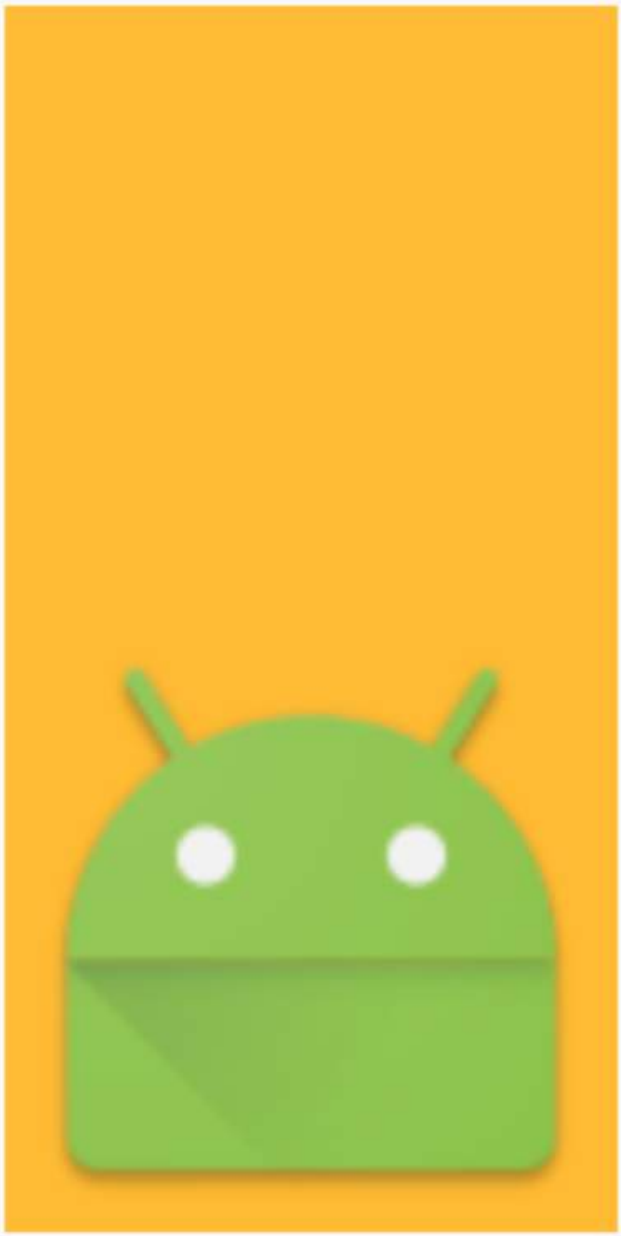
```
        android:src="@mipmap/ic_launcher"
        android:id="@+id/imageView"
        android:scaleType="fitStart"
        android:layout_gravity="center"
        android:background="@android:color/holo_orange_light"/>
```

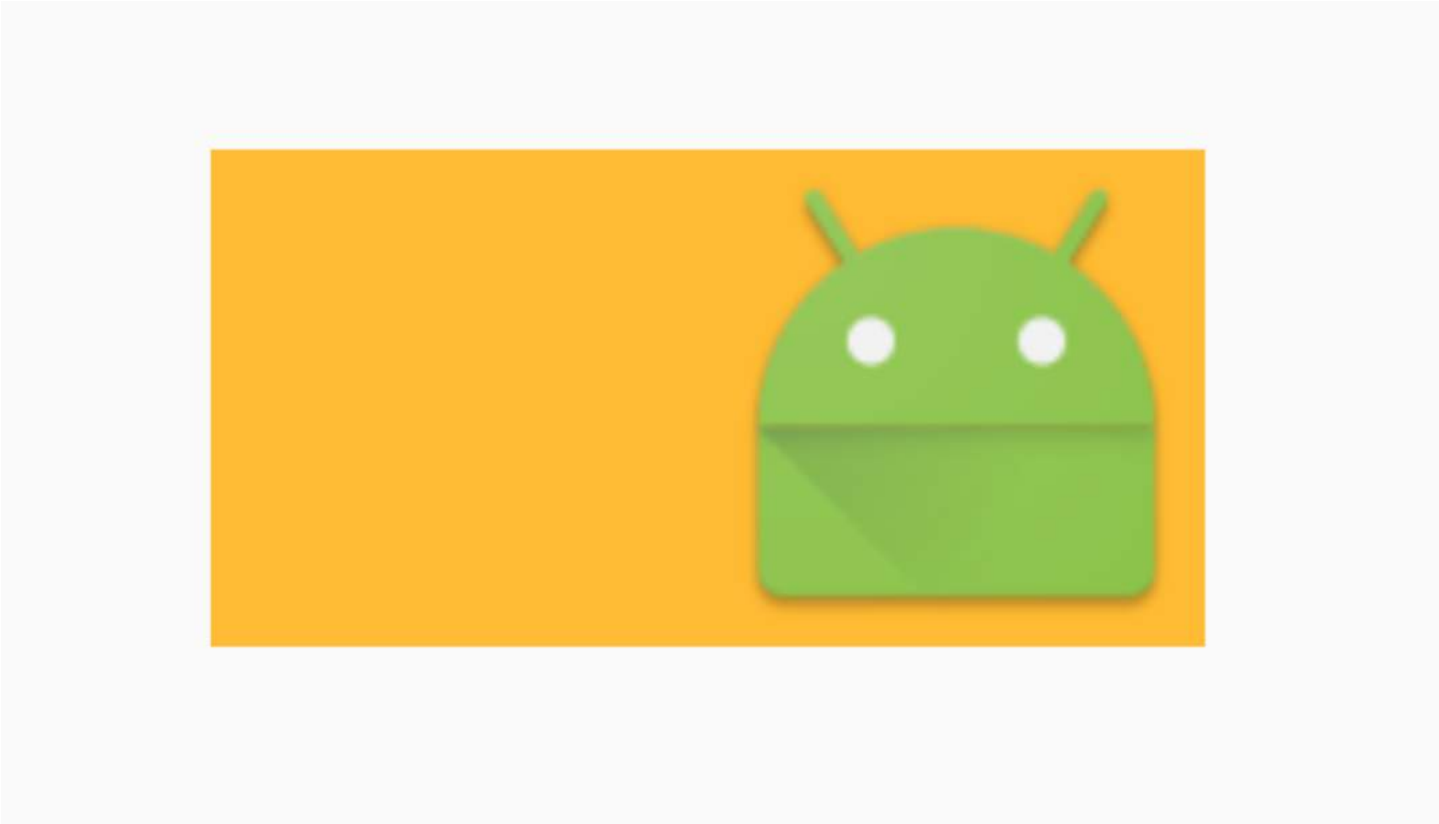


FitEnd

This will fit to the smallest size of the container, and it will align it to the end.

```
<ImageView android:layout_width="200dp"
           android:layout_height="100dp"
           android:src="@mipmap/ic_launcher"
           android:id="@+id/imageView"
           android:scaleType="fitEnd"
           android:layout_gravity="center"
           android:background="@android:color/holo_orange_light"/>
```





Section 21.8: ImageView ScaleType - FitCenter

Scale the image using CENTER.

Official Docs

This expands the image to try to match the container and it will align it to the center, it will fit to the smaller size.

Bigger height (fit to width)

Same width and height.

Section 21.9: Set Image Resource

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 ...
 />
```

set a drawable as content of `ImageView` using XML attribute:

```
android:src="@drawable/android2"
```

set a drawable programmatically:

```
ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
```

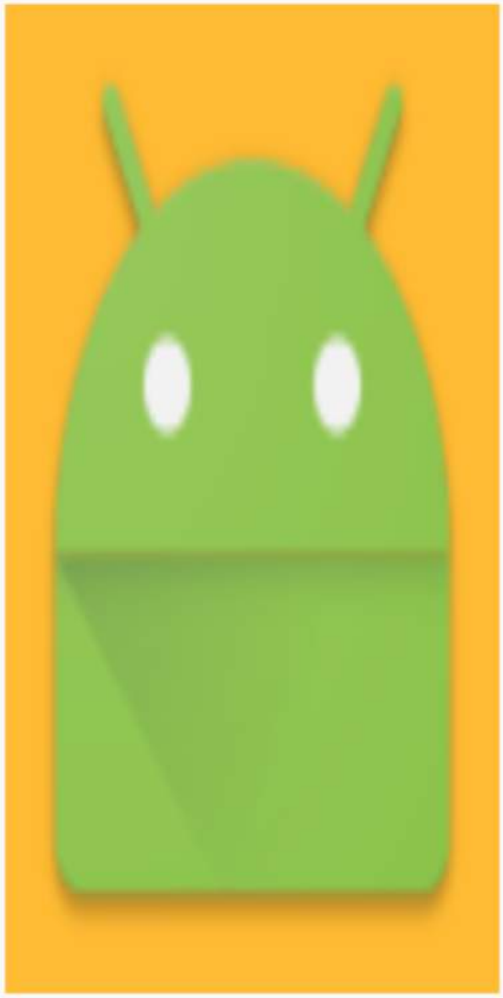
```
imgExample.setImageResource(R.drawable.android2);
```

Section 21.10: ImageView ScaleType - FitXy

Scale the image using FILL.

Official Docs

```
<ImageView android:layout_width="100dp"
          android:layout_height="200dp"
          android:src="@mipmap/ic_launcher"
          android:id="@+id/imageView"
          android:scaleType="fitXY"
          android:layout_gravity="center"
          android:background="@android:color/holo_orange_light"/>
```



Section 21.11: MLRoundedImageView.java

Copy and Paste following class in your package:

```
public class MLRoundedImageView extends ImageView {

    public MLRoundedImageView(Context context) {
        super(context);
    }

    public MLRoundedImageView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public MLRoundedImageView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }
```

```
    @Override
    protected void onDraw(Canvas canvas) {

        Drawable drawable = getDrawable();

        if (drawable == null) {
            return;
        }

        if (getWidth() == 0 || getHeight() == 0) {
            return;
        }
        Bitmap b = ((BitmapDrawable) drawable).getBitmap();
        Bitmap bitmap = b.copy(Bitmap.Config.ARGB_8888, true);

        int w = getWidth(), h = getHeight();

        Bitmap roundBitmap = getCroppedBitmap(bitmap, w);
        canvas.drawBitmap(roundBitmap, 0, 0, null);

    }

    public static Bitmap getCroppedBitmap(Bitmap bmp, int radius) {
        Bitmap sbmp;

        if (bmp.getWidth() != radius || bmp.getHeight() != radius) {
            float smallest = Math.min(bmp.getWidth(), bmp.getHeight());
            float factor = smallest / radius;
            sbmp = Bitmap.createScaledBitmap(bmp, (int)(bmp.getWidth() / factor),
(int)(bmp.getHeight() / factor), false);
        } else {
            sbmp = bmp;
        }

        Bitmap output = Bitmap.createBitmap(radius, radius,
                Config.ARGB_8888);
        Canvas canvas = new Canvas(output);

        final int color = 0xffa19774;
        final Paint paint = new Paint();
        final Rect rect = new Rect(0, 0, radius, radius);

        paint.setAntiAlias(true);
        paint.setFilterBitmap(true);
        paint.setDither(true);
        canvas.drawARGB(0, 0, 0, 0);
        paint.setColor(Color.parseColor("#BAB399"));
        canvas.drawCircle(radius / 2 + 0.7f,
                radius / 2 + 0.7f, radius / 2 + 0.1f, paint);
        paint.setXfermode(new PorterDuffXfermode(Mode.SRC_IN));
        canvas.drawBitmap(sbmp, rect, rect, paint);

        return output;
    }
}
```

Use this Class in XML with package name instead of `ImageView`

```
<com.androidbuts.example.MLRoundedImageView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
```

```
        android:src="@mipmap/ic_launcher" />
```

Chapter 22: VideoView

Section 22.1: Play video from URL with using VideoView

```
videoView.setVideoURI(Uri.parse("http://example.com/examplevideo.mp4"));
videoView.requestFocus();

videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
     @Override
     public void onCompletion(MediaPlayer mediaPlayer) {
     }
});

videoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mediaPlayer) {
         videoView.start();
         mediaPlayer.setOnVideoSizeChangedListener(new MediaPlayer.OnVideoSizeChangedListener() {
               @Override
               public void onVideoSizeChanged(MediaPlayer mp, int width, int height) {
                     MediaController mediaController = new MediaController(ActivityName.this);
                     videoView.setMediaController(mediaController);
                     mediaController.setAnchorView(videoView);
               }
         });
    }
});

videoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
      @Override
      public boolean onError(MediaPlayer mediaPlayer, int i, int i1) {
           return false;
      }
});
```

Section 22.2: VideoView Create

Find VideoView in Activity and add video into it.

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

Start playing video.

```
videoView.start();
```

Define VideoView in XML Layout file.

```
<VideoView
   android:id="@+id/videoView"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_gravity="center" />
```

Chapter 23: Optimized VideoView

Playing a video using a `VideoView` which extends `SurfaceView` inside of a row of a `ListView` seems to work at first, until the user tries to scroll the list. As soon as the list starts to scroll, the video turns black (sometimes displays white). It keeps playing in the background but you can't see it anymore because it renders the rest of the video as a black box. With the custom Optimized VideoView, the videos will play on scroll in the `ListView` just like our Instagram, Facebook, Twitter.

Section 23.1: Optimized VideoView in ListView

This the custom `VideoView` that you need to have it in your package.

Custom VideoView Layout:

```
<your.packagename.VideoView
   android:id="@+id/video_view"
   android:layout_width="300dp"
   android:layout_height="300dp" />
```

Code for custom Optimized `VideoView`:

```
package your.package.com.whateveritis;

import android.content.Context;
import android.content.Intent;
import android.graphics.SurfaceTexture;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnErrorListener;
import android.media.MediaPlayer.OnInfoListener;
import android.net.Uri;
import android.util.AttributeSet;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.widget.MediaController;
import android.widget.MediaController.MediaPlayerControl;

import java.io.IOException;


/**
 * VideoView is used to play video, just like
 * {@link android.widget.VideoView VideoView}. We define a custom view, because
 * we could not use {@link android.widget.VideoView VideoView} in ListView. <br/>
 * VideoViews inside ScrollViews do not scroll properly. Even if you use the
 * workaround to set the background color, the MediaController does not scroll
 * along with the VideoView. Also, the scrolling video looks horrendous with the
 * workaround, lots of flickering.
 *
 * @author leo
 */
public class VideoView extends TextureView implements MediaPlayerControl {
```

```
    private static final String TAG = "tag";

    // all possible internal states
    private static final int STATE_ERROR = -1;
    private static final int STATE_IDLE = 0;
    private static final int STATE_PREPARING = 1;
    private static final int STATE_PREPARED = 2;
    private static final int STATE_PLAYING = 3;
    private static final int STATE_PAUSED = 4;
    private static final int STATE_PLAYBACK_COMPLETED = 5;

    // currentState is a VideoView object's current state.
    // targetState is the state that a method caller intends to reach.
    // For instance, regardless the VideoView object's current state,
    // calling pause() intends to bring the object to a target state
    // of STATE_PAUSED.
    private int mCurrentState = STATE_IDLE;
    private int mTargetState = STATE_IDLE;

    // Stuff we need for playing and showing a video
    private MediaPlayer mMediaPlayer;
    private int mVideoWidth;
    private int mVideoHeight;
    private int mSurfaceWidth;
    private int mSurfaceHeight;
    private SurfaceTexture mSurfaceTexture;
    private Surface mSurface;
    private MediaController mMediaController;
    private MediaPlayer.OnCompletionListener mOnCompletionListener;
    private MediaPlayer.OnPreparedListener mOnPreparedListener;

    private MediaPlayer.OnErrorListener mOnErrorListener;
    private MediaPlayer.OnInfoListener mOnInfoListener;

    private int mSeekWhenPrepared; // recording the seek position while
    // preparing
    private int mCurrentBufferPercentage;
    private int mAudioSession;
    private Uri mUri;

    private Context mContext;

    public VideoView(final Context context) {
        super(context);
        mContext = context;
        initVideoView();
    }

    public VideoView(final Context context, final AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        initVideoView();
    }

    public VideoView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        mContext = context;
        initVideoView();
    }

    public void initVideoView() {
        mVideoHeight = 0;
```

```
        mVideoWidth = 0;
        setFocusable(false);
        setSurfaceTextureListener(mSurfaceTextureListener);
    }

    public int resolveAdjustedSize(int desiredSize, int measureSpec) {
        int result = desiredSize;
        int specMode = MeasureSpec.getMode(measureSpec);
        int specSize = MeasureSpec.getSize(measureSpec);

        switch (specMode) {
            case MeasureSpec.UNSPECIFIED:
            /*
             * Parent says we can be as big as we want. Just don't be larger
             * than max size imposed on ourselves.
             */
                result = desiredSize;
                break;

            case MeasureSpec.AT_MOST:
            /*
             * Parent says we can be as big as we want, up to specSize. Don't be
             * larger than specSize, and don't be larger than the max size
             * imposed on ourselves.
             */
                result = Math.min(desiredSize, specSize);
                break;

            case MeasureSpec.EXACTLY:
                // No choice. Do what we are told.
                result = specSize;
                break;
        }
        return result;
    }

    public void setVideoPath(String path) {
        Log.d(TAG, "Setting video path to: " + path);
        setVideoURI(Uri.parse(path));
    }

    public void setVideoURI(Uri _videoURI) {
        mUri = _videoURI;
        mSeekWhenPrepared = 0;
        requestLayout();
        invalidate();
        openVideo();
    }

    public Uri getUri() {
        return mUri;
    }

    public void setSurfaceTexture(SurfaceTexture _surfaceTexture) {
        mSurfaceTexture = _surfaceTexture;
    }

    public void openVideo() {
        if ((mUri == null) || (mSurfaceTexture == null)) {
            Log.d(TAG, "Cannot open video, uri or surface texture is null.");
            return;
        }
```

```
        // Tell the music playback service to pause
        // TODO: these constants need to be published somewhere in the
        // framework.
        Intent i = new Intent("com.android.music.musicservicecommand");
        i.putExtra("command", "pause");
        mContext.sendBroadcast(i);
        release(false);
        try {
            mSurface = new Surface(mSurfaceTexture);
            mMediaPlayer = new MediaPlayer();
            if (mAudioSession != 0) {
                mMediaPlayer.setAudioSessionId(mAudioSession);
            } else {
                mAudioSession = mMediaPlayer.getAudioSessionId();
            }

            mMediaPlayer.setOnBufferingUpdateListener(mBufferingUpdateListener);
            mMediaPlayer.setOnCompletionListener(mCompleteListener);
            mMediaPlayer.setOnPreparedListener(mPreparedListener);
            mMediaPlayer.setOnErrorListener(mErrorListener);
            mMediaPlayer.setOnInfoListener(mOnInfoListener);
            mMediaPlayer.setOnVideoSizeChangedListener(mVideoSizeChangedListener);

            mMediaPlayer.setSurface(mSurface);
            mCurrentBufferPercentage = 0;
            mMediaPlayer.setDataSource(mContext, mUri);

            mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
            mMediaPlayer.setScreenOnWhilePlaying(true);

            mMediaPlayer.prepareAsync();
            mCurrentState = STATE_PREPARING;
        } catch (IllegalStateException e) {
            mCurrentState = STATE_ERROR;
            mTargetState = STATE_ERROR;
            String msg = (e.getMessage()==null)?"":e.getMessage();
            Log.i("",msg);  // TODO auto-generated catch block
        } catch (IOException e) {
            mCurrentState = STATE_ERROR;
            mTargetState = STATE_ERROR;
            String msg = (e.getMessage()==null)?"":e.getMessage();
            Log.i("",msg);  // TODO auto-generated catch block
        }
    }

    public void stopPlayback() {
        if (mMediaPlayer != null) {
            mMediaPlayer.stop();
            mMediaPlayer.release();
            mMediaPlayer = null;
            if (null != mMediaControllListener) {
                mMediaControllListener.onStop();
            }
        }
    }

    public void setMediaController(MediaController controller) {
        if (mMediaController != null) {
            mMediaController.hide();
        }
        mMediaController = controller;
        attachMediaController();
```

```
    }

    private void attachMediaController() {
        if (mMediaPlayer != null && mMediaController != null) {
            mMediaController.setMediaPlayer(this);
            View anchorView = this.getParent() instanceof View ? (View) this.getParent() : this;
            mMediaController.setAnchorView(anchorView);
            mMediaController.setEnabled(isInPlaybackState());
        }
    }

    private void release(boolean cleartargetstate) {
        Log.d(TAG, "Releasing media player.");
        if (mMediaPlayer != null) {
            mMediaPlayer.reset();
            mMediaPlayer.release();
            mMediaPlayer = null;
            mCurrentState = STATE_IDLE;
            if (cleartargetstate) {
                mTargetState = STATE_IDLE;
            }
        } else {
            Log.d(TAG, "Media player was null, did not release.");
        }
    }

    @Override
    protected void onMeasure(final int widthMeasureSpec, final int heightMeasureSpec) {
        // Will resize the view if the video dimensions have been found.
        // video dimensions are found after onPrepared has been called by
        // MediaPlayer
        int width = getDefaultSize(mVideoWidth, widthMeasureSpec);
        int height = getDefaultSize(mVideoHeight, heightMeasureSpec);
        if ((mVideoWidth > 0) && (mVideoHeight > 0)) {
            if ((mVideoWidth * height) > (width * mVideoHeight)) {
                Log.d(TAG, "Video too tall, change size.");
                height = (width * mVideoHeight) / mVideoWidth;
            } else if ((mVideoWidth * height) < (width * mVideoHeight)) {
                Log.d(TAG, "Video too wide, change size.");
                width = (height * mVideoWidth) / mVideoHeight;
            } else {
                Log.d(TAG, "Aspect ratio is correct.");
            }
        }
        setMeasuredDimension(width, height);
    }

    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        if (isInPlaybackState() && mMediaController != null) {
            toggleMediaControlsVisiblity();
        }
        return false;
    }

    @Override
    public boolean onTrackballEvent(MotionEvent ev) {
        if (isInPlaybackState() && mMediaController != null) {
            toggleMediaControlsVisiblity();
        }
        return false;
    }
```

```
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        boolean isKeyCodeSupported = keyCode != KeyEvent.KEYCODE_BACK && keyCode !=
KeyEvent.KEYCODE_VOLUME_UP && keyCode != KeyEvent.KEYCODE_VOLUME_DOWN
                && keyCode != KeyEvent.KEYCODE_VOLUME_MUTE && keyCode != KeyEvent.KEYCODE_MENU &&
keyCode != KeyEvent.KEYCODE_CALL
                && keyCode != KeyEvent.KEYCODE_ENDCALL;
        if (isInPlaybackState() && isKeyCodeSupported && mMediaController != null) {
            if (keyCode == KeyEvent.KEYCODE_HEADSETHOOK || keyCode ==
KeyEvent.KEYCODE_MEDIA_PLAY_PAUSE) {
                if (mMediaPlayer.isPlaying()) {
                    pause();
                    mMediaController.show();
                } else {
                    start();
                    mMediaController.hide();
                }
                return true;
            } else if (keyCode == KeyEvent.KEYCODE_MEDIA_PLAY) {
                if (!mMediaPlayer.isPlaying()) {
                    start();
                    mMediaController.hide();
                }
                return true;
            } else if (keyCode == KeyEvent.KEYCODE_MEDIA_STOP || keyCode ==
KeyEvent.KEYCODE_MEDIA_PAUSE) {
                if (mMediaPlayer.isPlaying()) {
                    pause();
                    mMediaController.show();
                }
                return true;
            } else {
                toggleMediaControlsVisiblity();
            }
        }

        return super.onKeyDown(keyCode, event);
    }

    private void toggleMediaControlsVisiblity() {
        if (mMediaController.isShowing()) {
            mMediaController.hide();
        } else {
            mMediaController.show();
        }
    }

    public void start() {
        // This can potentially be called at several points, it will go through
        // when all conditions are ready
        // 1. When setting the video URI
        // 2. When the surface becomes available
        // 3. From the activity
        if (isInPlaybackState()) {
            mMediaPlayer.start();
            mCurrentState = STATE_PLAYING;
            if (null != mMediaControllListener) {
                mMediaControllListener.onStart();
            }
        } else {
            Log.d(TAG, "Could not start. Current state " + mCurrentState);
        }
```

```
        mTargetState = STATE_PLAYING;
    }

    public void pause() {
        if (isInPlaybackState()) {
            if (mMediaPlayer.isPlaying()) {
                mMediaPlayer.pause();
                mCurrentState = STATE_PAUSED;
                if (null != mMediaControllListener) {
                    mMediaControllListener.onPause();
                }
            }
        }
        mTargetState = STATE_PAUSED;
    }

    public void suspend() {
        release(false);
    }

    public void resume() {
        openVideo();
    }

    @Override
    public int getDuration() {
        if (isInPlaybackState()) {
            return mMediaPlayer.getDuration();
        }

        return -1;
    }

    @Override
    public int getCurrentPosition() {
        if (isInPlaybackState()) {
            return mMediaPlayer.getCurrentPosition();
        }
        return 0;
    }

    @Override
    public void seekTo(int msec) {
        if (isInPlaybackState()) {
            mMediaPlayer.seekTo(msec);
            mSeekWhenPrepared = 0;
        } else {
            mSeekWhenPrepared = msec;
        }
    }

    @Override
    public boolean isPlaying() {
        return isInPlaybackState() && mMediaPlayer.isPlaying();
    }

    @Override
    public int getBufferPercentage() {
        if (mMediaPlayer != null) {
            return mCurrentBufferPercentage;
        }
        return 0;
```

```
    }

    private boolean isInPlaybackState() {
        return ((mMediaPlayer != null) && (mCurrentState != STATE_ERROR) && (mCurrentState !=
STATE_IDLE) && (mCurrentState != STATE_PREPARING));
    }

    @Override
    public boolean canPause() {
        return false;
    }

    @Override
    public boolean canSeekBackward() {
        return false;
    }

    @Override
    public boolean canSeekForward() {
        return false;
    }

    @Override
    public int getAudioSessionId() {
        if (mAudioSession == 0) {
            MediaPlayer foo = new MediaPlayer();
            mAudioSession = foo.getAudioSessionId();
            foo.release();
        }
        return mAudioSession;
    }

    // Listeners
    private MediaPlayer.OnBufferingUpdateListener mBufferingUpdateListener = new
MediaPlayer.OnBufferingUpdateListener() {
        @Override
        public void onBufferingUpdate(final MediaPlayer mp, final int percent) {
            mCurrentBufferPercentage = percent;
        }
    };

    private MediaPlayer.OnCompletionListener mCompleteListener = new
MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(final MediaPlayer mp) {
            mCurrentState = STATE_PLAYBACK_COMPLETED;
            mTargetState = STATE_PLAYBACK_COMPLETED;
            mSurface.release();

            if (mMediaController != null) {
                mMediaController.hide();
            }

            if (mOnCompletionListener != null) {
                mOnCompletionListener.onCompletion(mp);
            }

            if (mMediaControllListener != null) {
                mMediaControllListener.onComplete();
            }
        }
    };
```

```
    private MediaPlayer.OnPreparedListener mPreparedListener = new MediaPlayer.OnPreparedListener()
{
        @Override
        public void onPrepared(final MediaPlayer mp) {
            mCurrentState = STATE_PREPARED;

            mMediaController = new MediaController(getContext());

            if (mOnPreparedListener != null) {
                mOnPreparedListener.onPrepared(mMediaPlayer);
            }
            if (mMediaController != null) {
                mMediaController.setEnabled(true);
                //mMediaController.setAnchorView(getRootView());
            }

            mVideoWidth = mp.getVideoWidth();
            mVideoHeight = mp.getVideoHeight();

            int seekToPosition = mSeekWhenPrepared; // mSeekWhenPrepared may be
            // changed after seekTo()
            // call
            if (seekToPosition != 0) {
                seekTo(seekToPosition);
            }

            requestLayout();
            invalidate();
            if ((mVideoWidth != 0) && (mVideoHeight != 0)) {
                if (mTargetState == STATE_PLAYING) {
                    mMediaPlayer.start();
                    if (null != mMediaControllListener) {
                        mMediaControllListener.onStart();
                    }
                }
            } else {
                if (mTargetState == STATE_PLAYING) {
                    mMediaPlayer.start();
                    if (null != mMediaControllListener) {
                        mMediaControllListener.onStart();
                    }
                }
            }
        }
    };

    private MediaPlayer.OnVideoSizeChangedListener mVideoSizeChangedListener = new
MediaPlayer.OnVideoSizeChangedListener() {
        @Override
        public void onVideoSizeChanged(final MediaPlayer mp, final int width, final int height) {
            mVideoWidth = mp.getVideoWidth();
            mVideoHeight = mp.getVideoHeight();
            if (mVideoWidth != 0 && mVideoHeight != 0) {
                requestLayout();
            }
        }
    };

    private MediaPlayer.OnErrorListener mErrorListener = new MediaPlayer.OnErrorListener() {
        @Override
        public boolean onError(final MediaPlayer mp, final int what, final int extra) {
            Log.d(TAG, "Error: " + what + "," + extra);
```

```
            mCurrentState = STATE_ERROR;
            mTargetState = STATE_ERROR;

            if (mMediaController != null) {
                mMediaController.hide();
            }

            /* If an error handler has been supplied, use it and finish. */
            if (mOnErrorListener != null) {
                if (mOnErrorListener.onError(mMediaPlayer, what, extra)) {
                    return true;
                }
            }

            /*
             * Otherwise, pop up an error dialog so the user knows that
             * something bad has happened. Only try and pop up the dialog if
             * we're attached to a window. When we're going away and no longer
             * have a window, don't bother showing the user an error.
             */
            if (getWindowToken() != null) {

//                new AlertDialog.Builder(mContext).setMessage("Error: " + what + "," +
extra).setPositiveButton("OK", new DialogInterface.OnClickListener() {
//                    public void onClick(DialogInterface dialog, int whichButton) {
//                        /*
//                         * If we get here, there is no onError listener, so at
//                         * least inform them that the video is over.
//                         */
//                        if (mOnCompletionListener != null) {
//                            mOnCompletionListener.onCompletion(mMediaPlayer);
//                        }
//                    }
//                }).setCancelable(false).show();
            }
            return true;
        }
    };

    SurfaceTextureListener mSurfaceTextureListener = new SurfaceTextureListener() {
        @Override
        public void onSurfaceTextureAvailable(final SurfaceTexture surface, final int width, final
int height) {
            Log.d(TAG, "onSurfaceTextureAvailable.");
            mSurfaceTexture = surface;
            openVideo();
        }

        @Override
        public void onSurfaceTextureSizeChanged(final SurfaceTexture surface, final int width,
final int height) {
            Log.d(TAG, "onSurfaceTextureSizeChanged: " + width + '/' + height);
            mSurfaceWidth = width;
            mSurfaceHeight = height;
            boolean isValidState = (mTargetState == STATE_PLAYING);
            boolean hasValidSize = (mVideoWidth == width && mVideoHeight == height);
            if (mMediaPlayer != null && isValidState && hasValidSize) {
                if (mSeekWhenPrepared != 0) {
                    seekTo(mSeekWhenPrepared);
                }
                start();
            }
```

```
        }

        @Override
        public boolean onSurfaceTextureDestroyed(final SurfaceTexture surface) {

            mSurface = null;
            if (mMediaController != null)
                mMediaController.hide();
            release(true);
            return true;
        }

        @Override
        public void onSurfaceTextureUpdated(final SurfaceTexture surface) {

        }
    };

    /**
     * Register a callback to be invoked when the media file is loaded and ready
     * to go.
     *
     * @param l The callback that will be run
     */
    public void setOnPreparedListener(MediaPlayer.OnPreparedListener l) {
        mOnPreparedListener = l;
    }

    /**
     * Register a callback to be invoked when the end of a media file has been
     * reached during playback.
     *
     * @param l The callback that will be run
     */
    public void setOnCompletionListener(OnCompletionListener l) {
        mOnCompletionListener = l;
    }

    /**
     * Register a callback to be invoked when an error occurs during playback or
     * setup. If no listener is specified, or if the listener returned false,
     * VideoView will inform the user of any errors.
     *
     * @param l The callback that will be run
     */
    public void setOnErrorListener(OnErrorListener l) {
        mOnErrorListener = l;
    }

    /**
     * Register a callback to be invoked when an informational event occurs
     * during playback or setup.
     *
     * @param l The callback that will be run
     */
    public void setOnInfoListener(OnInfoListener l) {
        mOnInfoListener = l;
    }

    public static interface MediaControllListener {
        public void onStart();
```

```
        public void onPause();

        public void onStop();

        public void onComplete();
    }

    MediaControllListener mMediaControllListener;

    public void setMediaControllListener(MediaControllListener mediaControllListener) {
        mMediaControllListener = mediaControllListener;
    }


    @Override
    public void setVisibility(int visibility) {
        System.out.println("setVisibility: " + visibility);
        super.setVisibility(visibility);
    }
}
```

Help from this gitub repository. Though It has some issues as it was written 3 years ago I managed to fix them on my own as written above.

Chapter 24: WebView

WebView is a view that display web pages inside your application. By this you can add your own URL.

Section 24.1: Troubleshooting WebView by printing console messages or by remote debugging

Printing webview console messages to logcat

To handle `console` messages from web page you can override `onConsoleMessage` in `WebChromeClient`:

```
final class ChromeClient extends WebChromeClient {
    @Override
    public boolean onConsoleMessage(ConsoleMessage msg) {
        Log.d(
            "WebView",
            String.format("%s %s:%d", msg.message(), msg.lineNumber(), msg.sourceId())
        );
        return true;
    }
}
```

And set it in your activity or fragment:

```
webView.setWebChromeClient(new ChromeClient());
```

So this sample page:

```
<html>
<head>
    <script type="text/javascript">
        console.log('test message');
    </script>
</head>
<body>
</body>
</html>
```

will write log 'test message' to logcat:

> WebView: test message sample.html:4

`console.info()`, `console.warn()` and `console.error()` are also supported by chrome-client.

Remote debugging android devices with Chrome

Your can remote debug webview based application from you desktop Chrome.

Enable USB debugging on your Android device

On your Android device, open up Settings, find the Developer options section, and enable USB debugging.

Connect and discover your Android device

Open page in chrome following page: chrome://inspect/#devices

From the Inspect Devices dialog, select your device and press **inspect**. A new instance of Chrome's DevTools opens up on your development machine.

More detailed guideline and description of DevTools can be found on developers.google.com

Section 24.2: Communication from Javascript to Java (Android)

Android Activity

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        WebView webView = new WebView(this);
        setContentView(webView);

       /*
        *   Note the label Android, this is used in the Javascript side of things
        *   You can of course change this.
        */
        webView.addJavascriptInterface(new JavascriptHandler(), "Android");

        webView.loadUrl("http://example.com");
    }
}
```

Java Javascript Handler

```
import android.webkit.JavascriptInterface;

public class JavascriptHandler {

    /**
     *  Key point here is the annotation @JavascriptInterface
     *
     */
    @JavascriptInterface
    public void jsCallback() {
       // Do something
    }

    @JavascriptInterface
    public void jsCallbackTwo(String dummyData) {
       // Do something
    }
}
```

Web Page, Javascript call

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

Extra Tip

Passing in a complex data structure, a possible solution is use JSON.

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

On the Android side use your favorite JSON parser ie: JSONObject

Section 24.3: Communication from Java to Javascript

Basic Example

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {

    private Webview webView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);

       webView = new WebView(this);
       webView.getSettings().setJavaScriptEnabled(true);

       setContentView(webView);

       webView.loadUrl("http://example.com");

       /*
        * Invoke Javascript function
        */
       webView.loadUrl("javascript:testJsFunction('Hello World!')");
    }

   /**
    * Invoking a Javascript function
    */
   public void doSomething() {
     this.webView.loadUrl("javascript:testAnotherFunction('Hello World Again!')");
   }
}
```

Section 24.4: Open dialer example

If the web page a contains phone number you can make a call using your phone's dialer. This code checks for the

url which starts with tel: then make an intent to open dialer and you can make a call to the clicked phone number:

```
public boolean shouldOverrideUrlLoading(WebView view, String url) {
    if (url.startsWith("tel:")) {
            Intent intent = new Intent(Intent.ACTION_DIAL,
                    Uri.parse(url));
            startActivity(intent);
    }else if(url.startsWith("http:") || url.startsWith("https:")) {
        view.loadUrl(url);
    }
    return true;
}
```

Section 24.5: Open Local File / Create dynamic content in Webview

Layout.xml

```
             <WebView
            android:id="@+id/WebViewToDisplay"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:fadeScrollbars="false" />
```

Load data into WebViewToDisplay

```
        WebView webViewDisplay;
        StringBuffer LoadWEb1;

        webViewDisplay = (WebView) findViewById(R.id.WebViewToDisplay);
        LoadWEb1 = new StringBuffer();
        LoadWEb1.append("<html><body><h1>My First Heading</h1><p>My first paragraph.</p>");
        //Sample code to read parameters at run time
        String strName = "Test Paragraph";
        LoadWEb1.append("<br/><p>"+strName+"</p>");
        String result = LoadWEb1.append("</body></html>").toString();
                WebSettings webSettings = webViewDisplay.getSettings();
                webSettings.setJavaScriptEnabled(true);
                webViewDisplay.getSettings().setBuiltInZoomControls(true);
                if (android.os.Build.VERSION.SDK_INT >= 11){
                    webViewDisplay.setLayerType(View.LAYER_TYPE_SOFTWARE, null);
                    webViewDisplay.getSettings().setDisplayZoomControls(false);
                }

                webViewDisplay.loadDataWithBaseURL(null, result, "text/html", "utf-8",
                        null);
                //To load local file directly from assets folder use below code
                //webViewDisplay.loadUrl("file:///android_asset/aboutapp.html");
```

Section 24.6: JavaScript alert dialogs in WebView - How to make them work

By default, WebView does not implement JavaScript alert dialogs, ie. `alert()` will do nothing. In order to make you need to firstly enable JavaScript (obviously..), and then set a `WebChromeClient` to handle requests for alert dialogs from the page:

```
webView.setWebChromeClient(new WebChromeClient() {
    //Other methods for your WebChromeClient here, if needed..

    @Override
    public boolean onJsAlert(WebView view, String url, String message, JsResult result) {
        return super.onJsAlert(view, url, message, result);
    }
});
```

Here, we override `onJsAlert`, and then we call through to the super implementation, which gives us a standard Android dialog. You can also use the message and URL yourself, for example if you want to create a custom styled dialog or if you want to log them.

Chapter 25: SearchView

Section 25.1: Setting Theme for SearchView

Basically to apply a theme for SearchView extracted as `app:actionViewClass` from the `menu.xml`, we need understand that it depends completely on the style applied to the underlying Toolbar. To achieve themeing the Toolbar apply the following steps.

Create a style in the `styles.xml`

```
<style name="ActionBarThemeOverlay">
     <item name="android:textColorPrimary">@color/prim_color</item>
     <item name="colorControlNormal">@color/normal_color</item>
     <item name="colorControlHighlight">@color/high_color</item>
     <item name="android:textColorHint">@color/hint_color</item>
</style>
```

Apply the style to the Toolbar.

```
<android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        app:theme="@style/ActionBarThemeOverlay"
        app:popupTheme="@style/ActionBarThemeOverlay"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@color/colorPrimary"
        android:title="@string/title"
        tools:targetApi="m" />
```

This gives the desired color to the all the views corresponding to the Toolbar (back button, Menu icons and SearchView).

Section 25.2: SearchView in Toolbar with Fragment

menu.xml - (`res` -> `menu`)

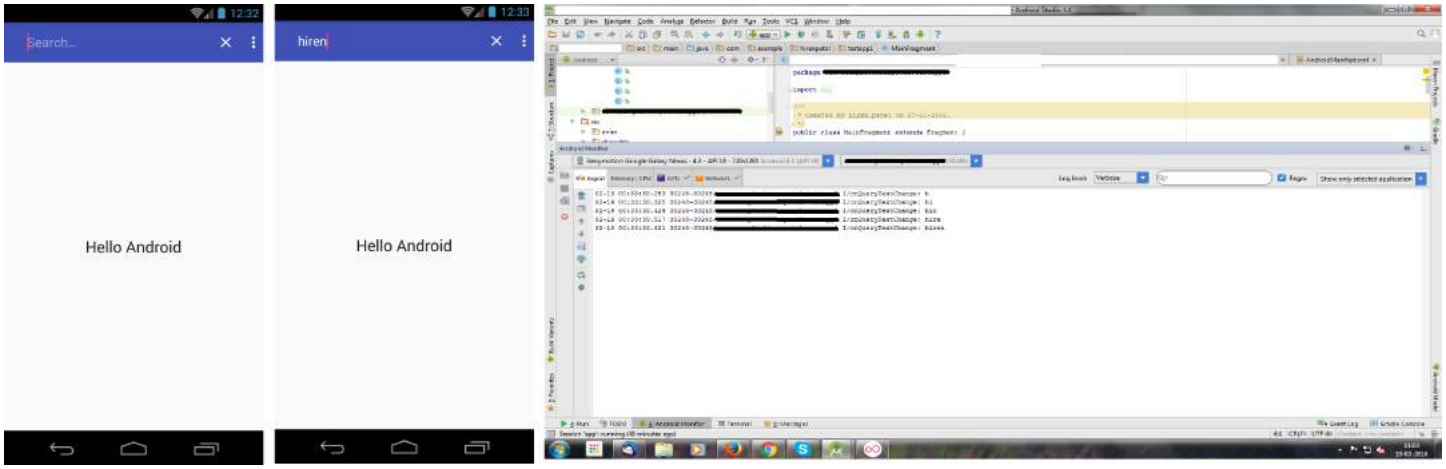
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   tools:context=".HomeActivity">

    <item
       android:id="@+id/action_search"
       android:icon="@android:drawable/ic_menu_search"
       android:title="Search"
       app:actionViewClass="android.support.v7.widget.SearchView"
       app:showAsAction="always" />

</menu>
```

MainFragment.java

```
public class MainFragment extends Fragment {

    private SearchView searchView = null;
    private SearchView.OnQueryTextListener queryTextListener;
```

```
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.fragment_main, container, false);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.menu, menu);
        MenuItem searchItem = menu.findItem(R.id.action_search);
        SearchManager searchManager = (SearchManager)
getActivity().getSystemService(Context.SEARCH_SERVICE);

        if (searchItem != null) {
            searchView = (SearchView) searchItem.getActionView();
        }
        if (searchView != null) {

searchView.setSearchableInfo(searchManager.getSearchableInfo(getActivity().getComponentName()));

            queryTextListener = new SearchView.OnQueryTextListener() {
                @Override
                public boolean onQueryTextChange(String newText) {
                    Log.i("onQueryTextChange", newText);

                    return true;
                }
                @Override
                public boolean onQueryTextSubmit(String query) {
                    Log.i("onQueryTextSubmit", query);

                    return true;
                }
            };
            searchView.setOnQueryTextListener(queryTextListener);
        }
        super.onCreateOptionsMenu(menu, inflater);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_search:
                // Not implemented here
                return false;
            default:
                break;
        }
        searchView.setOnQueryTextListener(queryTextListener);
        return super.onOptionsItemSelected(item);
    }
}
```

Reference screenshot:



Section 25.3: Appcompat SearchView with RxBindings watcher

build.gradle:

```
dependencies {
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.4.0'
}
```

menu/menu.xml:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_search" android:title="Search"
       android:icon="@android:drawable/ic_menu_search"
       app:actionViewClass="android.support.v7.widget.SearchView"
       app:showAsAction="always"/>
</menu>
```

MainActivity.java:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    setupSearchView(searchMenuItem );

    return true;
}

private void setupSearchView(MenuItem searchMenuItem) {
    SearchView searchView = (SearchView) searchMenuItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint)); // your hint here

    SearchAdapter searchAdapter = new SearchAdapter(this);
    searchView.setSuggestionsAdapter(searchAdapter);

    // optional: set the letters count after which the search will begin to 1
    // the default is 2
    try {
        int autoCompleteTextViewID = getResources().getIdentifier("android:id/search_src_text",
null, null);
```

```
        AutoCompleteTextView searchAutoCompleteTextView = (AutoCompleteTextView)
searchView.findViewById(autoCompleteTextViewID);
        searchAutoCompleteTextView.setThreshold(1);
    } catch (Exception e) {
        Logs.e(TAG, "failed to set search view letters threshold");
    }

    searchView.setOnSearchClickListener(v -> {
        // optional actions to search view expand
    });
    searchView.setOnCloseListener(() -> {
        // optional actions to search view close
        return false;
    });

    RxSearchView.queryTextChanges(searchView)
            .doOnEach(notification -> {
                CharSequence query = (CharSequence) notification.getValue();
                searchAdapter.filter(query);
            })
            .debounce(300, TimeUnit.MILLISECONDS) // to skip intermediate letters
            .flatMap(query -> MyWebService.search(query)) // make a search request
            .retry(3)
            .subscribe(results -> {
                searchAdapter.populateAdapter(results);
            });

    //optional: collapse the searchView on close
    searchView.setOnQueryTextFocusChangeListener((view, queryTextFocused) -> {
        if (!queryTextFocused) {
            collapseSearchView();
        }
    });
}
```

SearchAdapter.java

```
public class SearchAdapter extends CursorAdapter {
    private List<SearchResult> items = Collections.emptyList();

    public SearchAdapter(Activity activity) {
        super(activity, null, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }

    public void populateAdapter(List<SearchResult> items) {
        this.items = items;
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            c.addRow(new Object[]{i});
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    public void filter(CharSequence query) {
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            SearchResult result = items.get(i);
            if (result.getText().startsWith(query.toString())) {
                c.addRow(new Object[]{i});
            }
```

```
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        ViewHolder holder = (ViewHolder) view.getTag();
        int position = cursor.getPosition();
        if (position < items.size()) {
            SearchResult result = items.get(position);
            // bind your view here
        }
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View v = inflater.inflate(R.layout.search_list_item, parent, false);
        ViewHolder holder = new ViewHolder(v);

        v.setTag(holder);
        return v;
    }

    private static class ViewHolder {
        public final TextView text;

        public ViewHolder(View v) {
            this.text= (TextView) v.findViewById(R.id.text);
        }
    }
}
```

Chapter 26: BottomNavigationView

The Bottom Navigation View has been in the material design guidelines for some time, but it hasn't been easy for us to implement it into our apps.

Some applications have built their own solutions, whilst others have relied on third-party open-source libraries to get the job done.

Now the design support library is seeing the addition of this bottom navigation bar, let's take a dive into how we can use it!

Section 26.1: Basic implemetation

To add the `BottomNavigationView` follow these steps:

1. Add in your `build.gradle` the **dependency**:

```
compile 'com.android.support:design:25.1.0'
```

2. Add the `BottomNavigationView` in **your layout**:

```
<android.support.design.widget.BottomNavigationView
       xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:app="http://schemas.android.com/apk/res-auto"
       android:id="@+id/bottom_navigation"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       app:menu="@menu/bottom_navigation_menu"/>
```

3. Create the **menu** to populate the view:

```
<!-- res/menu/bottom_navigation_menu.xml -->

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/my_action1"
        android:enabled="true"
        android:icon="@drawable/my_drawable"
        android:title="@string/text"
        app:showAsAction="ifRoom" />
    ....
</menu>
```

4. Attach **a listener** for the click events:

```
//Get the view
BottomNavigationView bottomNavigationView = (BottomNavigationView)
                findViewById(R.id.bottom_navigation);
//Attach the listener
bottomNavigationView.setOnNavigationItemSelectedListener(
        new BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()) {

                    case R.id.my_action1:
```

```
                        //Do something...
                        break;

                    //...
                }
                return true;//returning false disables the Navigation bar animations
            }
        });
```

Checkout demo code at BottomNavigation-Demo

Section 26.2: Customization of BottomNavigationView

Note : I am assuming that you know about how to use ***`BottomNavigationView`****.*

This example I will explain how to add selector for **`BottomNavigationView`**. So you can state on UI for icons and texts.

Create drawable `bottom_navigation_view_selector.xml` as

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item  android:color="@color/bottom_nv_menu_selected" android:state_checked="true" />
    <item android:color="@color/bottom_nv_menu_default" />
</selector>
```

And use below attributes into **`BottomNavigationView`** in layout file

```
app:itemIconTint="@drawable/bottom_navigation_view_selector"
app:itemTextColor="@drawable/bottom_navigation_view_selector"
```

> In above example, I have used same selector `bottom_navigation_view_selector` for `app:itemIconTint` and `app:itemTextColor` both to keep text and icon colors same. But if your design has different color for text and icon, you can define 2 different selectors and use them.

Output will be similar to below

Section 26.3: Handling Enabled / Disabled states

Create Selector for Enable/Disable Menu Item.

selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/white" android:state_enabled="true" />
    <item android:color="@color/colorPrimaryDark" android:state_enabled="false" />
</selector>
```

design.xml

```
<android.support.design.widget.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    app:itemBackground="@color/colorPrimary"
    app:itemIconTint="@drawable/nav_item_color_state"
    app:itemTextColor="@drawable/nav_item_color_state"
    app:menu="@menu/bottom_navigation_main" />
```

Section 26.4: Allowing more than 3 menus

This example is strictly a workaround since, currently there is no way to disable a behaviour known as ShiftMode.

Create a function as such.

```
public static void disableMenuShiftMode(BottomNavigationView view) {
    BottomNavigationMenuView menuView = (BottomNavigationMenuView) view.getChildAt(0);
    try {
        Field shiftingMode = menuView.getClass().getDeclaredField("mShiftingMode");
        shiftingMode.setAccessible(true);
        shiftingMode.setBoolean(menuView, false);
        shiftingMode.setAccessible(false);
        for (int i = 0; i < menuView.getChildCount(); i++) {
            BottomNavigationItemView item = (BottomNavigationItemView) menuView.getChildAt(i);
            //noinspection RestrictedApi
            item.setShiftingMode(false);
            // set once again checked value, so view will be updated
            //noinspection RestrictedApi
            item.setChecked(item.getItemData().isChecked());
        }
    } catch (NoSuchFieldException e) {
        Log.e("BNVHelper", "Unable to get shift mode field", e);
    } catch (IllegalAccessException e) {
        Log.e("BNVHelper", "Unable to change value of shift mode", e);
    }
}
```

This disables the Shifting behaviour of the menu when item count exceeds 3 nos.

USAGE

```
BottomNavigationView navView = (BottomNavigationView) findViewById(R.id.bottom_navigation_bar);
disableMenuShiftMode(navView);
```

Proguard Issue : Add following line proguard configuration file as well else, this wouldn't work.

```
-keepclassmembers class android.support.design.internal.BottomNavigationMenuView {
boolean mShiftingMode;
}
```

Alternatively, you can create a Class and access this method from there. See Original Reply Here

NOTE : This is a Reflection based **HOTFIX**, please update this once Google's support library is updated with a direct function call.

Chapter 27: Canvas drawing using SurfaceView

Section 27.1: SurfaceView with drawing thread

This example describes how to create a SurfaceView with a dedicated drawing thread. This implementation also handles edge cases such as manufacture specific issues as well as starting/stopping the thread to save cpu time.

```
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
/**
* Defines a custom SurfaceView class which handles the drawing thread
**/
public class BaseSurface extends SurfaceView implements SurfaceHolder.Callback,
View.OnTouchListener, Runnable
{

    /**
     * Holds the surface frame
     */
    private SurfaceHolder holder;

    /**
     * Draw thread
     */
    private Thread drawThread;

    /**
     * True when the surface is ready to draw
     */
    private boolean surfaceReady = false;


    /**
     * Drawing thread flag
     */

    private boolean drawingActive = false;

    /**
     * Paint for drawing the sample rectangle
     */
    private Paint samplePaint = new Paint();

    /**
     * Time per frame for 60 FPS
     */
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);

    private static final String LOGTAG = "surface";

    public BaseSurface(Context context, AttributeSet attrs)
```

```
    {
        super(context, attrs);
        SurfaceHolder holder = getHolder();
        holder.addCallback(this);
        setOnTouchListener(this);

        // red
        samplePaint.setColor(0xffff0000);
        // smooth edges
        samplePaint.setAntiAlias(true);
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
    {
        if (width == 0 || height == 0)
        {
            return;
        }

        // resize your UI
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder)
    {
        this.holder = holder;

        if (drawThread != null)
        {
            Log.d(LOGTAG, "draw thread still active..");
            drawingActive = false;
            try
            {
                drawThread.join();
            } catch (InterruptedException e)
            { // do nothing
            }
        }

        surfaceReady = true;
        startDrawThread();
        Log.d(LOGTAG, "Created");
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder)
    {
        // Surface is not used anymore - stop the drawing thread
        stopDrawThread();
        // and release the surface
        holder.getSurface().release();

        this.holder = null;
        surfaceReady = false;
        Log.d(LOGTAG, "Destroyed");
    }

    @Override
    public boolean onTouch(View v, MotionEvent event)
    {
        // Handle touch events
```

```
        return true;
    }

    /**
     * Stops the drawing thread
     */
    public void stopDrawThread()
    {
        if (drawThread == null)
        {
            Log.d(LOGTAG, "DrawThread is null");
            return;
        }
        drawingActive = false;
        while (true)
        {
            try
            {
                Log.d(LOGTAG, "Request last frame");
                drawThread.join(5000);
                break;
            } catch (Exception e)
            {
                Log.e(LOGTAG, "Could not join with draw thread");
            }
        }
        drawThread = null;
    }

    /**
     * Creates a new draw thread and starts it.
     */
    public void startDrawThread()
    {
        if (surfaceReady && drawThread == null)
        {
            drawThread = new Thread(this, "Draw thread");
            drawingActive = true;
            drawThread.start();
        }
    }

    @Override
    public void run()
    {
        Log.d(LOGTAG, "Draw thread started");
        long frameStartTime;
        long frameTime;

        /*
         * In order to work reliable on Nexus 7, we place ~500ms delay at the start of drawing thread
         * (AOSP - Issue 58385)
         */
        if (android.os.Build.BRAND.equalsIgnoreCase("google") &&
android.os.Build.MANUFACTURER.equalsIgnoreCase("asus") &&
android.os.Build.MODEL.equalsIgnoreCase("Nexus 7"))
        {
            Log.w(LOGTAG, "Sleep 500ms (Device: Asus Nexus 7)");
            try
            {
                Thread.sleep(500);
            } catch (InterruptedException ignored)
```

```
            {
            }
        }
        try
        {
            while (drawingActive)
            {
                if (holder == null)
                {
                    return;
                }

                frameStartTime = System.nanoTime();
                Canvas canvas = holder.lockCanvas();
                if (canvas != null)
                {
                    // clear the screen using black
                    canvas.drawARGB(255, 0, 0, 0);

                    try
                    {
                        // Your drawing here
                        canvas.drawRect(0, 0, getWidth() / 2, getHeight() / 2, samplePaint);
                    } finally
                    {

                        holder.unlockCanvasAndPost(canvas);
                    }
                }

                // calculate the time required to draw the frame in ms
                frameTime = (System.nanoTime() - frameStartTime) / 1000000;

                if (frameTime < MAX_FRAME_TIME) // faster than the max fps - limit the FPS
                {
                    try
                    {
                        Thread.sleep(MAX_FRAME_TIME - frameTime);
                    } catch (InterruptedException e)
                    {
                        // ignore
                    }
                }
            }
        } catch (Exception e)
        {
            Log.w(LOGTAG, "Exception while locking/unlocking");
        }
        Log.d(LOGTAG, "Draw thread finished");
    }
}
```

This layout only contains the custom SurfaceView and maximizes it to the screen size.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="sample.devcore.org.surfaceviewsample.MainActivity">
```

```
    <sample.devcore.org.surfaceviewsample.BaseSurface
       android:id="@+id/baseSurface"
       android:layout_width="match_parent"
       android:layout_height="match_parent"/>
</LinearLayout>
```

The activity which uses the SurfaceView is responsible for starting and stopping the drawing thread. This approach saves battery as the drawing is stopped as soon as the activity gets in the background.

```
import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity
{

    /**
     * Surface object
     */
    private BaseSurface surface;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        surface = (BaseSurface) findViewById(R.id.baseSurface);
    }

    @Override
    protected void onResume()
    {
        super.onResume();
        // start the drawing
        surface.startDrawThread();
    }

    @Override
    protected void onPause()
    {
        // stop the drawing to save cpu time
        surface.stopDrawThread();
        super.onPause();
    }
}
```

Chapter 28: Creating Custom Views

Section 28.1: Creating Custom Views

If you need a completely customized view, you'll need to subclass `View` (the superclass of all Android views) and provide your custom sizing (`onMeasure(...)`) and drawing (`onDraw(...)`) methods:

1. **Create your custom view skeleton:** this is basically the same for every custom view. Here we create the skeleton for a custom view that can draw a smiley, called `SmileyView`:

```
public class SmileyView extends View {
    private Paint mCirclePaint;
    private Paint mEyeAndMouthPaint;

    private float mCenterX;
    private float mCenterY;
    private float mRadius;
    private RectF mArcBounds = new RectF();

    public SmileyView(Context context) {
        this(context, null, 0);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        initPaints();
    }

    private void initPaints() {/* ... */}

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {/* ... */}

    @Override
    protected void onDraw(Canvas canvas) {/* ... */}
}
```

2. **Initialize your paints:** the `Paint` objects are the brushes of your virtual canvas defining how your geometric objects are rendered (e.g. color, fill and stroke style, etc.). Here we create two `Paint`s, one yellow filled paint for the circle and one black stroke paint for the eyes and the mouth:

```
private void initPaints() {
    mCirclePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mCirclePaint.setStyle(Paint.Style.FILL);
    mCirclePaint.setColor(Color.YELLOW);
    mEyeAndMouthPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    mEyeAndMouthPaint.setStyle(Paint.Style.STROKE);
    mEyeAndMouthPaint.setStrokeWidth(16 * getResources().getDisplayMetrics().density);
    mEyeAndMouthPaint.setStrokeCap(Paint.Cap.ROUND);
    mEyeAndMouthPaint.setColor(Color.BLACK);
}
```

3. **Implement your own `onMeasure(...)` method:** this is required so that the parent layouts (e.g.

`FrameLayout`) can properly align your custom view. It provides a set of `measureSpecs` that you can use to determine your view's height and width. Here we create a square by making sure that the height and width are the same:

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int w = MeasureSpec.getSize(widthMeasureSpec);
    int h = MeasureSpec.getSize(heightMeasureSpec);

    int size = Math.min(w, h);
    setMeasuredDimension(size, size);
}
```

Note that `onMeasure(...)` must contain at least one call to `setMeasuredDimension(..)` or else your custom view will crash with an `IllegalStateException`.

4. **Implement your own `onSizeChanged(...)` method:** this allows you to catch the current height and width of your custom view to properly adjust your rendering code. Here we just calculate our center and our radius:

```
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    mCenterX = w / 2f;
    mCenterY = h / 2f;
    mRadius = Math.min(w, h) / 2f;
}
```

5. **Implement your own `onDraw(...)` method:** this is where you implement the actual rendering of your view. It provides a `Canvas` object that you can draw on (see the official `Canvas` documentation for all drawing methods available).

```
@Override
protected void onDraw(Canvas canvas) {
    // draw face
    canvas.drawCircle(mCenterX, mCenterY, mRadius, mCirclePaint);
    // draw eyes
    float eyeRadius = mRadius / 5f;
    float eyeOffsetX = mRadius / 3f;
    float eyeOffsetY = mRadius / 3f;
    canvas.drawCircle(mCenterX - eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
mEyeAndMouthPaint);
    canvas.drawCircle(mCenterX + eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
mEyeAndMouthPaint);
    // draw mouth
    float mouthInset = mRadius /3f;
    mArcBounds.set(mouthInset, mouthInset, mRadius * 2 - mouthInset, mRadius * 2 -
mouthInset);
    canvas.drawArc(mArcBounds, 45f, 90f, false, mEyeAndMouthPaint);
}
```

6. **Add your custom view to a layout:** the custom view can now be included in any layout files that you have. Here we just wrap it inside a `FrameLayout`:

```
<FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
```

```
    <com.example.app.SmileyView
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
</FrameLayout>
```

Note that it is recommended to build your project after the view code is finished. Without building it you won't be able to see the view on a preview screen in Android Studio.

After putting everything together, you should be greeted with the following screen after launching the activity containing the above layout:

Section 28.2: Adding attributes to views

Custom views can also take custom attributes which can be used in Android layout resource files. To add attributes to your custom view you need to do the following:

1. **Define the name and type of your attributes:** this is done inside `res/values/attrs.xml` (create it if necessary). The following file defines a color attribute for our smiley's face color and an enum attribute for the smiley's expression:

```
<resources>
    <declare-styleable name="SmileyView">
        <attr name="smileyColor" format="color" />
        <attr name="smileyExpression" format="enum">
            <enum name="happy" value="0"/>
            <enum name="sad" value="1"/>
        </attr>
    </declare-styleable>
    <!-- attributes for other views -->
</resources>
```

2. **Use your attributes inside your layout:** this can be done inside any layout files that use your custom view. The following layout file creates a screen with a happy yellow smiley:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    android:layout_width="match_parent">

    <com.example.app.SmileyView
        android:layout_height="56dp"
        android:layout_width="56dp"
        app:smileyColor="#ffff00"
        app:smileyExpression="happy" />
</FrameLayout>
```

Tip: Custom attributes do not work with the `tools:` prefix in Android Studio 2.1 and older (and possibly in future versions). In this example, replacing `app:smileyColor` with `tools:smileyColor` would result in `smileyColor` neither being set during runtime nor at design time.

3. **Read your attributes:** this is done inside your custom view source code. The following snippet of `SmileyView` demonstrates how the attributes can be extracted:

```
public class SmileyView extends View {
    // ...

    public SmileyView(Context context) {
        this(context, null);
    }

    public SmileyView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public SmileyView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);

        TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
defStyleAttr, 0);
        mFaceColor = a.getColor(R.styleable.SmileyView_smileyColor, Color.TRANSPARENT);
        mFaceExpression = a.getInteger(R.styleable.SmileyView_smileyExpression,
Expression.HAPPY);
        // Important: always recycle the TypedArray
        a.recycle();

        // initPaints(); ...
    }
}
```

4. **(Optional) Add default style:** this is done by adding a style with the default values and loading it inside your custom view. The following default smiley style represents a happy yellow one:

```
<!-- styles.xml -->
<style name="DefaultSmileyStyle">
    <item name="smileyColor">#ffff00</item>
    <item name="smileyExpression">happy</item>
</style>
```

Which gets applied in our `SmileyView` by adding it as the last parameter of the call to `obtainStyledAttributes` (see code in step 3):

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView, defStyleAttr,
R.style.DefaultSmileyViewStyle);
```

Note that any attribute values set in the inflated layout file (see code in step 2) will override the corresponding values of the default style.

5. **(Optional) Provide styles inside themes:** this is done by adding a new style reference attribute which can be used inside your themes and providing a style for that attribute. Here we simply name our reference attribute `smileyStyle`:

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

Which we then provide a style for in our app theme (here we just reuse the default style from step 4):

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
</style>
```

Section 28.3: CustomView performance tips

Do not allocate new objects in **onDraw**

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //Do not allocate here
}
```

Instead of drawing drawables in canvas...

```
drawable.setBounds(boundsRect);

drawable.draw(canvas);
```

Use a Bitmap for faster drawing:

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

Do not redraw the entire view to update just a small part of it. Instead redraw the specific part of view.

```
invalidate(boundToBeRefreshed);
```

If your view is doing some continuous animation, for instance a watch-face showing each and every second, at least stop the animation at `onStop()` of the activity and start it back on `onStart()` of the activity.

Do not do any calculations inside the `onDraw` method of a view, you should instead finish drawing before calling `invalidate()`. By using this technique you can avoid frame dropping in your view.

Rotations

The basic operations of a view are translate, rotate, etc... Almost every developer has faced this problem when they

use bitmap or gradients in their custom view. If the view is going to show a rotated view and the bitmap has to be rotated in that custom view, many of us will think that it will be expensive. Many think that rotating a bitmap is very expensive because in order to do that, you need to translate the bitmap's pixel matrix. But the truth is that it is not that tough! Instead of rotating the bitmap, just rotate the canvas itself!

```
// Save the canvas state
int save = canvas.save();
// Rotate the canvas by providing the  center point as pivot and angle
canvas.rotate(pivotX, pivotY, angle);
// Draw whatever you want
// Basically whatever you draw here will be drawn as per the angle you rotated the canvas
canvas.drawBitmap(...);
// Now restore your your canvas to its original state
canvas.restore(save);
// Unless canvas is restored to its original state, further draw will also be rotated.
```

Section 28.4: Creating a compound view

A **compound view** is a custom `ViewGroup` that's treated as a single view by the surrounding program code. Such a ViewGroup can be really useful in DDD-like design, because it can correspond to an aggregate, in this example, a Contact. It can be reused everywhere that contact is displayed.

This means that the surrounding controller code, an Activity, Fragment or Adapter, can simply pass the data object to the view without picking it apart into a number of different UI widgets.

This facilitates code reuse and makes for a better design according to SOLID priciples.

The layout XML

This is usually where you start. You have an existing bit of XML that you find yourself reusing, perhaps as an `<include/>`. Extract it into a separate XML file and wrap the root tag in a `<merge>` element:

```
<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <ImageView
            android:id="@+id/photo"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_alignParentRight="true" />

        <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/photo" />

        <TextView
            android:id="@+id/phone_number"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/name"
            android:layout_toLeftOf="@id/photo" />
</merge>
```

This XML file keeps working in the Layout Editor in Android Studio perfectly fine. You can treat it like any other

layout.

The compound ViewGroup

Once you have the XML file, create the custom view group.

```
import android.annotation.TargetApi;
import android.content.Context;
import android.os.Build;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.RelativeLayout;
import android.widget.ImageView;
import android.widget.TextView;

import myapp.R;

/**
 * A compound view to show contacts.
 *
 * This class can be put into an XML layout or instantiated programmatically, it
 * will work correctly either way.
 */
public class ContactView extends RelativeLayout {

    // This class extends RelativeLayout because that comes with an automatic
    // (MATCH_PARENT, MATCH_PARENT) layout for its child item. You can extend
    // the raw android.view.ViewGroup class if you want more control. See the
    // note in the layout XML why you wouldn't want to extend a complex view
    // such as RelativeLayout.

    // 1. Implement superclass constructors.
    public ContactView(Context context) {
        super(context);
        init(context, null);
    }

    // two extra constructors left out to keep the example shorter

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public ContactView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init(context, attrs);
    }

    // 2. Initialize the view by inflating an XML using `this` as parent
    private TextView mName;
    private TextView mPhoneNumber;
    private ImageView mPhoto;

    private void init(Context context, AttributeSet attrs) {
        LayoutInflater.from(context).inflate(R.layout.contact_view, this, true);
        mName = (TextView) findViewById(R.id.name);
        mPhoneNumber = (TextView) findViewById(R.id.phone_number);
        mPhoto = (ImageView) findViewById(R.id.photo);
    }

    // 3. Define a setter that's expressed in your domain model. This is what the example is
    //    all about. All controller code can just invoke this setter instead of fiddling with
    //    lots of strings, visibility options, colors, animations, etc. If you don't use a
```

```
    //    custom view, this code will usually end up in a static helper method (bad) or copies
    //    of this code will be copy-pasted all over the place (worse).
    public void setContact(Contact contact) {
        mName.setText(contact.getName());
        mPhoneNumber.setText(contact.getPhoneNumber());
        if (contact.hasPhoto()) {
            mPhoto.setVisibility(View.VISIBLE);
            mPhoto.setImageBitmap(contact.getPhoto());
        } else {
            mPhoto.setVisibility(View.GONE);
        }
    }
}
```

The `init(Context, AttributeSet)` method is where you would read any custom XML attributes as explained in Adding Attributes to Views.

With these pieces in place, you can use it in your app.

Usage in XML

Here's an example `fragment_contact_info.xml` that illustrates how you'd put a single ContactView on top of a list of messages:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">

    <!-- The compound view becomes like any other view XML element -->
    <myapp.ContactView
       android:id="@+id/contact"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"/>

    <android.support.v7.widget.RecyclerView
       android:id="@+id/message_list"
       android:layout_width="match_parent"
       android:layout_height="0dp"
       android:layout_weight="1"/>

</LinearLayout>
```

Usage in Code

Here's an example `RecyclerView.Adapter` that shows a list of contacts. This example illustrates just how much cleaner the controller code gets when it's completely free of View manipulation.

```
package myapp;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.ViewGroup;

public class ContactsAdapter extends RecyclerView.Adapter<ContactsViewHolder> {

    private final Context context;

    public ContactsAdapter(final Context context) {
        this.context = context;
```

```
    }

    @Override
    public ContactsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        ContactView v = new ContactView(context); // <--- this
        return new ContactsViewHolder(v);
    }

    @Override
    public void onBindViewHolder(ContactsViewHolder holder, int position) {
        Contact contact = this.getItem(position);
        holder.setContact(contact);  // <--- this
    }

    static class ContactsViewHolder extends RecyclerView.ViewHolder {

        public ContactsViewHolder(ContactView itemView) {
            super(itemView);
        }

        public void setContact(Contact contact) {
            ((ContactView) itemView).setContact(contact); // <--- this
        }
    }
}
```

Section 28.5: Compound view for SVG/VectorDrawable as drawableRight

Main motive to develop this compound view is, below 5.0 devices does not support svg in drawable inside TextView/EditText. One more pros is, we can set `height` and `width` of `drawableRight` inside `EditText`. I have separated it from my project and created in separate module. **Module Name : custom_edit_drawable (short name for prefix- c_d_e)**

"c_d_e_" prefix to use so that app module resources should not override them by mistake. Example : "abc" prefix is used by google in support library.

build.gradle

```
dependencies {
   compile 'com.android.support:appcompat-v7:25.3.1'
}
```

use AppCompat >= 23

Layout file : c_e_d_compound_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content">

    <EditText
       android:id="@+id/edt_search"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:inputType="text"
       android:maxLines="1"
```

```
        android:paddingEnd="40dp"
        android:paddingLeft="5dp"
        android:paddingRight="40dp"
        android:paddingStart="5dp" />

    <!--make sure you are not using ImageView instead of this-->
    <android.support.v7.widget.AppCompatImageView
        android:id="@+id/drawbleRight_search"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_gravity="right|center_vertical"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp" />
</FrameLayout>
```

Custom Attributes : attrs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="EditTextWithDrawable">
        <attr name="c_e_d_drawableRightSVG" format="reference" />
        <attr name="c_e_d_hint" format="string" />
        <attr name="c_e_d_textSize" format="dimension" />
        <attr name="c_e_d_textColor" format="color" />
    </declare-styleable>
</resources>
```

Code : EditTextWithDrawable.java

```
public class EditTextWithDrawable extends FrameLayout {
    public AppCompatImageView mDrawableRight;
    public EditText mEditText;

    public EditTextWithDrawable(Context context) {
        super(context);
        init(null);
    }

    public EditTextWithDrawable(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs);
    }

    public EditTextWithDrawable(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init(attrs);
    }

    @TargetApi(Build.VERSION_CODES.LOLLIPOP)
    public EditTextWithDrawable(Context context, AttributeSet attrs, int defStyleAttr, int
defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init(attrs);
    }

    private void init(AttributeSet attrs) {
        if (attrs != null && !isInEditMode()) {
            LayoutInflater inflater = (LayoutInflater) getContext()
                    .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            inflater.inflate(R.layout.c_e_d_compound_view, this, true);
            mDrawableRight = (AppCompatImageView) ((FrameLayout) getChildAt(0)).getChildAt(1);
```

```
            mEditText = (EditText) ((FrameLayout) getChildAt(0)).getChildAt(0);

            TypedArray attributeArray = getContext().obtainStyledAttributes(
                    attrs,
                    R.styleable.EditTextWithDrawable);

            int drawableRes =
                    attributeArray.getResourceId(
                            R.styleable.EditTextWithDrawable_c_e_d_drawableRightSVG, -1);
            if (drawableRes != -1) {
                mDrawableRight.setImageResource(drawableRes);
            }

            mEditText.setHint(attributeArray.getString(
                    R.styleable.EditTextWithDrawable_c_e_d_hint));
            mEditText.setTextColor(attributeArray.getColor(
                    R.styleable.EditTextWithDrawable_c_e_d_textColor, Color.BLACK));
            int textSize =
attributeArray.getDimensionPixelSize(R.styleable.EditTextWithDrawable_c_e_d_textSize, 15);
            mEditText.setTextSize(TypedValue.COMPLEX_UNIT_PX, textSize);
            android.view.ViewGroup.LayoutParams layoutParams = mDrawableRight.getLayoutParams();
            layoutParams.width = (textSize * 3) / 2;
            layoutParams.height = (textSize * 3) / 2;
            mDrawableRight.setLayoutParams(layoutParams);

            attributeArray.recycle();
        }
    }
}
```

Example : How to use above view

Layout : activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">

    <com.customeditdrawable.AppEditTextWithDrawable
       android:id="@+id/edt_search_emp"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       app:c_e_d_drawableRightSVG="@drawable/ic_svg_search"
       app:c_e_d_hint="@string/hint_search_here"
       app:c_e_d_textColor="@color/text_color_dark_on_light_bg"
       app:c_e_d_textSize="@dimen/text_size_small" />
</LinearLayout>
```

Activity : MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditTextWithDrawable mEditTextWithDrawable;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mEditTextWithDrawable= (EditTextWithDrawable) findViewById(R.id.edt_search_emp);
    }
}
```

Section 28.6: Responding to Touch Events

Many custom views need to accept user interaction in the form of touch events. You can get access to touch events by overriding `onTouchEvent`. There are a number of actions you can filter out. The main ones are

- `ACTION_DOWN`: This is triggered once when your finger first touches the view.
- `ACTION_MOVE`: This is called every time your finger moves a little across the view. It gets called many times.
- `ACTION_UP`: This is the last action to be called as you lift your finger off the screen.

You can add the following method to your view and then observe the log output when you touch and move your finger around your view.

```
@Override
public boolean onTouchEvent(MotionEvent event) {

    int x = (int) event.getX();
    int y = (int) event.getY();
    int action = event.getAction();

    switch (action) {
        case MotionEvent.ACTION_DOWN:
            Log.i("CustomView", "onTouchEvent: ACTION_DOWN: x = " + x + ", y = " + y);
            break;

        case MotionEvent.ACTION_MOVE:
            Log.i("CustomView", "onTouchEvent: ACTION_MOVE: x = " + x + ", y = " + y);
            break;

        case MotionEvent.ACTION_UP:
            Log.i("CustomView", "onTouchEvent: ACTION_UP: x = " + x + ", y = " + y);
            break;
    }
    return true;
}
```

Further reading:

- Android official documentation: Responding to Touch Events

Chapter 29: Getting Calculated View Dimensions

Section 29.1: Calculating initial View dimensions in an Activity

```
package com.example;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.util.Log;
import android.view.View;
import android.view.ViewTreeObserver;

public class ExampleActivity extends Activity {

    @Override
    protected void onCreate(@Nullable final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_example);

        final View viewToMeasure = findViewById(R.id.view_to_measure);

        // viewToMeasure dimensions are not known at this point.
        // viewToMeasure.getWidth() and viewToMeasure.getHeight() both return 0,
        // regardless of on-screen size.

        viewToMeasure.getViewTreeObserver().addOnPreDrawListener(new
ViewTreeObserver.OnPreDrawListener() {
            @Override
            public boolean onPreDraw() {
                // viewToMeasure is now measured and laid out, and displayed dimensions are known.
                logComputedViewDimensions(viewToMeasure.getWidth(), viewToMeasure.getHeight());

                // Remove this listener, as we have now successfully calculated the desired
dimensions.
                viewToMeasure.getViewTreeObserver().removeOnPreDrawListener(this);

                // Always return true to continue drawing.
                return true;
            }
        });
    }

    private void logComputedViewDimensions(final int width, final int height) {
        Log.d("example", "viewToMeasure has width " + width);
        Log.d("example", "viewToMeasure has height " + height);
    }

}
```

Chapter 30: Adding a FuseView to an Android Project

Export a Fuse.View from fusetools and use it inside an existing android project.

Our goal is to export the entire hikr sample app and use it inside an `Activity`.

Final work can be found @lucamtudor/hikr-fuse-view

Section 30.1: hikr app, just another android.view.View

Prerequisites

- you should have fuse installed (https://www.fusetools.com/downloads)
- you should have done the introduction tutorial
- in terminal: `fuse install android`
- in terminal: `uno install Fuse.Views`

Step 1

```
git clone https://github.com/fusetools/hikr
```

Step 2 : Add package reference to `Fuse.Views`

Find `hikr.unoproj` file inside the project root folder and add `"Fuse.Views"` to the `"Packages"` array.

```
{
  "RootNamespace":"",
  "Packages": [
    "Fuse",
    "FuseJS",
    "Fuse.Views"
  ],
  "Includes": [
    "*",
    "Modules/*.js:Bundle"
  ]
}
```

Step 3 : Make `HikrApp` component to hold the entire app

3.1 In the project root folder make a new file called `HikrApp.ux` and paste the contents of `MainView.ux`.

HikrApp.ux

```
<App Background="#022328">
    <iOS.StatusBarConfig Style="Light" />
    <Android.StatusBarConfig Color="#022328" />

    <Router ux:Name="router" />

    <ClientPanel>
        <Navigator DefaultPath="splash">
```

```
            <SplashPage ux:Template="splash" router="router" />
            <HomePage ux:Template="home" router="router" />
            <EditHikePage ux:Template="editHike" router="router" />
        </Navigator>
    </ClientPanel>
</App>
```

3.2 In `HikrApp.ux`

- replace the **`<App>`** tags with **`<Page>`**
- add `ux:Class="HikrApp"` to the opening **`<Page>`**
- remove **`<ClientPanel>`**, we don't have to worry anymore about the status bar or the bottom nav buttons

HikrApp.ux

```
<Page ux:Class="HikrApp" Background="#022328">
    <iOS.StatusBarConfig Style="Light" />
    <Android.StatusBarConfig Color="#022328" />

    <Router ux:Name="router" />

    <Navigator DefaultPath="splash">
        <SplashPage ux:Template="splash" router="router" />
        <HomePage ux:Template="home" router="router" />
        <EditHikePage ux:Template="editHike" router="router" />
    </Navigator>
</Page>
```

3.3 Use the newly created `HikrApp` component inside `MainView.ux`

Replace the content of `MainView.ux` file with:

```
<App>
    <HikrApp/>
</App>
```

Our app is back to its normal behavior, but we now have extracted it to a separate component called `HikrApp`

Step 4 Inside `MainView.ux` replace the **`<App>`** tags with **`<ExportedViews>`** and add `ux:Template="HikrAppView"` to **`<HikrApp />`**

```
<ExportedViews>
    <HikrApp ux:Template="HikrAppView" />
</ExportedViews>
```

Remember the template `HikrAppView`, because we'll need it to get a reference to our view from Java.

Note. From the fuse docs:

> `ExportedViews` will behave as `App` when doing normal `fuse preview` and `uno build`

Not true. You will get this error when previewing from Fuse Studio:

> Error: Couldn't find an App tag in any of the included UX files. Have you forgot to include the UX file that contains the app tag?

Step 5 Wrap `SplashPage.ux`'s `<DockPanel>` in a `<GraphicsView>`

```
<Page ux:Class="SplashPage">
    <Router ux:Dependency="router" />

    <JavaScript File="SplashPage.js" />

    <GraphicsView>
        <DockPanel ClipToBounds="true">
            <Video Layer="Background" File="../Assets/nature.mp4" IsLooping="true" AutoPlay="true"
StretchMode="UniformToFill" Opacity="0.5">
            <Blur Radius="4.75" />
        </Video>

            <hikr.Text Dock="Bottom" Margin="10" Opacity=".5" TextAlignment="Center"
FontSize="12">original video by Graham Uhelski</hikr.Text>

            <Grid RowCount="2">
                <StackPanel Alignment="VerticalCenter">
                    <hikr.Text Alignment="HorizontalCenter" FontSize="70">hikr</hikr.Text>
                    <hikr.Text Alignment="HorizontalCenter" Opacity=".5">get out there</hikr.Text>
                </StackPanel>

                <hikr.Button Text="Get Started" FontSize="18" Margin="50,0"
Alignment="VerticalCenter" Clicked="{goToHomePage}" />
            </Grid>
        </DockPanel>
    </GraphicsView>
</Page>
```

Step 6 Export the fuse project as an aar library

- in terminal, in root project folder: `uno clean`
- in terminal, in root project folder: `uno build -t=android -DLIBRARY`

Step 7 Prepare your android project

- copy the aar from `.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar` to `.../androidRootProject/app/libs`
- add `flatDir { dirs 'libs' }` to the root `build.gradle` file

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript { ... }

...

allprojects {
    repositories {
        jcenter()
```

```
        flatDir {
            dirs 'libs'
        }
    }
}

...
```

- add `compile(name: 'app-debug', ext: 'aar')` to dependencies in `app/build.gradle`

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.shiftstudio.fuseviewtest"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile(name: 'app-debug', ext: 'aar')
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.3.1'
    testCompile 'junit:junit:4.12'
}
```

- add the following properties to the activity inside `AndroidManifest.xml`

```
android:launchMode="singleTask"
android:taskAffinity=""
android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize"
```

Your `AndroidManifest.xml` will look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.shiftstudio.fuseviewtest">

    <application
       android:allowBackup="true"
       android:icon="@mipmap/ic_launcher"
       android:label="@string/app_name"
       android:supportsRtl="true"
       android:theme="@style/AppTheme">
        <activity
           android:name=".MainActivity"
```

```
            android:launchMode="singleTask"
            android:taskAffinity=""
            android:configChanges="orientation|keyboardHidden|screenSize|smallestScreenSize">
             <intent-filter>
                 <action android:name="android.intent.action.MAIN" />

                 <category android:name="android.intent.category.LAUNCHER" />
             </intent-filter>
         </activity>
     </application>

</manifest>
```

Step 8: Show the `Fuse.View HikrAppView` in your `Activity`

- note that your `Activity` needs to inherit `FuseViewsActivity`

```
public class MainActivity extends FuseViewsActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final ViewHandle fuseHandle = ExportedViews.instantiate("HikrAppView");

        final FrameLayout root = (FrameLayout) findViewById(R.id.fuse_root);
        final View fuseApp = fuseHandle.getView();
        root.addView(fuseApp);
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.shiftstudio.fuseviewtest.MainActivity">

    <TextView
       android:layout_width="wrap_content"
       android:layout_gravity="center_horizontal"
       android:textSize="24sp"
       android:textStyle="bold"
       android:layout_height="wrap_content"
       android:text="Hello World, from Kotlin" />

    <FrameLayout
       android:id="@+id/fuse_root"
       android:layout_width="match_parent"
       android:layout_height="match_parent">
```

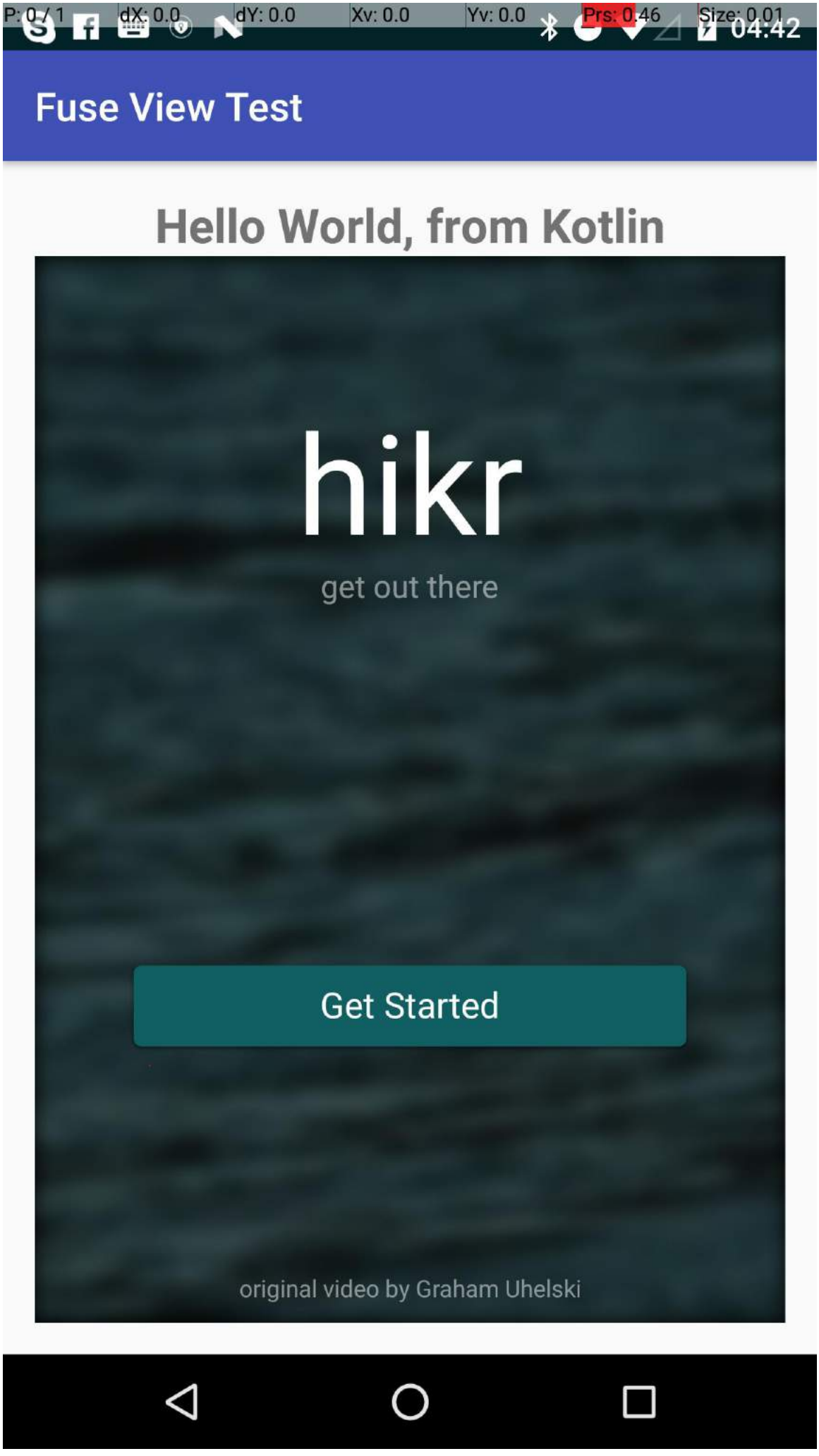
```
        <TextView
           android:layout_width="wrap_content"
           android:text="THIS IS FROM NATIVE.\nBEHIND FUSE VIEW"
           android:layout_gravity="center"
           android:textStyle="bold"
           android:textSize="30sp"
           android:background="@color/colorAccent"
           android:textAlignment="center"
           android:layout_height="wrap_content" />

    </FrameLayout>

</LinearLayout>
```

Note

When you press the back button, on android, the app crashes. You can follow the issue on the fuse forum.

```
A/libc: Fatal signal 11 (SIGSEGV), code 1, fault addr 0xdeadcab1 in tid 18026 (io.fuseviewtest)

        [ 05-25 11:52:33.658 16567:16567 W/ ]

        debuggerd: handling request: pid=18026 uid=10236 gid=10236 tid=18026
```

And the final result is something like this. You can also find a short clip on github.



Chapter 31: Supporting Screens With Different Resolutions, Sizes

Section 31.1: Using configuration qualifiers

Android supports several configuration qualifiers that allow you to control how the system selects your alternative resources based on the characteristics of the current device screen. A configuration qualifier is a string that you can append to a resource directory in your Android project and specifies the configuration for which the resources inside are designed.

To use a configuration qualifier:

1. Create a new directory in your project's res/ directory and name it using the format: `<resources_name>-<qualifier>`. `<resources_name>` is the standard resource name (such as drawable or layout).
2. `<qualifier>` is a configuration qualifier, specifying the screen configuration for which these resources are to be used (such as hdpi or xlarge).

For example, the following application resource directories provide different layout designs for different screen sizes and different drawables. Use the `mipmap/` folders for launcher icons.

```
res/layout/my_layout.xml               // layout for normal screen size ("default")
res/layout-large/my_layout.xml         // layout for large screen size
res/layout-xlarge/my_layout.xml        // layout for extra-large screen size
res/layout-xlarge-land/my_layout.xml   // layout for extra-large in landscape orientation

res/drawable-mdpi/graphic.png          // bitmap for medium-density
res/drawable-hdpi/graphic.png          // bitmap for high-density
res/drawable-xhdpi/graphic.png         // bitmap for extra-high-density
res/drawable-xxhdpi/graphic.png        // bitmap for extra-extra-high-density

res/mipmap-mdpi/my_icon.png          // launcher icon for medium-density
res/mipmap-hdpi/my_icon.png          // launcher icon for high-density
res/mipmap-xhdpi/my_icon.png         // launcher icon for extra-high-density
res/mipmap-xxhdpi/my_icon.png        // launcher icon for extra-extra-high-density
res/mipmap-xxxhdpi/my_icon.png       // launcher icon for extra-extra-extra-high-density
```

Section 31.2: Converting dp and sp to pixels

When you need to set a pixel value for something like `Paint.setTextSize` but still want it be scaled based on the device, you can convert dp and sp values.

```
DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, 12f, metrics);

DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 12f, metrics);
```

Alternatively, you can convert a dimension resource to pixels if you have a context to load the resource from.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="size_in_sp">12sp</dimen>
    <dimen name="size_in_dp">12dp</dimen>
</resources>
```

```
// Get the exact dimension specified by the resource
float pixels = context.getResources().getDimension(R.dimen.size_in_sp);
float pixels = context.getResources().getDimension(R.dimen.size_in_dp);

// Get the dimension specified by the resource for use as a size.
// The value is rounded down to the nearest integer but is at least 1px.
int pixels = context.getResources().getDimensionPixelSize(R.dimen.size_in_sp);
int pixels = context.getResources().getDimensionPixelSize(R.dimen.size_in_dp);

// Get the dimension specified by the resource for use as an offset.
// The value is rounded down to the nearest integer and can be 0px.
int pixels = context.getResources().getDimensionPixelOffset(R.dimen.size_in_sp);
int pixels = context.getResources().getDimensionPixelOffset(R.dimen.size_in_dp);
```

Section 31.3: Text size and different android screen sizes

Sometimes, it's better to have only three options

```
style="@android:style/TextAppearance.Small"
style="@android:style/TextAppearance.Medium"
style="@android:style/TextAppearance.Large"
```

Use small and large to differentiate from normal screen size.

```
<TextView
       android:id="@+id/TextViewTopBarTitle"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       style="@android:style/TextAppearance.Small"/>
```

For normal, you don't have to specify anything.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

Using this, you can avoid testing and specifying dimensions for different screen sizes.

Chapter 32: ViewFlipper

A `ViewFlipper` is a `ViewAnimator` that switches between two or more views that have been added to it. Only one child is shown at a time. If requested, the `ViewFlipper` can automatically flip between each child at a regular interval.

Section 32.1: ViewFlipper with image sliding

XML file:

```
<ViewFlipper
   android:id="@+id/viewflip"
   android:layout_width="match_parent"
   android:layout_height="250dp"
   android:layout_weight="1"
   />
```

Java code:

```
public class BlankFragment extends Fragment{
    ViewFlipper viewFlipper;
    FragmentManager fragmentManager;
    int gallery_grid_Images[] = {drawable.image1, drawable.image2, drawable.image3,
            drawable.image1, drawable.image2, drawable.image3, drawable.image1,
            drawable.image2, drawable.image3, drawable.image1
    };

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState){
        View rootView = inflater.inflate(fragment_blank, container, false);
        viewFlipper = (ViewFlipper)rootView.findViewById(R.id.viewflip);
        for(int i=0; i<gallery_grid_Images.length; i++){
            // This will create dynamic image views and add them to the ViewFlipper.
            setFlipperImage(gallery_grid_Images[i]);
        }
        return rootView;
    }

    private void setFlipperImage(int res) {
        Log.i("Set Filpper Called", res+"");
        ImageView image = new ImageView(getContext());
        image.setBackgroundResource(res);
        viewFlipper.addView(image);
        viewFlipper.setFlipInterval(1000);
        viewFlipper.setAutoStart(true);
    }
}
```

Chapter 33: Design Patterns

Design patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.

Design patterns can speed up the development process by providing tested, proven development paradigms.

Reusing design patterns helps to prevent subtle issues that can cause major problems, and it also improves code readability for coders and architects who are familiar with the patterns.

Section 33.1: Observer pattern

The observer pattern is a common pattern, which is widely used in many contexts. A real example can be taken from YouTube: When you like a channel and want to get all news and watch new videos from this channel, you have to subscribe to that channel. Then, whenever this channel publishes any news, you (and all other subscribers) will receive a notification.

An observer will have two components. One is a broadcaster (channel) and the other is a receiver (you or any other subscriber). The broadcaster will handle all receiver instances that subscribed to it. When the broadcaster fires a new event, it will announce this to all receiver instances. When the receiver receives an event, it will have to react to that event, for example, by turning on YouTube and playing the new video.

Implementing the observer pattern

1. The broadcaster has to provide methods that permit receivers to subscribe and unsubscribe to it. When the broadcaster fires an event, subscribers need to be notified that an event has occurred:

```
class Channel{
    private List<Subscriber> subscribers;
    public void subscribe(Subscriber sub) {
        // Add new subscriber.
    }
    public void unsubscribe(Subscriber sub) {
        // Remove subscriber.
    }
    public void newEvent() {
        // Notification event for all subscribers.
    }
}
```

2. The receiver needs to implement a method that handles the event from the broadcaster:

```
interface Subscriber {
    void doSubscribe(Channel channel);
    void doUnsubscribe(Channel channel);
    void handleEvent();  // Process the new event.
}
```

Section 33.2: Singleton Class Example

Java Singleton Pattern

To implement Singleton pattern, we have different approaches but all of them have following common concepts.

- Private constructor to restrict instantiation of the class from other classes.
- Private static variable of the same class that is the only instance of the class.
- Public static method that returns the instance of the class, this is the global access
- point for outer world to get the instance of the singleton class.

```
/**
 * Singleton class.
 */
public final class Singleton {

  /**
   * Private constructor so nobody can instantiate the class.
   */
  private Singleton() {}

  /**
   * Static to class instance of the class.
   */
  private static final Singleton INSTANCE = new Singleton();

  /**
   * To be called by user to obtain instance of the class.
   *
   * @return instance of the singleton.
   */
  public static Singleton getInstance() {
    return INSTANCE;
  }
}
```

Chapter 34: Activity

Parameter	Details
Intent	Can be used with startActivity to launch an Activity
Bundle	A mapping from String keys to various Parcelable values.
Context	Interface to global information about an application environment.

An Activity represents a single screen with a user **interface(UI)**. An Android App may have more than one Activity, for example, An email App can have one activity to list all the emails, another activity to show email contents, yet another activity to compose new email. All the activities in an App work together to create perfect user experience.

Section 34.1: Activity launchMode

Launch mode defines the behaviour of new or existing activity in the task.
There are possible launch modes:

- standard
- singleTop
- singleTask
- singleInstance

It should be defined in android manifest in `<activity/>` element as `android:launchMode` attribute.

```
<activity
   android:launchMode=["standard" | "singleTop" | "singleTask" | "singleInstance"] />
```

Standard:

Default value. If this mode set, new activity will always be created for each new intent. So it's possible to get many activities of same type. New activity will be placed on the top of the task. There is some difference for different android version: if activity is starting from another application, on androids <= 4.4 it will be placed on same task as starter application, but on >= 5.0 new task will be created.

SingleTop:

This mode is almost the same as `standard`. Many instances of singleTop activity could be created. The difference is, if an instance of activity already exists on the top of the current stack, `onNewIntent()` will be called instead of creating new instance.

SingleTask:

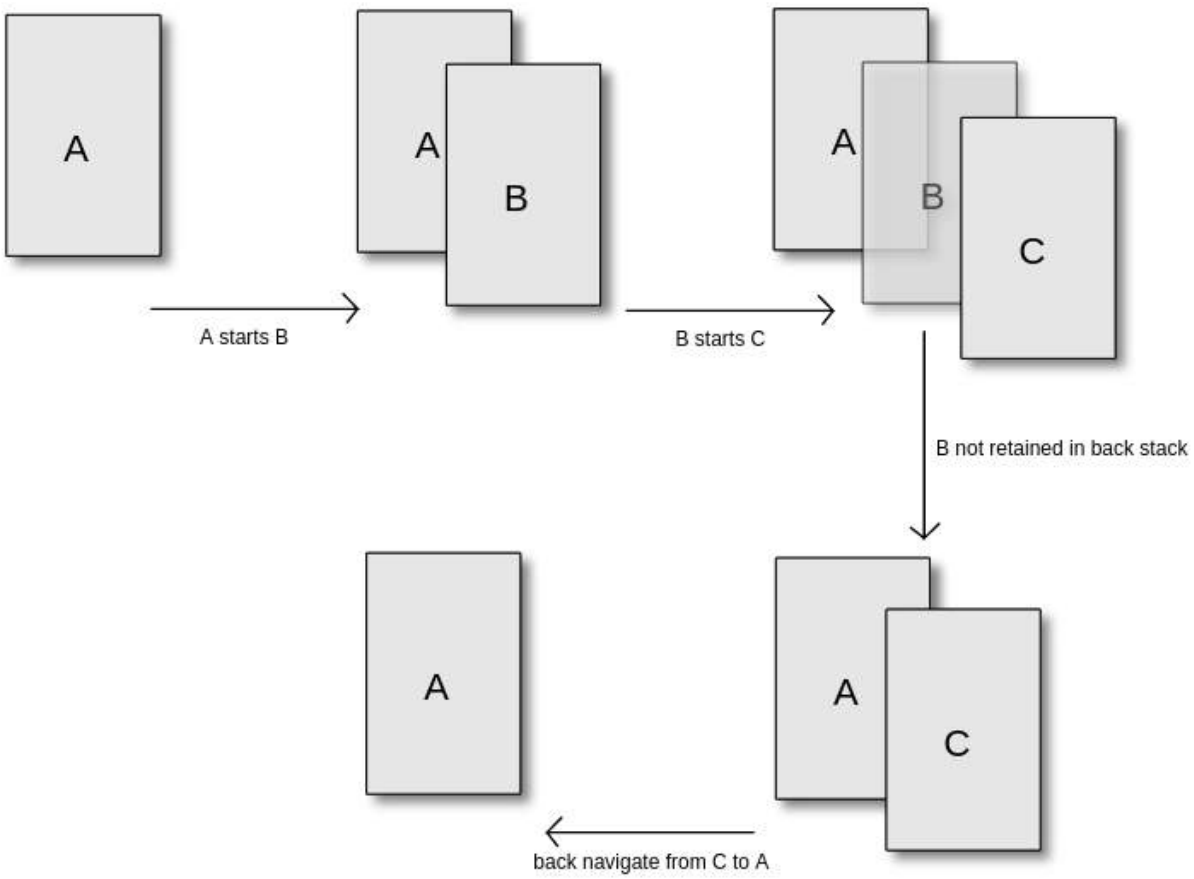
Activity with this launch mode can have only one instance **in the system**. New task for activity will be created, if it doesn't exist. Otherwise, task with activity will be moved to front and `onNewIntent` will be called.

SingleInstance:

This mode is similar to `singleTask`. The difference is task that holds an activity with `singleInstance` could have only this activity and nothing more. When `singleInstance` activity create another activity, new task will be created to place that activity.

Section 34.2: Exclude an activity from back-stack history

Let there be Activity `B` that can be opened, and can further start more Activities. But, user should not encounter it when navigating back in task activities.



The simplest solution is to set the attribute `noHistory` to **`true`** for that **`<activity>`** tag in `AndroidManifest.xml`:

```
<activity
    android:name=".B"
    android:noHistory="true">
```

This same behavior is also possible from code if `B` calls `finish()` before starting the next activity:

```
finish();
startActivity(new Intent(context, C.class));
```

Typical usage of `noHistory` flag is with "Splash Screen" or Login Activities.

Section 34.3: Android Activity LifeCycle Explained

Assume an application with a MainActivity which can call the Next Activity using a button click.

```
public class MainActivity extends AppCompatActivity {

    private final String LOG_TAG = MainActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(LOG_TAG, "calling onCreate from MainActivity");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d(LOG_TAG, "calling onStart from MainActivity");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d(LOG_TAG, "calling onResume  from MainActivity");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d(LOG_TAG, "calling onPause  from MainActivity");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d(LOG_TAG, "calling onStop  from MainActivity");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(LOG_TAG, "calling onDestroy  from MainActivity");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d(LOG_TAG, "calling onRestart  from MainActivity");
    }
    public void toNextActivity(){
        Log.d(LOG_TAG, "calling Next Activity");
        Intent intent = new Intent(this, NextActivity.class);
        startActivity(intent);
    } }
```

and

```
public class NextActivity extends AppCompatActivity {
    private final String LOG_TAG = NextActivity.class.getSimpleName();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_next);
        Log.d(LOG_TAG, "calling onCreate from Next Activity");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d(LOG_TAG, "calling onStart from Next Activity");
    }
    @Override
    protected void onResume() {
```

```
        super.onResume();
        Log.d(LOG_TAG, "calling onResume  from Next Activity");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d(LOG_TAG, "calling onPause  from Next Activity");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d(LOG_TAG, "calling onStop  from Next Activity");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(LOG_TAG, "calling onDestroy  from Next Activity");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d(LOG_TAG, "calling onRestart  from Next Activity");
    } }
```

When app is first created
D/MainActivity: calling onCreate from MainActivity
D/MainActivity: calling onStart from MainActivity
D/MainActivity: calling onResume from MainActivity
are called

When screen sleeps
08:11:03.142 D/MainActivity: calling onPause from MainActivity
08:11:03.192 D/MainActivity: calling onStop from MainActivity
are called. And again when it wakes up
08:11:55.922 D/MainActivity: calling onRestart from MainActivity
08:11:55.962 D/MainActivity: calling onStart from MainActivity
08:11:55.962 D/MainActivity: calling onResume from MainActivity
are called

Case1: When Next Activity is called from Main Activity
D/MainActivity: calling Next Activity
D/MainActivity: calling onPause from MainActivity
D/NextActivity: calling onCreate from Next Activity
D/NextActivity: calling onStart from Next Activity
D/NextActivity: calling onResume from Next Activity
D/MainActivity: calling onStop from MainActivity

When Returning back to the Main Activity from Next Activity using back button
D/NextActivity: calling onPause from Next Activity
D/MainActivity: calling onRestart from MainActivity
D/MainActivity: calling onStart from MainActivity
D/MainActivity: calling onResume from MainActivity

D/NextActivity: calling onStop from Next Activity
D/NextActivity: calling onDestroy from Next Activity

Case2: When Activity is partially obscured (When overview button is pressed) or When app goes to background and another app completely obscures it
D/MainActivity: calling onPause from MainActivity
D/MainActivity: calling onStop from MainActivity
and when the app is back in the foreground ready to accept User inputs,
D/MainActivity: calling onRestart from MainActivity
D/MainActivity: calling onStart from MainActivity
D/MainActivity: calling onResume from MainActivity
are called

Case3: When an activity is called to fulfill implicit intent and user has make a selection. For eg., when share button is pressed and user has to select an app from the list of applications shown
D/MainActivity: calling onPause from MainActivity

The activity is visible but not active now. When the selection is done and app is active
D/MainActivity: calling onResume from MainActivity
is called

Case4:
When the app is killed in the background(to free resources for another foreground app), *onPause*(for pre-honeycomb device) or *onStop*(for since honeycomb device) will be the last to be called before the app is terminated.

onCreate and onDestroy will be called utmost once each time the application is run. But the onPause, onStop, onRestart, onStart, onResume maybe called many times during the lifecycle.

Section 34.4: End Application with exclude from Recents

First define an ExitActivity in the AndroidManifest.xml

```
<activity
       android:name="com.your_example_app.activities.ExitActivity"
       android:autoRemoveFromRecents="true"
       android:theme="@android:style/Theme.NoDisplay" />
```

Afterwards the ExitActivity-class

```
/**
 * Activity to exit Application without staying in the stack of last opened applications
 */
public class ExitActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (Utils.hasLollipop()) {
            finishAndRemoveTask();
        } else if (Utils.hasJellyBean()) {
            finishAffinity();
        } else {
            finish();
        }
    }
```

```
    /**
     * Exit Application and Exclude from Recents
     *
     * @param context Context to use
     */
    public static void exitApplication(ApplicationContext context) {
        Intent intent = new Intent(context, ExitActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NO_ANIMATION | Intent.FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS);
        context.startActivity(intent);
    }
}
```

Section 34.5: Presenting UI with setContentView

Activity class takes care of creating a window for you in which you can place your UI with `setContentView`.
There are three `setContentView` methods:

- `setContentView(int layoutResID)` - Set the activity content from a layout resource.
- `setContentView(View view)` - Set the activity content to an explicit view.
- `setContentView(View view, ViewGroup.LayoutParams params)` - Set the activity content to an explicit view with provided params.

When `setContentView` is called, this view is placed directly into the activity's view hierarchy. It can itself be a complex view hierarchy.

Examples

Set content from resource file:

Add resource file (main.xml in this example) with view hierarchy:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >

    <TextView android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Hello" />

</FrameLayout>
```

Set it as content in activity:

```
public final class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // The resource will be inflated,
        // adding all top-level views to the activity.
        setContentView(R.layout.main);
    }
}
```

Set content to an explicit view:

```
public final class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Creating view with container
        final FrameLayout root = new FrameLayout(this);
        final TextView text = new TextView(this);
        text.setText("Hello");
        root.addView(text);

        // Set container as content view
        setContentView(root);
    }
}
```

Section 34.6: Up Navigation for Activities

Up navigation is done in android by adding `android:parentActivityName=""` in Manifest.xml to the activity tag. Basically with this tag you tell the system about the parent activity of a activity.

How is it done?

```
<uses-permission android:name="android.permission.INTERNET" />

<application
   android:name=".SkillSchoolApplication"
   android:allowBackup="true"
   android:icon="@mipmap/ic_launcher"
   android:label="@string/app_name"
   android:supportsRtl="true"
   android:theme="@style/AppTheme">
    <activity
       android:name=".ui.activities.SplashActivity"
       android:theme="@style/SplashTheme">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".ui.activities.MainActivity" />
    <activity android:name=".ui.activities.HomeActivity"
    android:parentActivityName=".ui.activities.MainActivity/> // HERE I JUST TOLD THE SYSTEM THAT
MainActivity is the parent of HomeActivity
</application>
```

Now when I will click on the arrow inside the toolbar of HomeActivity it will take me back to the parent activity.

Java Code

Here I will write the appropriate Java code required for this functionality.

```
public class HomeActivity extends AppCompatActivity {
    @BindView(R.id.toolbar)
    Toolbar toolbar;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        ButterKnife.bind(this);
        //Since i am using custom tool bar i am setting refernce of that toolbar to Actionbar. If you
are not using custom then you can simple leave this and move to next line
        setSupportActionBar(toolbar);
        getSupportActionBar.setDisplayHomeAsUpEnabled(true); // this will show the back arrow in
the tool bar.
}
}
```

If you run this code you will see when you press back button it will take you back to MainActivity. For futher understanding of Up Navigation i would recommend reading docs

You can more customize this behaviour upon your needs by overriding

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    // Respond to the action bar's Up/Home button
    case android.R.id.home:
        NavUtils.navigateUpFromSameTask(this); // Here you will write your logic for handling up
navigation
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Simple Hack

This is simple hack which is mostly used to navigate to parent activity if parent is in backstack. By calling `onBackPressed()` if id is equal to `android.R.id.home`

```
@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        switch (id) {
            case android.R.id.home:
                onBackPressed();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

Section 34.7: Clear your current Activity stack and launch a new Activity

If you want to clear your current Activity stack and launch a new Activity (for example, logging out of the app and launching a log in Activity), there appears to be two approaches.

1. Target (API >= 16)

Calling `finishAffinity()` from an Activity

2. Target (11 <= API < 16)

```
Intent intent = new Intent(this, LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK
|Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
finish();
```

Chapter 35: Activity Recognition

Activity recognition is the detection of a user's physical activity in order to perform certain actions on the device, such as taking points when a drive is detected, turn wifi off when a phone is still, or putting the ring volume to max when the user is walking.

Section 35.1: Google Play ActivityRecognitionAPI

This is a just a simple example of how to use GooglePlay Service's ActivityRecognitionApi. Although this is a great library, it does not work on devices that do not have Google Play Services installed.

Docs for ActivityRecognition API

Manifest

```
<!-- This is needed to use Activity Recognition! -->
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />

<application
   android:allowBackup="true"
   android:icon="@mipmap/ic_launcher"
   android:label="@string/app_name"
   android:roundIcon="@mipmap/ic_launcher_round"
   android:supportsRtl="true"
   android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ActivityReceiver" />
</application>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient apiClient;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        apiClient = new GoogleApiClient.Builder(this)
                .addApi(ActivityRecognition.API)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .build();

        //This just gets the activity intent from the ActivityReceiver class
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
```

```
        localActivityReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                ActivityRecognitionResult recognitionResult =
ActivityRecognitionResult.extractResult(intent);
                TextView textView = (TextView) findViewById(R.id.activityText);

                //This is just to get the activity name. Use at your own risk.

textView.setText(DetectedActivity.zzkf(recognitionResult.getMostProbableActivity().getType()));
            }
        };
    }

    @Override
    protected void onResume() {
        super.onResume();

        //Register local broadcast receiver
        localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

        //Connect google api client
        apiClient.connect();
    }

    @Override
    protected void onPause() {
        super.onPause();

        //Unregister for activity recognition
        ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates(apiClient,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT));

        //Disconnects api client
        apiClient.disconnect();

        //Unregister local receiver
        localBroadcastManager.unregisterReceiver(localActivityReceiver);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        //Only register for activity recognition if google api client has connected
        ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates(apiClient, 0,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT));
    }

    @Override
    public void onConnectionSuspended(int i) {
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }
}
```

ActivityReceiver

```
public class ActivityReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        LocalBroadcastManager.getInstance(context).sendBroadcast(intent.setAction("activity"));
    }
}
```

Section 35.2: PathSense Activity Recognition

PathSense activity recognition is another good library for devices which don't have Google Play Services, as they have built their own activity recognition model, but requires developers register at http://developer.pathsense.com to get an API key and Client ID.

Manifest

```
<application
   android:allowBackup="true"
   android:icon="@mipmap/ic_launcher"
   android:label="@string/app_name"
   android:roundIcon="@mipmap/ic_launcher_round"
   android:supportsRtl="true"
   android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <receiver android:name=".ActivityReceiver" />

    <!-- You need to acquire these from their website (http://developer.pathsense.com) -->
    <meta-data
       android:name="com.pathsense.android.sdk.CLIENT_ID"
       android:value="YOUR_CLIENT_ID" />
    <meta-data
       android:name="com.pathsense.android.sdk.API_KEY"
       android:value="YOUR_API_KEY" />
</application>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private PathsenseLocationProviderApi pathsenseLocationProviderApi;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        pathsenseLocationProviderApi = PathsenseLocationProviderApi.getInstance(this);

        //This just gets the activity intent from the ActivityReceiver class
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
```

```
        localActivityReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                //The detectedActivities object is passed as a serializable
                PathsenseDetectedActivities detectedActivities = (PathsenseDetectedActivities)
intent.getSerializableExtra("ps");
                TextView textView = (TextView) findViewById(R.id.activityText);

textView.setText(detectedActivities.getMostProbableActivity().getDetectedActivity().name());
            }
        };
    }

    @Override
    protected void onResume() {
        super.onResume();

        //Register local broadcast receiver
        localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

        //This gives an update every time it receives one, even if it was the same as the last update
        pathsenseLocationProviderApi.requestActivityUpdates(ActivityReceiver.class);

//        This gives updates only when it changes (ON_FOOT -> IN_VEHICLE for example)
//        pathsenseLocationProviderApi.requestActivityChanges(ActivityReceiver.class);
    }

    @Override
    protected void onPause() {
        super.onPause();

        pathsenseLocationProviderApi.removeActivityUpdates();

//        pathsenseLocationProviderApi.removeActivityChanges();

        //Unregister local receiver
        localBroadcastManager.unregisterReceiver(localActivityReceiver);
    }
}
```

ActivityReceiver.java

```
// You don't have to use their broadcastreceiver, but it's best to do so, and just pass the result
// as needed to another class.
public class ActivityReceiver extends PathsenseActivityRecognitionReceiver {

    @Override
    protected void onDetectedActivities(Context context, PathsenseDetectedActivities
pathsenseDetectedActivities) {
        Intent intent = new Intent("activity").putExtra("ps", pathsenseDetectedActivities);
        LocalBroadcastManager.getInstance(context).sendBroadcast(intent);
    }
}
```

Chapter 36: Split Screen / Multi-Screen Activities

Section 36.1: Split Screen introduced in Android Nougat implemented

Set this attribute in your manifest's or element to enable or disable multi-window display:

```
android:resizeableActivity=["true" | "false"]
```

If this attribute is set to true, the activity can be launched in split-screen and freeform modes. If the attribute is set to false, the activity does not support multi-window mode. If this value is false, and the user attempts to launch the activity in multi-window mode, the activity takes over the full screen.

If your app targets API level 24, but you do not specify a value for this attribute, the attribute's value defaults to true.

The following code shows how to specify an activity's default size and location, and its minimum size, when the activity is displayed in freeform mode:

```
<--These are default values suggested by google.-->
<activity android:name=".MyActivity">
<layout android:defaultHeight="500dp"
     android:defaultWidth="600dp"
     android:gravity="top|end"
     android:minHeight="450dp"
     android:minWidth="300dp" />
</activity>
```

Disabled features in multi-window mode

Certain features are disabled or ignored when a device is in multi-window mode, because they don't make sense for an activity which may be sharing the device screen with other activities or apps. Such features include:

1. Some System UI customization options are disabled; for example, apps cannot hide the status bar if they are not running in full-screen mode.
2. The system ignores changes to the **android:screenOrientation** attribute.

If your app targets API level 23 or lower

If your app targets API level 23 or lower and the user attempts to use the app in multi-window mode, the system forcibly resizes the app unless the app declares a fixed orientation.

If your app does not declare a fixed orientation, you should launch your app on a device running Android 7.0 or higher and attempt to put the app in split-screen mode. Verify that the user experience is acceptable when the app is forcibly resized.

If the app declares a fixed orientation, you should attempt to put the app in multi-window mode. Verify that when you do so, the app remains in full-screen mode.

Chapter 37: Material Design

Material Design is a comprehensive guide for visual, motion, and interaction design across platforms and devices.

Section 37.1: Adding a Toolbar

A `Toolbar` is a generalization of `ActionBar` for use within application layouts. While an `ActionBar` is traditionally part of an `Activity's` opaque window decor controlled by the framework, a `Toolbar` may be placed at any arbitrary level of nesting within a view hierarchy. It can be added by performing the following steps:

1. Make sure the following dependency is added to your module's (e.g. app's) **build.gradle** file under dependencies:

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

2. Set the theme for your app to one that does **not** have an `ActionBar`. To do that, edit your **styles.xml** file under `res/values`, and set a `Theme.AppCompat` theme.
   In this example we are using `Theme.AppCompat.NoActionBar` as parent of your `AppTheme`:

```
<style name="AppTheme" parent="Theme.AppCompat.NoActionBar">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primaryDark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

   You can also use `Theme.AppCompat.Light.NoActionBar` or `Theme.AppCompat.DayNight.NoActionBar`, or any other theme that does not inherently have an `ActionBar`

3. Add the `Toolbar` to your activity layout:

```
<android.support.v7.widget.Toolbar
   android:id="@+id/toolbar"
   android:layout_width="match_parent"
   android:layout_height="?attr/actionBarSize"
   android:background="?attr/colorPrimary"
   android:elevation="4dp"/>
```

   Below the `Toolbar` you can add the rest of your layout.

4. In your `Activity`, set the `Toolbar` as the `ActionBar` for this `Activity`. Provided that you're using the *appcompat* library and an `AppCompatActivity`, you would use the `setSupportActionBar()` method:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    final Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //...
```

```
}
```

After performing the above steps, you can use the `getSupportActionBar()` method to manipulate the `Toolbar` that is set as the `ActionBar`.

For example, you can set the title as shown below:

```
getSupportActionBar().setTitle("Activity Title");
```

For example, you can also set title and background color as shown below:

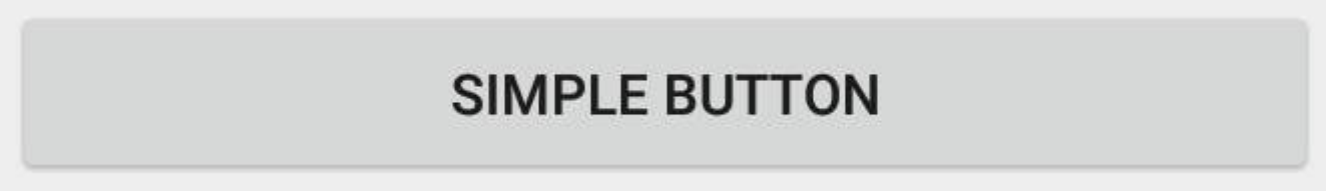
```
CharSequence title = "Your App Name";
SpannableString s = new SpannableString(title);
s.setSpan(new ForegroundColorSpan(Color.RED), 0, title.length(),
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
getSupportActionBar().setTitle(s);
getSupportActionBar().setBackgroundDrawable(new ColorDrawable(Color.argb(128, 0, 0, 0)));
```

Section 37.2: Buttons styled with Material Design

The AppCompat Support Library defines several useful styles for Buttons, each of which extend a base `Widget.AppCompat.Button` style that is applied to all buttons by default if you are using an `AppCompat` theme. This style helps ensure that all buttons look the same by default following the Material Design specification.

In this case the accent color is pink.

1. Simple Button: `@style/Widget.AppCompat.Button`




```
<Button
   style="@style/Widget.AppCompat.Button"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_margin="16dp"
   android:text="@string/simple_button"/>
```

2. Colored Button: `@style/Widget.AppCompat.Button.Colored`
   The `Widget.AppCompat.Button.Colored` style extends the `Widget.AppCompat.Button` style and applies automatically the **accent color** you selected in your app theme.


```
<Button
   style="@style/Widget.AppCompat.Button.Colored"
   android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/colored_button"/>
```

If you want to customize the background color without changing the accent color in your *main theme* you can create a *custom theme* (extending the `ThemeOverlay` theme) for your `Button` and assign it to the button's `android:theme` attribute:

```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:theme="@style/MyButtonTheme"/>
```

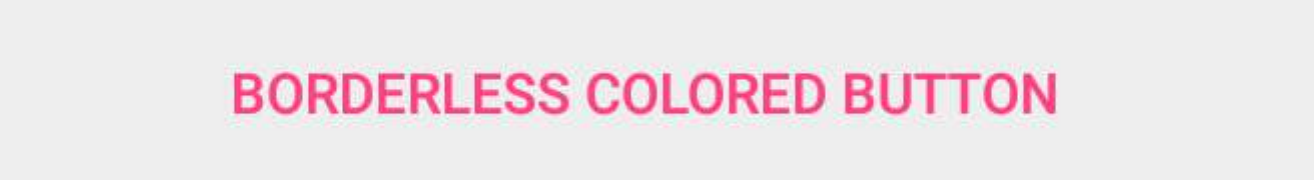
Define the theme in `res/values/themes.xml`:

```
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
     <item name="colorAccent">@color/my_color</item>
</style>
```

3. Borderless Button: `@style/Widget.AppCompat.Button.Borderless`


```
<Button
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/borderless_button"/>
```

4. Borderless Colored Button: `@style/Widget.AppCompat.Button.Borderless.Colored`




```
<Button
    style="@style/Widget.AppCompat.Button.Borderless.Colored"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/borderless_colored_button"/>
```

Section 37.3: Adding a FloatingActionButton (FAB)

In the material design, a Floating action button represents the primary action in an Activity.
They are distinguished by a circled icon floating above the UI and have motion behaviors that include morphing,

launching, and a transferring anchor point.

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Now add the `FloatingActionButton` to your layout file:

```
<android.support.design.widget.FloatingActionButton
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_margin="16dp"
     android:src="@drawable/some_icon"/>
```

where the `src` attribute references the icon that should be used for the floating action.

The result should look something like this (presuming your accent color is Material Pink):

By default, the background color of your `FloatingActionButton` will be set to your theme's accent color. Also, note that a `FloatingActionButton` requires a margin around it to work properly. The recommended margin for the bottom is `16dp` for phones and `24dp` for tablets.

Here are properties which you can use to customize the `FloatingActionButton` further (assuming `xmlns:app="http://schemas.android.com/apk/res-auto` is declared as namespace the top of your layout):

- `app:fabSize`: Can be set to `normal` or `mini` to switch between a normal sized or a smaller version.
- `app:rippleColor`: Sets the color of the ripple effect of your `FloatingActionButton`. Can be a color resource or hex string.
- `app:elevation`: Can be a string, integer, boolean, color value, floating point, dimension value.
- `app:useCompatPadding`: Enable compat padding. Maybe a boolean value, such as **`true`** or **`false`**. Set to **`true`** to use compat padding on api-21 and later, in order to maintain a consistent look with older api levels.

You can find more examples about FAB here.

Section 37.4: RippleDrawable

Ripple touch effect was introduced with material design in Android 5.0 (API level 21) and the animation is implemented by the new RippleDrawable class.

> Drawable that shows a ripple effect in response to state changes. The anchoring position of the ripple for a given state may be specified by calling `setHotspot(float x, float y)` with the corresponding state attribute identifier.

`Version ≥ 5.0`

In general, ripple effect for **regular buttons works by default** in API 21 and above, and for other touchable views, it can be achieved by specifying:

```
android:background="?android:attr/selectableItemBackground">
```

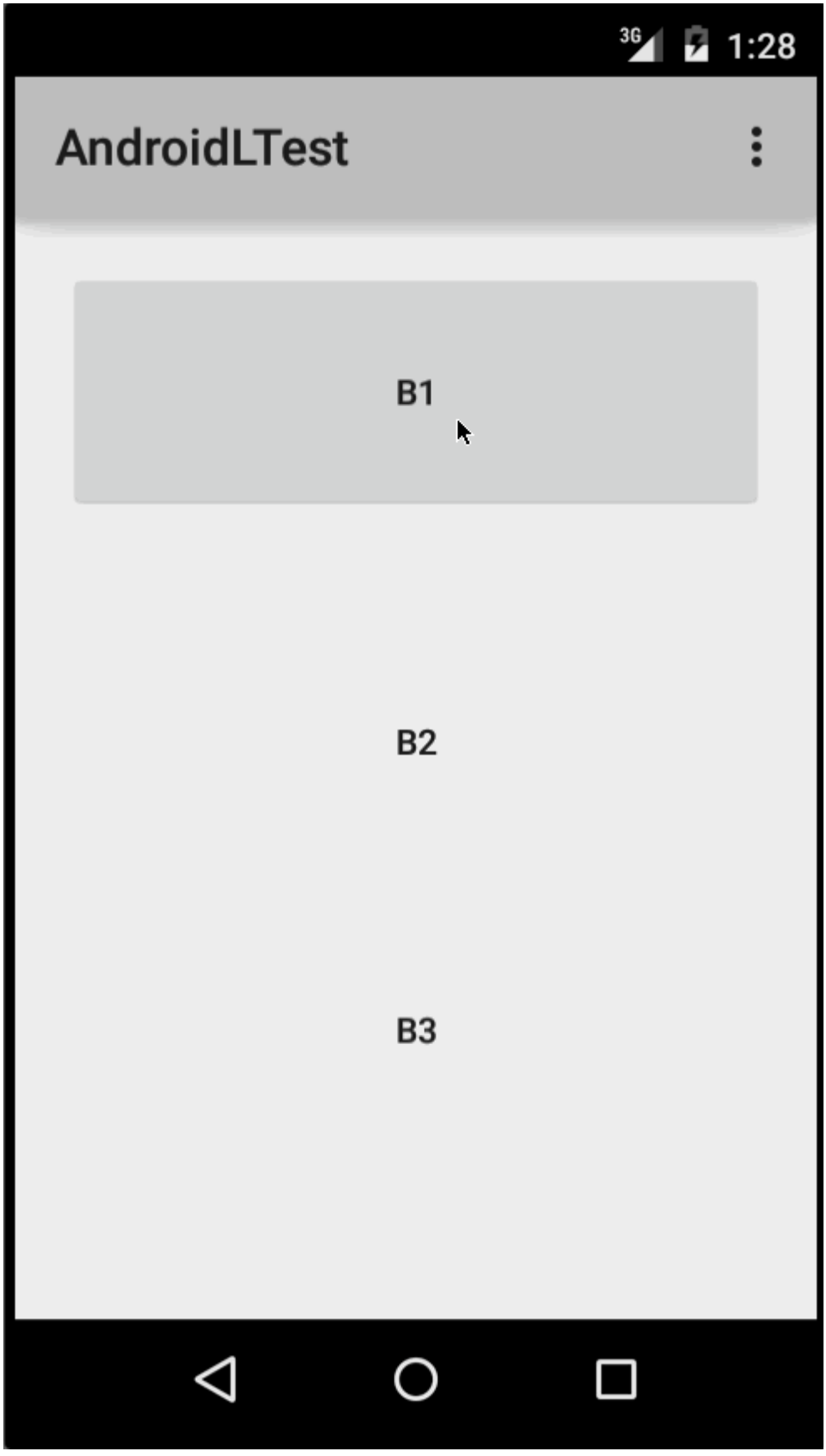
for ripples contained within the view or:

```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

for ripples that extend beyond the view's bounds.

For example, in the image below,

- B1 is a button that does not have any background,
- B2 is set up with `android:background="android:attr/selectableItemBackground"`
- B3 is set up with `android:background="android:attr/selectableItemBackgroundBorderless"`



(Image courtesy: http://blog.csdn.net/a396901990/article/details/40187203)

You can achieve the same in code using:

```
int[] attrs = new int[]{R.attr.selectableItemBackground};
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);
int backgroundResource = typedArray.getResourceId(0, 0);
```

```
myView.setBackgroundResource(backgroundResource);
```

Ripples can also be added to a view using the `android:foreground` attribute the same way as above. As the name suggests, in case the ripple is added to the foreground, the ripple will show up above any view it is added to (e.g. `ImageView`, a `LinearLayout` containing multiple views, etc).

If you want to customize the ripple effect into a view, you need to create a new `XML` file, inside the drawable directory.

Here are few examples:

Example 1: An unbounded ripple

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
   android:color="#ffff0000" />
```

Example 2: Ripple with mask and background color

```
<ripple android:color="#7777777"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/mask"
        android:drawable="#ffff00" />
    <item android:drawable="@android:color/white"/>
</ripple>
```

If there is `view` with a background *already specified* with a `shape`, `corners` and any other tags, to add a ripple to that view use a `mask layer` and set the ripple as the background of the view.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
   <ripple xmlns:android="http://schemas.android.com/apk/res/android"
      android:color="?android:attr/colorControlHighlight">
       <item android:id="@android:id/mask">
          <shape
            android:shape="rectangle">
             solid android:color="#000000"/>
              <corners
               android:radius="25dp"/>
           </shape>
       </item>
       <item android:drawable="@drawable/rounded_corners" />
   </ripple>
```

Example 3: Ripple on top a drawable resource

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ff0000ff">
    <item android:drawable="@drawable/my_drawable" />
</ripple>
```

Usage: To attach your ripple xml file to any view, set it as background as following (assuming your ripple file is named `my_ripple.xml`):

```
<View
   android:id="@+id/myViewId"
   android:layout_width="wrap_content"
```

```
   android:layout_height="wrap_content"
   android:background="@drawable/my_ripple" />
```

Selector:

The ripple drawable can also be used in place of color state list selectors if your target version is v21 or above (you can also place the ripple selector in the `drawable-v21` folder):

```
<!-- /drawable/button.xml: -->
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:drawable="@drawable/button_pressed"/>
    <item android:drawable="@drawable/button_normal"/>
</selector>

<!--/drawable-v21/button.xml:-->
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
   android:color="?android:colorControlHighlight">
   <item android:drawable="@drawable/button_normal" />
</ripple>
```

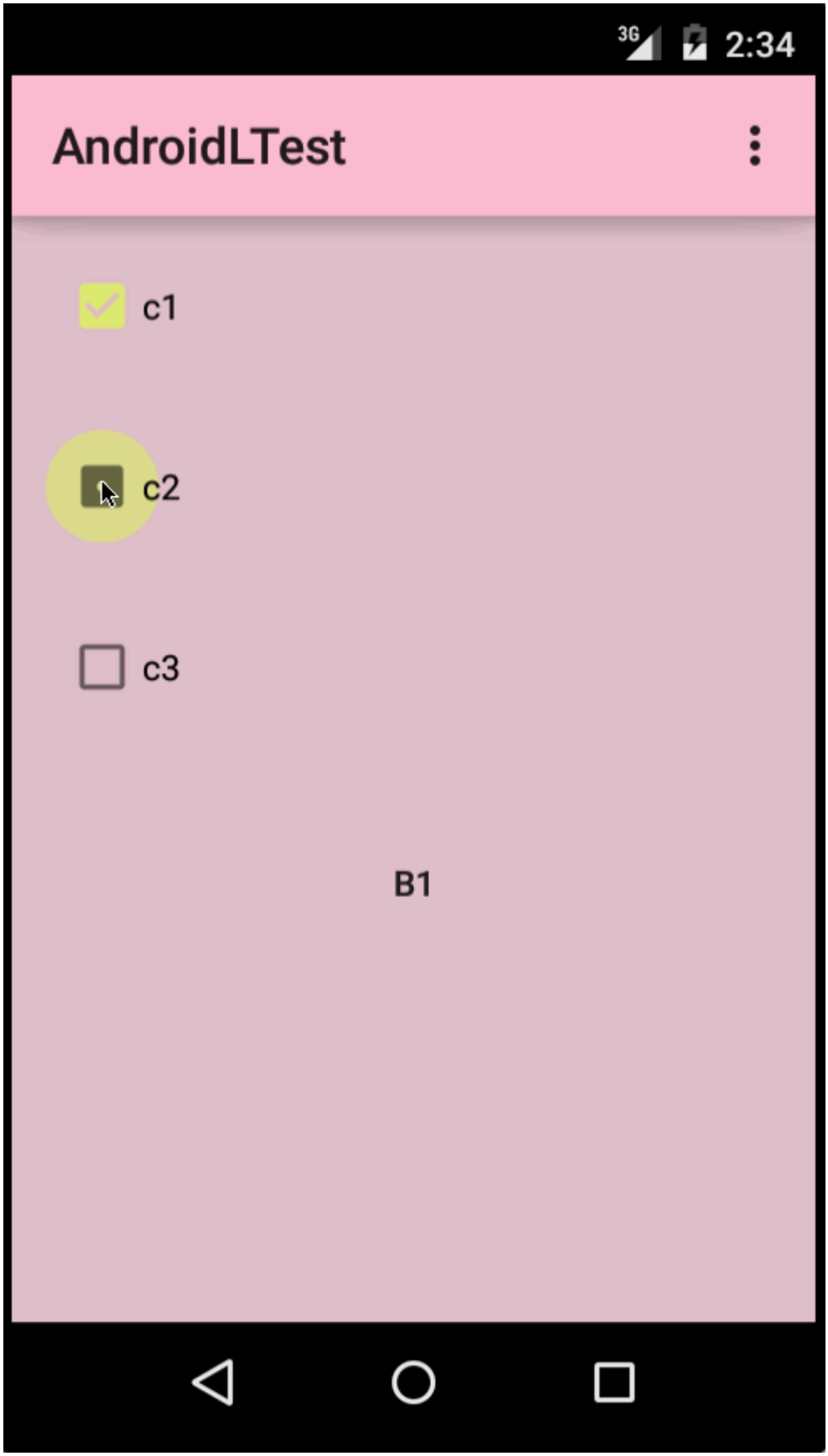
In this case, the color of the default state of your view would be white and the pressed state would show the ripple drawable.

Point to note: Using `?android:colorControlHighlight` will give the ripple the same color as the built-in ripples in your app.

To change just the ripple color, you can customize the color `android:colorControlHighlight` in your theme like so:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

  <style name="AppTheme" parent="android:Theme.Material.Light.DarkActionBar">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
  </style>

</resources>
```

and then use this theme in your activities, etc. The effect would be like the image below:



(Image courtesy: http://blog.csdn.net/a396901990/article/details/40187203)

Section 37.5: Adding a TabLayout

TabLayout provides a horizontal layout to display tabs, and is commonly used in conjunction with a ViewPager.

Make sure the following dependency is added to your app's `build.gradle` file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Now you can add items to a TabLayout in your layout using the TabItem class.

For example:

```
<android.support.design.widget.TabLayout
   android:layout_height="wrap_content"
   android:layout_width="match_parent"
   android:id="@+id/tabLayout">

    <android.support.design.widget.TabItem
       android:text="@string/tab_text_1"
       android:icon="@drawable/ic_tab_1" />

    <android.support.design.widget.TabItem
       android:text="@string/tab_text_2"
       android:icon="@drawable/ic_tab_2" />

</android.support.design.widget.TabLayout>
```

Add an `OnTabSelectedListener` to be notified when a tab in the `TabLayout` is selected/unselected/reselected:

```
TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        int position = tab.getPosition();
        // Switch to view for this tab
    }

    @Override
    public void onTabUnselected(TabLayout.Tab tab) {

    }

    @Override
    public void onTabReselected(TabLayout.Tab tab) {

    }
});
```

Tabs can also be added/removed from the `TabLayout` programmatically.

```
TabLayout.Tab tab = tabLayout.newTab();
tab.setText(R.string.tab_text_1);
tab.setIcon(R.drawable.ic_tab_1);
tabLayout.addTab(tab);

tabLayout.removeTab(tab);
tabLayout.removeTabAt(0);
tabLayout.removeAllTabs();
```

`TabLayout` has two modes, fixed and scrollable.

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

These can also be applied in XML:

```
<android.support.design.widget.TabLayout
   android:id="@+id/tabLayout"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   app:tabMode="fixed|scrollable" />
```

Note: the `TabLayout` modes are mutually exclusive, meaning only one can be active at a time.

The tab indicator color is the accent color defined for your Material Design theme.
You can override this color by defining a custom style in `styles.xml` and then applying the style to your TabLayout:

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">
    <item name="tabIndicatorColor">@color/your_color</item>
</style>
```

Then you can apply the style to the view using:

```
<android.support.design.widget.TabLayout
        android:id="@+id/tabs"
        style="@style/MyCustomTabLayoutStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
</android.support.design.widget.TabLayout>
```

Section 37.6: Bottom Sheets in Design Support Library

Bottom sheets slide up from the bottom of the screen to reveal more content.
They were added to the Android Support Library in v25.1.0 version and supports above all the versions.

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Persistent Bottom Sheets

You can achieve a Persistent Bottom Sheet attaching a `BottomSheetBehavior` to a child View of a `CoordinatorLayout`:

```
<android.support.design.widget.CoordinatorLayout >

    <!-- .....   -->

    <LinearLayout
      android:id="@+id/bottom_sheet"
      android:elevation="4dp"
      android:minHeight="120dp"
      app:behavior_peekHeight="120dp"
      ...
      app:layout_behavior="android.support.design.widget.BottomSheetBehavior">

           <!-- .....   -->

       </LinearLayout>

</android.support.design.widget.CoordinatorLayout>
```

Then in your code you can create a reference using:

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

You can set the state of your BottomSheetBehavior using the setState() method:

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

You can use one of these states:

- STATE_COLLAPSED: this collapsed state is the default and shows just a portion of the layout along the bottom. The height can be controlled with the `app:behavior_peekHeight` attribute (defaults to 0)
- STATE_EXPANDED: the fully expanded state of the bottom sheet, where either the whole bottom sheet is visible (if its height is less than the containing `CoordinatorLayout`) or the entire `CoordinatorLayout` is filled
- STATE_HIDDEN: disabled by default (and enabled with the `app:behavior_hideable` attribute), enabling this allows users to swipe down on the bottom sheet to completely hide the bottom sheet

Further to open or close the BottomSheet on click of a View of your choice, A Button let's say, here is how to toggle the sheet behavior and update view.

```
mButton = (Button) findViewById(R.id.button_2);
    //On Button click we monitor the state of the sheet
    mButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_EXPANDED) {
                //If expanded then collapse it (setting in Peek mode).
                mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_COLLAPSED);
                mButton.setText(R.string.button2_hide);
            } else if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_COLLAPSED) {
                //If Collapsed then hide it completely.
                mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_HIDDEN);
                mButton.setText(R.string.button2);
            } else if (mBottomSheetBehavior.getState() == BottomSheetBehavior.STATE_HIDDEN) {
                //If hidden then Collapse or Expand, as the need be.
                mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
                mButton.setText(R.string.button2_peek);
            }
        }
    });
```

But `BottomSheet` behavior also has a feature where user can interact with the swipe UP or Down it with a DRAG motion. In such a case, we might not be able to update the dependent View (like the button above) If the Sheet state has changed. For that matter, you'd like to receive callbacks of state changes, hence you can add a `BottomSheetCallback` to listen to user swipe events:

```
mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
      // React to state change and notify views of the current state
    }
      @Override
      public void onSlide(@NonNull View bottomSheet, float slideOffset) {
       // React to dragging events and animate views or transparency of dependent views
   }
```

```
});
```

And if you only want your Bottom Sheet to be visible only in COLLAPSED and EXPANDED mode toggles and never HIDE use:

```
mBottomSheetBehavior2.setHideable(false);
```

Bottom Sheet DialogFragment

You can also display a BottomSheetDialogFragment in place of a View in the bottom sheet. To do this, you first need to create a new class that extends BottomSheetDialogFragment.

Within the `setupDialog()` method, you can inflate a new layout file and retrieve the BottomSheetBehavior of the container view in your Activity. Once you have the behavior, you can create and associate a BottomSheetCallback with it to dismiss the Fragment when the sheet is hidden.

```
public class BottomSheetDialogFragmentExample extends BottomSheetDialogFragment {

    private BottomSheetBehavior.BottomSheetCallback mBottomSheetBehaviorCallback = new
BottomSheetBehavior.BottomSheetCallback() {

        @Override
        public void onStateChanged(@NonNull View bottomSheet, int newState) {
            if (newState == BottomSheetBehavior.STATE_HIDDEN) {
                dismiss();
            }

        }

        @Override
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {
        }
    };

    @Override
    public void setupDialog(Dialog dialog, int style) {
        super.setupDialog(dialog, style);
        View contentView = View.inflate(getContext(), R.layout.fragment_bottom_sheet, null);
        dialog.setContentView(contentView);

        CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) ((View)
contentView.getParent()).getLayoutParams();
        CoordinatorLayout.Behavior behavior = params.getBehavior();

        if( behavior != null && behavior instanceof BottomSheetBehavior ) {
            ((BottomSheetBehavior) behavior).setBottomSheetCallback(mBottomSheetBehaviorCallback);
        }
    }
}
```

Finally, you can call show() on an instance of your Fragment to display it in the bottom sheet.

```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();
bottomSheetDialogFragment.show(getSupportFragmentManager(), bottomSheetDialogFragment.getTag());
```

You can find more details in the dedicated topic

Section 37.7: Apply an AppCompat theme

The AppCompat support library provides themes to build apps with the Material Design specification. A theme with a parent of `Theme.AppCompat` is also required for an Activity to extend `AppCompatActivity`.

The first step is to customize your theme's color palette to automatically colorize your app.
In your app's `res/styles.xml` you can define:

```
<!-- inherit from the AppCompat theme -->
<style name="AppTheme" parent="Theme.AppCompat">

    <!-- your app branding color for the app bar -->
    <item name="colorPrimary">#2196f3</item>

    <!-- darker variant for the status bar and contextual app bars -->
    <item name="colorPrimaryDark">#1976d2</item>

    <!-- theme UI controls like checkboxes and text fields -->
    <item name="colorAccent">#f44336</item>
</style>
```

Instead of `Theme.AppCompat`, which has a dark background, you can also use `Theme.AppCompat.Light` or `Theme.AppCompat.Light.DarkActionBar`.

You can customize the theme with your own colours. Good choices are in the Material design specification colour chart, and Material Palette. The "500" colours are good choices for primary (blue 500 in this example); choose "700" of the same hue for the dark one; and an a shade from a different hue as the accent colour. The primary colour is used for your app's toolbar and its entry in the overview (recent apps) screen, the darker variant to tint the status bar, and the accent colour to highlight some controls.

After creating this theme, apply it to your app in the `AndroidManifest.xml` and also apply the theme to any particular activity. This is useful for applying a `AppTheme.NoActionBar` theme, which lets you implement non-default toolbar configurations.

```
<application android:theme="@style/AppTheme"
   ...>
    <activity
       android:name=".MainActivity"
       android:theme="@style/AppTheme" />
</application>
```

You can also apply themes to individual Views using `android:theme` and a `ThemeOverlay` theme. For example with a `Toolbar`:

```
<android.support.v7.widget.Toolbar
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:background="?attr/colorPrimary"
 android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />
```

or a `Button`:

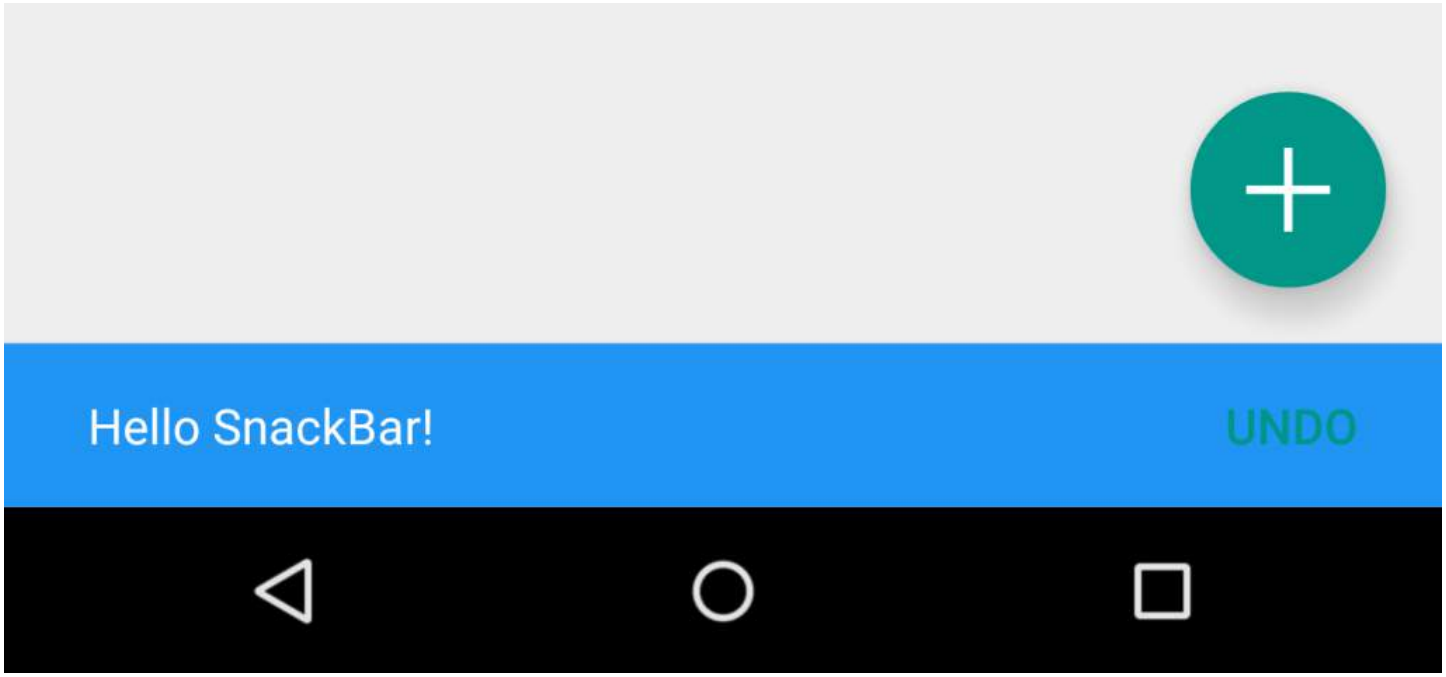
```
<Button
   style="@style/Widget.AppCompat.Button.Colored"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:theme="@style/MyButtonTheme"/>
```

```
<!-- res/values/themes.xml -->
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
    <item name="colorAccent">@color/my_color</item>
</style>
```

Section 37.8: Add a Snackbar

One of the main features in Material Design is the addition of a `Snackbar`, which in theory replaces the previous `Toast`. As per the Android documentation:

> Snackbars contain a single line of text directly related to the operation performed. They may contain a text action, but no icons. Toasts are primarily used for system messaging. They also display at the bottom of the screen, but may not be swiped off-screen.



Toasts can still be used in Android to display messages to users, however if you have decided to opt for material design usage in your app, it is recommended that you actually use a snackbar. Instead of being displayed as an overlay on your screen, a `Snackbar` pops from the bottom.

Here is how it is done:

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

As for the length of time to show the `Snackbar`, we have the options similar to the ones offered by a `Toast` or we could set a custom duration in milliseconds:

- `LENGTH_SHORT`
- `LENGTH_LONG`
- `LENGTH_INDEFINITE`
- `setDuration()` (since version `22.2.1`)

You can also add dynamic features to your `Snackbar` such as `ActionCallback` or custom color. However do pay attention to the design guideline offered by Android when customising a `Snackbar`.

Implementing the `Snackbar` has one limitation however. The parent layout of the view you are going to implement a `Snackbar` in needs to be a `CoordinatorLayout`. This is so that the actual popup from the bottom can be made.

This is how to define a `CoordinatorLayout` in your layout xml file:

```
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/coordinatorLayout"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

    //any other widgets in your layout go here.

</android.support.design.widget.CoordinatorLayout>
```

The `CoordinatorLayout` then needs to be defined in your Activity's `onCreate` method, and then used when creating the `Snackbar` itself.

For more information about about the `Snackbar`, please check the official documentation or the dedicated topic in the documentation.

Section 37.9: Add a Navigation Drawer

Navigation Drawers are used to navigate to top-level destinations in an app.

Make sure that you have added design support library in your `build.gradle` file under dependencies:

```
 dependencies {
    // ...
    compile 'com.android.support:design:25.3.1'
}
```

Next, add the `DrawerLayout` and `NavigationView` in your XML layout resource file.
The `DrawerLayout` is just a fancy container that allows the `NavigationView`, the actual navigation drawer, to slide out from the left or right of the screen. Note: for mobile devices, the standard drawer size is 320dp.

```
<!-- res/layout/activity_main.xml -->
<android.support.v4.widget.DrawerLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/navigation_drawer_layout"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:fitsSystemWindows="true"
   tools:openDrawer="start">
    <! -- You can use "end" to open drawer from the right side -->

    <android.support.design.widget.CoordinatorLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:fitsSystemWindows="true">

        <android.support.design.widget.AppBarLayout
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:theme="@style/AppTheme.AppBarOverlay">
```

```
            <android.support.v7.widget.Toolbar
               android:id="@+id/toolbar"
               android:layout_width="match_parent"
               android:layout_height="?attr/actionBarSize"
               android:background="?attr/colorPrimary"
               app:popupTheme="@style/AppTheme.PopupOverlay" />

        </android.support.design.widget.AppBarLayout>

    </android.support.design.widget.CoordinatorLayout>

    <android.support.design.widget.NavigationView
       android:id="@+id/navigation_drawer"
       android:layout_width="320dp"
       android:layout_height="match_parent"
       android:layout_gravity="start"
       android:fitsSystemWindows="true"
       app:headerLayout="@layout/drawer_header"
       app:menu="@menu/navigation_menu" />

</android.support.v4.widget.DrawerLayout>
```

Now, if you wish, create a **header file** that will serve as the top of your navigation drawer. This is used to give a much more elegant look to the drawer.

```
<!-- res/layout/drawer_header.xml -->
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="190dp">

    <ImageView
        android:id="@+id/header_image"
        android:layout_width="140dp"
        android:layout_height="120dp"
        android:layout_centerInParent="true"
        android:scaleType="centerCrop"
        android:src="@drawable/image" />

    <TextView
        android:id="@+id/header_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/header_image"
        android:text="User name"
        android:textSize="20sp" />

</RelativeLayout>
```

It is referenced in the `NavigationView` tag in the `app:headerLayout="@layout/drawer_header"` attribute.
This `app:headerLayout` inflates the specified layout into the header automatically. This can alternatively be done at runtime with:

```
// Lookup navigation view
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
// Inflate the header view at runtime
View headerLayout = navigationView.inflateHeaderView(R.layout.drawer_header);
```

To automatically populate your navigation drawer with material design-compliant navigation items, create a menu

file and add items as needed. Note: while icons for items aren't required, they are suggested in the Material Design specification.
It is referenced in the `NavigationView` tag in the `app:menu="@menu/navigation_menu"` attribute.

```
<!-- res/menu/menu_drawer.xml -->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
       android:id="@+id/nav_item_1"
       android:title="Item #1"
       android:icon="@drawable/ic_nav_1" />
    <item
       android:id="@+id/nav_item_2"
       android:title="Item #2"
       android:icon="@drawable/ic_nav_2" />
    <item
       android:id="@+id/nav_item_3"
       android:title="Item #3"
       android:icon="@drawable/ic_nav_3" />
    <item
       android:id="@+id/nav_item_4"
       android:title="Item #4"
       android:icon="@drawable/ic_nav_4" />
</menu>
```

To separate items into groups, put them into a **`<menu>`** nested in another **`<item>`** with an `android:title` attribute or wrap them with the **`<group>`** tag.

Now that the layout is done, move on to the `Activity` code:

```
  // Find the navigation view
  NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_drawer);
  navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
      @Override
      public boolean onNavigationItemSelected(MenuItem item) {
          // Get item ID to determine what to do on user click
          int itemId = item.getItemId();
          // Respond to Navigation Drawer selections with a new Intent
          startActivity(new Intent(this, OtherActivity.class));
          return true;
      }
  });

  DrawerLayout drawer = (DrawerLayout) findViewById(R.id.navigation_drawer_layout);
  // Necessary for automatically animated navigation drawer upon open and close
  ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, "Open navigation drawer",
"Close navigation drawer");
  // The two Strings are not displayed to the user, but be sure to put them into a separate
strings.xml file.
  drawer.addDrawerListener(toggle);
  toogle.syncState();
```

You can now do whatever you want in the header view of the `NavigationView`

```
View headerView = navigationView.getHeaderView();
TextView headerTextView = (TextView) headerview.findViewById(R.id.header_text_view);
ImageView headerImageView = (ImageView) headerview.findViewById(R.id.header_image);
// Set navigation header text
headerTextView.setText("User name");
// Set navigation header image
```

```
headerImageView.setImageResource(R.drawable.header_image);
```

The header view behaves like any other `View`, so once you use `findViewById()` and add some other `View`s to your layout file, you can set the properties of anything in it.

You can find more details and examples in the dedicated topic.

Section 37.10: How to use TextInputLayout

Make sure the following dependency is added to your app's `build.gradle` file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Show the hint from an EditText as a floating label when a value is entered.

```
<android.support.design.widget.TextInputLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content">

     <android.support.design.widget.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/form_username"/>

</android.support.design.widget.TextInputLayout>
```

For displaying the password display eye icon with TextInputLayout, we can make use of the following code:

```
<android.support.design.widget.TextInputLayout
    android:id="@+id/input_layout_current_password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:passwordToggleEnabled="true">

    <android.support.design.widget.TextInputEditText

        android:id="@+id/current_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/current_password"
        android:inputType="textPassword" />

</android.support.design.widget.TextInputLayout>
```

where `app:passwordToggleEnabled="true"` & `android:inputType="textPassword"` parameters are required.

`app` should use the namespace `xmlns:app="http://schemas.android.com/apk/res-auto"`

You can find more details and examples in the dedicated topic.

Chapter 38: Resources

Section 38.1: Define colors

Colors are usually stored in a resource file named `colors.xml` in the `/res/values/` folder.

They are defined by `<color>` elements:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>

    <color name="blackOverlay">#66000000</color>
</resources>
```

Colors are represented by hexadecimal color values for each color channel `(0 - FF)` in one of the formats:

- `#RGB`
- `#ARGB`
- `#RRGGBB`
- `#AARRGGBB`

Legend

- A - alpha channel - 0 value is fully transparent, FF value is opaque
- R - red channel
- G - green channel
- B - blue channel

Defined colors can be used in XML with following syntax `@color/name_of_the_color`

For example:

```
<RelativeLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/blackOverlay">
```

Using colors in code

These examples assume **`this`** is an Activity reference. A Context reference can be used in its place as well.

`Version ≥ 1.6`

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

`Version < 6.0`

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

In above declaration `colorPrimary`, `colorPrimaryDark` and `colorAccent` are used to define Material design colors that will be used in defining custom Android theme in `styles.xml`. They are automatically added when new project is created with Android Studio.

Section 38.2: Color Transparency(Alpha) Level

Hex Opacity Values

Alpha(%)	Hex Value
100%	FF
95%	F2
90%	E6
85%	D9
80%	CC
75%	BF
70%	B3
65%	A6
60%	99
55%	8C
50%	80
45%	73
40%	66
35%	59
30%	4D
25%	40
20%	33
15%	26
10%	1A
5%	0D
0%	00

If you want to set 45% to red color.

```
<color name="red_with_alpha_45">#73FF0000</color>
```

hex value for red - #FF0000

You can add 73 for 45% opacity in prefix - #73FF0000

Section 38.3: Define String Plurals

To differentiate between plural and singular strings, you can define a plural in your *strings.xml* file and list the different quantities, as shown in the example below:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="hello_people">
        <item quantity="one">Hello to %d person</item>
        <item quantity="other">Hello to %d people</item>
    </plurals>
</resources>
```

This definition can be accessed from Java code by using the `getQuantityString()` method of the `Resources` class, as shown in the following example:

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

Here, the first parameter `R.plurals.hello_people` is the resource name. The second parameter (3 in this example)

is used to pick the correct `quantity` string. The third parameter (also `3` in this example) is the format argument that will be used for substituting the format specifier `%d`.

Possible quantity values (listed in alphabetical order) are:

```
few
many
one
other
two
zero
```

It is important to note that not all locales support every denomination of `quantity`. For example, the Chinese language does not have a concept of `one` item. English does not have a `zero` item, as it is grammatically the same as `other`. Unsupported instances of `quantity` will be flagged by the IDE as Lint warnings, but won't cause complication errors if they are used.

Section 38.4: Define strings

Strings are typically stored in the resource file `strings.xml`. They are defined using a **`<string>`** XML element.

The purpose of strings.xml is to allow internationalisation. You can define a strings.xml for each language iso code. Thus when the system looks for the string 'app_name' it first checks the xml file corresponding to the current language, and if it is not found, looks for the entry in the default strings.xml file. This means you can choose to only localise some of your strings while not others.

/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Hello World App</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

Once a string is defined in an XML resource file, it can be used by other parts of the app.

An app's XML project files can use a **`<string>`** element by referring to `@string/string_name`. For example, an app's manifest (/manifests/AndroidManifest.xml) file includes the following line by default in Android Studio:

```
android:label="@string/app_name"
```

This tells android to look for a **`<string>`** resource called "app_name" to use as the name for the app when it is installed or displayed in a launcher.

Another time you would use a **`<string>`** resource from an XML file in android would be in a layout file. For example, the following represents a TextView which displays the `hello_world` string we defined earlier:

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/hello_world"/>
```

You can also access **`<string>`** resources from the java portion of your app. To recall our same `hello_world` string from above within an Activity class, use:

```
String helloWorld = getString(R.string.hello_world);
```

Section 38.5: Define dimensions

Dimensions are typically stored in a resource file names `dimens.xml`. They are defined using a **`<dimen>`** element.

res/values/dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="small_padding">5dp</dimen>
    <dimen name="medium_padding">10dp</dimen>
    <dimen name="large_padding">20dp</dimen>

    <dimen name="small_font">14sp</dimen>
    <dimen name="medium_font">16sp</dimen>
    <dimen name="large_font">20sp</dimen>
</resources>
```

You can use different units :

- **sp :** Scale-independent Pixels. For fonts.
- **dp :** Density-independent Pixels. For everything else.
- **pt :** Points
- **px :** Pixels
- **mm :** Millimeters
- **im :** Inches

Dimensions can now be referenced in XML with the syntax `@dimen/name_of_the_dimension`.

For example:

```
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="@dimen/large_padding">
</RelativeLayout>
```

Section 38.6: String formatting in strings.xml

Defining Strings in the strings.xml file also allows for string formatting. The only caveat is that the String will need to be dealt with in code like below, versus simply attaching it to a layout.

```
<string name="welcome_trainer">Hello Pokémon Trainer, %1$s! You have caught %2$d Pokémon.</string>
```

```
String welcomePokemonTrainerText = getString(R.string.welcome_trainer, tranerName, pokemonCount);
```

In above example,
%1$s
'%' separates from normal characters,
'1' denotes first parameter,
'$' is used as separator between parameter number and type,
's' denotes string type ('d' is used for integer)

Note that `getString()` is a method of `Context` or `Resources`, i.e. you can use it directly within an `Activity` instance, or else you may use `getActivity().getString()` or `getContext().getString()` respectively.

Section 38.7: Define integer array

In order to define an integer array write in a resources file

res/values/filename.xml

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
    <item>@integer/integer_id</item>
</integer-array>
```

for example

res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer-array name="fibo">
        <item>@integer/zero</item>
        <item>@integer/one</item>
        <item>@integer/one</item>
        <item>@integer/two</item>
        <item>@integer/three</item>
        <item>@integer/five</item>
    </integer-array>
</resources>
```

and use it from java like

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG",Arrays.toString(values)));
```

Output

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

Section 38.8: Define a color state list

Color state lists can be used as colors, but will change depending on the state of the view they are used for.

To define one, create a resource file in `res/color/foo.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="#888888" android:state_enabled="false"/>
    <item android:color="@color/lightGray" android:state_selected="false"/>
    <item android:color="@android:color/white" />
</selector>
```

Items are evaluated in the order they are defined, and the first item whose specified states match the current state of the view is used. So it's a good practice to specify a catch-all at the end, without any state selectors specified.

Each item can either use a color literal, or reference a color defined somewhere else.

Section 38.9: 9 Patches

9 Patches are **stretchable** images in which the areas which can be stretched are defined by black markers on a transparent border.

There is a great tutorial here.
Despite being so old, it's still so valuable and it helped many of us to deeply understand the 9 patch gear.

Unfortunately, recently that page has been put down for a while (it's currently up again).

Hence, the need to have a physical copy of that page for android developers on our reliable server/s.

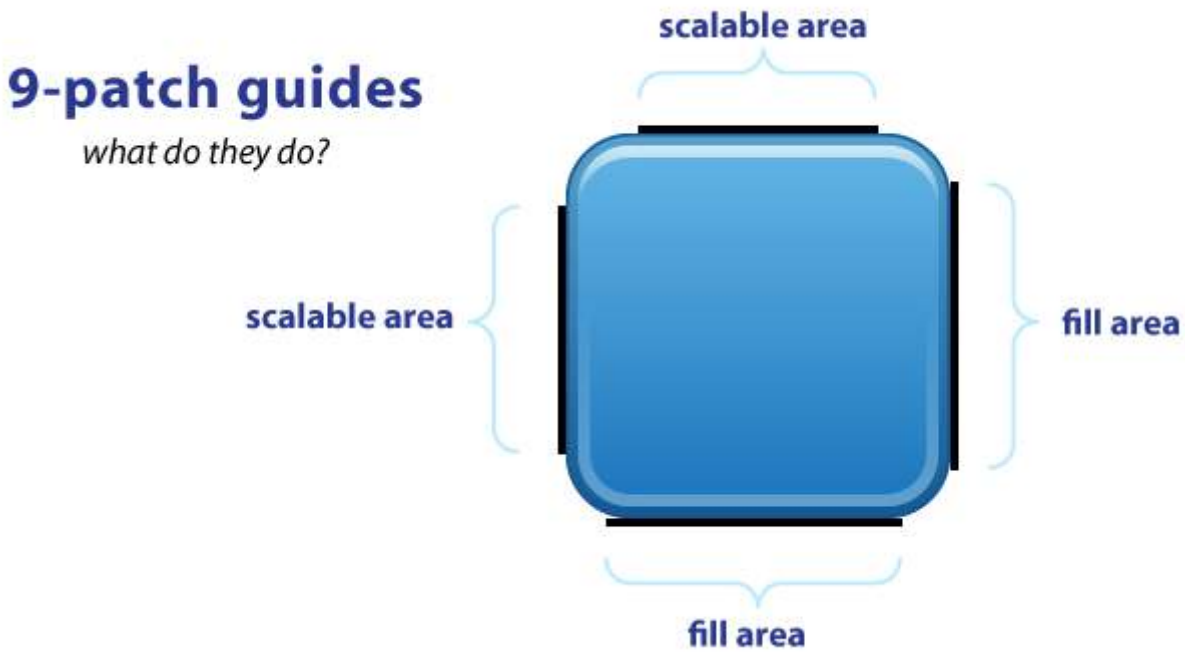
Here it is.

A SIMPLE GUIDE TO 9-PATCH FOR ANDROID UI May 18, 2011

While I was working on my first Android app, I found 9-patch (aka 9.png) to be confusing and poorly documented. After a little while, I finally picked up on how it works and decided to throw together something to help others figure it out.

Basically, 9-patch uses png transparency to do an advanced form of 9-slice or scale9. The guides are straight, 1-pixel black lines drawn on the edge of your image that define the scaling and fill of your image. By naming your image file name.9.png, Android will recognize the 9.png format and use the black guides to scale and fill your bitmaps.

Here's a basic guide map:



As you can see, you have guides on each side of your image. The TOP and LEFT guides are for scaling your image (i.e. 9-slice), while the RIGHT and BOTTOM guides define the fill area.

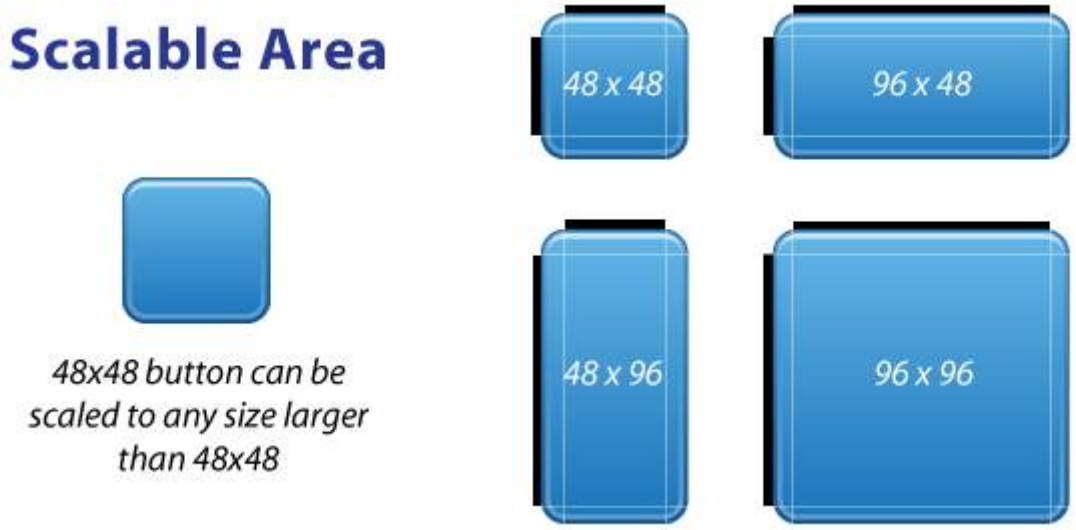
The black guide lines are cut-off/removed from your image – they won't show in the app. Guides must only be one pixel wide, so if you want a 48×48 button, your png will actually be 50×50. Anything thicker than one pixel will remain part of your image. (My examples have 4-pixel wide guides for better visibility. They should really be only 1-pixel).

Your guides must be solid black (#000000). Even a slight difference in color (#000001) or alpha will cause it to fail

and stretch normally. This failure won't be obvious either*, it fails silently! Yes. Really. Now you know.

Also you should keep in mind that remaining area of the one-pixel outline must be completely transparent. This includes the four corners of the image – those should always be clear. This can be a bigger problem than you realize. For example, if you scale an image in Photoshop it will add anti-aliased pixels which may include almost-invisible pixels which will also cause it to fail*. If you must scale in Photoshop, use the Nearest Neighbor setting in the Resample Image pulldown menu (at the bottom of the Image Size pop-up menu) to keep sharp edges on your guides.

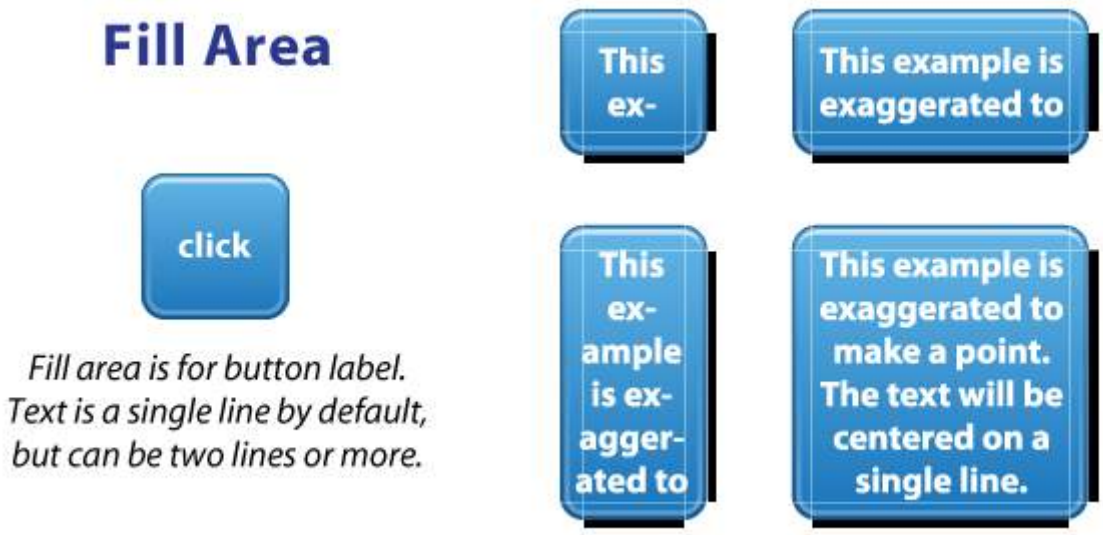
*(updated 1/2012) This is actually a "fix" in the latest dev kit. Previously it would manifest itself as all of your other images and resources suddenly breaking, not the actually broken 9-patch image.



The TOP and LEFT guides are used to define the scalable portion of your image – LEFT for scaling height, TOP for scaling width. Using a button image as an example, this means the button can stretch horizontally and vertically within the black portion and everything else, such as the corners, will remain the same size. The allows you to have buttons that can scale to any size and maintain a uniform look.

It's important to note that 9-patch images don't scale down – they only scale up. So it's best to start as small as possible.

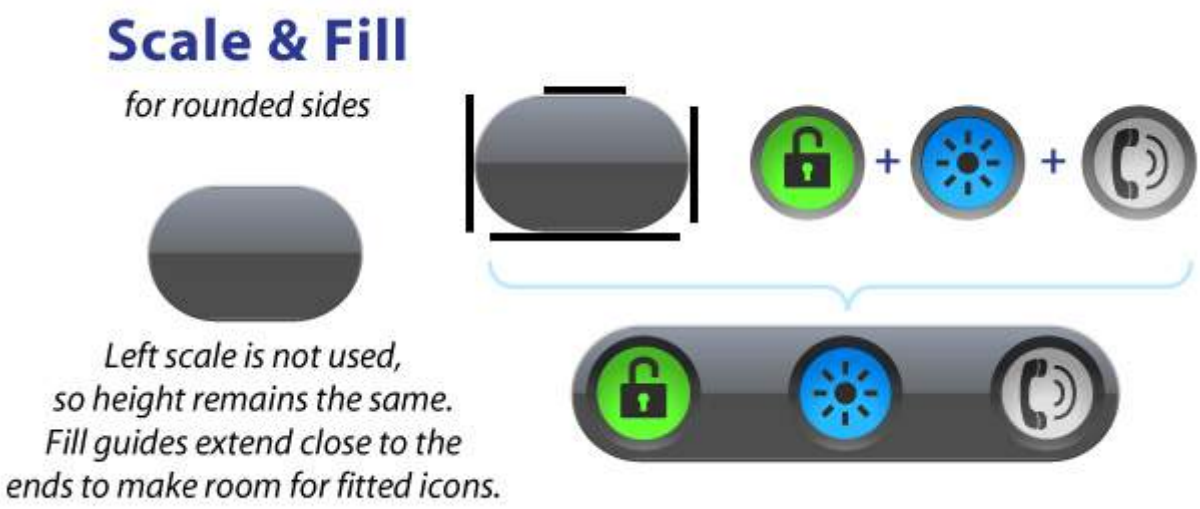
Also, you can leave out portions in the middle of the scale line. So for example, if you have a button with a sharp glossy edge across the middle, you can leave out a few pixels in the middle of the LEFT guide. The center horizontal axis of your image won't scale, just the parts above and below it, so your sharp gloss won't get anti-aliased or fuzzy.



Fill area guides are optional and provide a way define the area for stuff like your text label. Fill determines how much room there is within your image to place text, or an icon, or other things. 9-patch isn't just for buttons, it works for background images as well.

The above button & label example is exaggerated simply to explain the idea of fill – the label isn't completely accurate. To be honest, I haven't experienced how Android does multi-line labels since a button label is usually a single row of text.

Finally, here's a good demonstration of how scale and fill guides can vary, such as a LinearLayout with a background image & fully rounded sides:

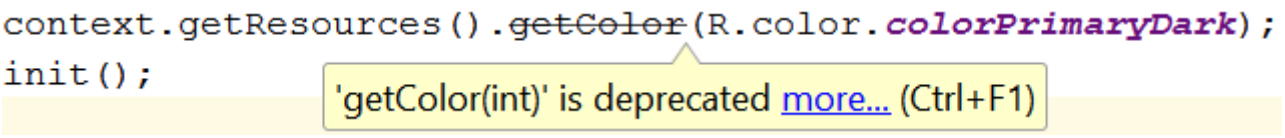


With this example, the LEFT guide isn't used but we're still required to have a guide. The background image don't scale vertically; it just scales horizontally (based on the TOP guide). Looking at the fill guides, the RIGHT and BOTTOM guides extend beyond where they meet the image's curved edges. This allows me to place my round buttons close to the edges of the background for a tight, fitted look.

So that's it. 9-patch is super easy, once you get it. It's not a perfect way to do scaling, but the fill-area and multi-line scale-guides does offer more flexibility than traditional 9-slice and scale9. Give it a try and you'll figure it out quickly.

Section 38.10: Getting resources without "deprecated" warnings

Using the Android API 23 or higher, very often such situation can be seen:



This situation is caused by the structural change of the Android API regarding getting the resources.
Now the function:

```
public int getColor(@ColorRes int id, @Nullable Theme theme) throws NotFoundException
```

should be used. But the `android.support.v4` library has another solution.

Add the following dependency to the `build.gradle` file:

```
com.android.support:support-v4:24.0.0
```

Then all methods from support library are available:

```
ContextCompat.getColor(context, R.color.colorPrimaryDark);
ContextCompat.getDrawable(context, R.drawable.btn_check);
ContextCompat.getColorStateList(context, R.color.colorPrimary);
DrawableCompat.setTint(drawable);
ContextCompat.getColor(context,R.color.colorPrimaryDark));
```

Moreover more methods from support library can be used:

```
ViewCompat.setElevation(textView, 1F);
ViewCompat.animate(textView);
TextViewCompat.setTextAppearance(textView, R.style.AppThemeTextStyle);
...
```

Section 38.11: Working with strings.xml file

A string resource provides text strings for your application with optional text styling and formatting. There are three types of resources that can provide your application with strings:

String

```
XML resource that provides a single string.
```

Syntax:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_name">text_string</string>
</resources>
```

And to use this string in layout:

```
<TextView
    android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
    android:text="@string/string_name" />
```

String Array

```
XML resource that provides an array of strings.
```

Syntax:

```
    <resources>
<string-array name="planets_array">
    <item>Mercury</item>
    <item>Venus</item>
    <item>Earth</item>
    <item>Mars</item>
</string-array>
```

Usage

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Quantity Strings (Plurals)

```
XML resource that carries different strings for pluralization.
```

Syntax:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals
       name="plural_name">
        <item
           quantity=["zero" | "one" | "two" | "few" | "many" | "other"]
           >text_string</item>
    </plurals>
</resources>
```

Usage:

```
int count = getNumberOfsongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.plural_name, count, count);
```

Section 38.12: Define string array

In order to define a string array write in a resources file

res/values/filename.xml

```
<string-array name="string_array_name">
    <item>text_string</item>
    <item>@string/string_id</item>
</string-array>
```

for example

res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="string_array_example">
        <item>@string/app_name</item>
        <item>@string/hello_world</item>
    </string-array>
</resources>
```

and use it from java like

```
String[] strings = getResources().getStringArray(R.array.string_array_example;
Log.i("TAG",Arrays.toString(strings)));
```

Output

```
I/TAG: [HelloWorld, Hello World!]
```

Section 38.13: Define integers

Integers are typically stored in a resource file named `integers.xml`, but the file name can be chosen arbitrarily. Each integer is defined by using an `<integer>` element, as shown in the following file:

res/values/integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

Integers can now be referenced in XML with the syntax `@integer/name_of_the_integer`, as shown in the following example:

```
<ProgressBar
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:max="@integer/max"/>
```

Section 38.14: Define a menu resource and use it inside Activity/Fragment

Define a menu in **res/menu**

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/first_item_id"
        android:orderInCategory="100"
        android:title="@string/first_item_string"
        android:icon="@drawable/first_item_icon"
        app:showAsAction="ifRoom"/>
```

```
    <item
        android:id="@+id/second_item_id"
        android:orderInCategory="110"
        android:title="@string/second_item_string"
        android:icon="@drawable/second_item_icon"
        app:showAsAction="ifRoom"/>

</menu>
```

For more options of configuration refer to: Menu resource

Inside `Activity`:

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    ///Override defining menu resource
    inflater.inflate(R.menu.menu_resource_id, menu);
    super.onCreateOptionsMenu(menu, inflater);
}

@Override
public void onPrepareOptionsMenu(Menu menu) {
    //Override for preparing items (setting visibility, change text, change icon...)
    super.onPrepareOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    //Override it for handling items
    int menuItemId = item.getItemId();
    switch (menuItemId) {
        case: R.id.first_item_id
             return true; //return true, if is handled
    }
    return super.onOptionsItemSelected(item);
}
```

For invoking the methods above during showing the view, call `getActivity().invalidateOptionsMenu();`

Inside `Fragment` one additional call is needed:

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    setHasOptionsMenu(true);
    super.onCreateView(inflater, container, savedInstanceState);
}
```

Chapter 39: Data Binding Library

Section 39.1: Basic text field binding

Gradle (Module:app) Configuration

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Data model

```
public class Item {
    public String name;
    public String description;

    public Item(String name, String description) {
        this.name = name;
        this.description = description;
    }
}
```

Layout XML

The first step is wrapping your layout in a `<layout>` tag, adding a `<data>` element, and adding a `<variable>` element for your data model.

Then you can bind XML attributes to fields in the data model using `@{model.fieldname}`, where `model` is the variable's name and `fieldname` is the field you want to access.

item_detail_activity.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

       <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@{item.name}"/>

       <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@{item.description}"/>

   </LinearLayout>
```

```
</layout>
```

For each XML layout file properly configured with bindings, the Android Gradle plugin generates a corresponding class : bindings. Because we have a layout named *item_detail_activity*, the corresponding generated binding class is called `ItemDetailActivityBinding`.

This binding can then be used in an Activity like so:

```
public class ItemDetailActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       ItemDetailActivityBinding binding = DataBindingUtil.setContentView(this,
R.layout.item_detail_activity);
       Item item = new Item("Example item", "This is an example item.");
       binding.setItem(item);
    }
}
```

Section 39.2: Built-in two-way Data Binding

Two-way Data-Binding supports the following attributes:

Element	Properties
`AbsListView`	`android:selectedItemPosition`
`CalendarView`	`android:date`
`CompoundButton`	`android:checked`
`DatePicker`	• `android:year` • `android:month` • `android:day`
`EditText`	`android:text`
`NumberPicker`	`android:value`
`RadioGroup`	`android:checkedButton`
`RatingBar`	`android:rating`
`SeekBar`	`android:progress`
`TabHost`	`android:currentTab`
`TextView`	`android:text`
`TimePicker`	• `android:hour` • `android:minute`
`ToggleButton`	`android:checked`
`Switch`	`android:checked`

Usage

```
<layout ...>
    <data>
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

Notice that the Binding expression `@={}` **has an additional** `=`, which is necessary for the **two-way Binding**. It is not possible to use methods in two-way Binding expressions.

Section 39.3: Custom event using lambda expression

Define Interface

```
public interface ClickHandler {
    public void onButtonClick(User user);
}
```

Create Model class

```
public class User {
    private String name;

    public User(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Layout XML

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
           name="handler"
           type="com.example.ClickHandler"/>

        <variable
           name="user"
           type="com.example.User"/>
    </data>

    <RelativeLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent">

        <Button
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{user.name}"
           android:onClick="@{() -> handler.onButtonClick(user)}"/>
    </RelativeLayout>
</layout>
```

Activity code :

```
public class MainActivity extends Activity implements ClickHandler {

    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this,R.layout.activity_main);
        binding.setUser(new User("DataBinding User"));
        binding.setHandler(this);
    }

    @Override
    public void onButtonClick(User user) {
        Toast.makeText(MainActivity.this,"Welcome " + user.getName(),Toast.LENGTH_LONG).show();
    }
}
```

For some view listener which is not available in xml code but can be set in Java code, it can be bind with custom binding.

Custom class

```
public class BindingUtil {
    @BindingAdapter({"bind:autoAdapter"})
    public static void setAdapter(AutoCompleteTextView view, ArrayAdapter<String> pArrayAdapter) {
        view.setAdapter(pArrayAdapter);
    }
    @BindingAdapter({"bind:onKeyListener"})
    public static void setOnKeyListener(AutoCompleteTextView view , View.OnKeyListener
pOnKeyListener)
    {
        view.setOnKeyListener(pOnKeyListener);
    }
}
```

Handler class

```
public class Handler extends BaseObservable {
    private ArrayAdapter<String> roleAdapter;

    public ArrayAdapter<String> getRoleAdapter() {
        return roleAdapter;
    }
    public void setRoleAdapter(ArrayAdapter<String> pRoleAdapter) {
        roleAdapter = pRoleAdapter;
    }
}
```

XML

```
<layout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:bind="http://schemas.android.com/tools" >

    <data>
        <variable
           name="handler"
           type="com.example.Handler" />
    </data>

    <LinearLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:orientation="vertical" >
```

```
        <AutoCompleteTextView
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:singleLine="true"
           bind:autoAdapter="@{handler.roleAdapter}" />

    </LinearLayout>
</layout>
```

Section 39.4: Default value in Data Binding

> The Preview pane displays default values for data binding expressions if provided.

For example :

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

It will take `wrap_content` while designing and will act as a `wrap_content` in preview pane.

Another example is

```
android:text="@{user.name, default=`Preview Text`}"
```

It will display `Preview Text` in preview pane but when you run it in device/emulator actual text binded to it will be displayed

Section 39.5: Databinding in Dialog

```
public void doSomething() {
    DialogTestBinding binding = DataBindingUtil
            .inflate(LayoutInflater.from(context), R.layout.dialog_test, null, false);

    Dialog dialog = new Dialog(context);
    dialog.setContentView(binding.getRoot());
    dialog.show();
}
```

Section 39.6: Binding with an accessor method

If your model has private methods, the databinding library still allows you to access them in your view without using the full name of the method.

Data model

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Layout XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

       <!-- Since the "name" field is private on our data model,
           this binding will utilize the public getName() method instead. -->
       <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@{item.name}"/>

   </LinearLayout>
</layout>
```

Section 39.7: Pass widget as reference in BindingAdapter

Layout XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>

   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

        <ImageView
        android:id="@+id/img"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        app:imageUrl="@{url}"
        app:progressbar="@{progressBar}"/>

   </LinearLayout>
</layout>
```

BindingAdapter method

```
@BindingAdapter({"imageUrl","progressbar"})
public static void loadImage(ImageView view, String imageUrl, ProgressBar progressBar){
    Glide.with(view.getContext()).load(imageUrl)
                        .listener(new RequestListener<String, GlideDrawable>() {
                    @Override
                    public boolean onException(Exception e, String model, Target<GlideDrawable>
```

```
target, boolean isFirstResource) {
                        return false;
                    }

                    @Override
                    public boolean onResourceReady(GlideDrawable resource, String model,
Target<GlideDrawable> target, boolean isFromMemoryCache, boolean isFirstResource) {
                        progressBar.setVisibility(View.GONE);
                        return false;
                    }
                }).into(view);
}
```

Section 39.8: Click listener with Binding

Create interface for clickHandler

```
public interface ClickHandler {
    public void onButtonClick(View v);
}
```

Layout XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>
        <variable
           name="handler"
           type="com.example.ClickHandler"/>
    </data>

    <RelativeLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent">

        <Button
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="click me"
           android:onClick="@{handler.onButtonClick}"/>
    </RelativeLayout>
</layout>
```

Handle event in your Activity

```
public class MainActivity extends Activity implements ClickHandler {

    private ActivityMainBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = DataBindingUtil.setContentView(this,R.layout.activity_main);
        binding.setHandler(this);
    }

    @Override
    public void onButtonClick(View v) {
        Toast.makeText(context,"Button clicked",Toast.LENGTH_LONG).show();
```

```
    }
}
```

Section 39.9: Data binding in RecyclerView Adapter

It's also possible to use data binding within your `RecyclerView` Adapter.

Data model

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

XML Layout

```
<TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@{item.name}"/>
```

Adapter class

```
public class ListItemAdapter extends RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private Activity host;
    private List<Item> items;

    public ListItemAdapter(Activity activity, List<Item> items) {
        this.host = activity;
        this.items = items;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        // inflate layout and retrieve binding
        ListItemBinding binding = DataBindingUtil.inflate(host.getLayoutInflater(),
                R.layout.list_item, parent, false);

        return new ItemViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        Item item = items.get(position);

        ItemViewHolder itemViewHolder = (ItemViewHolder)holder;
        itemViewHolder.bindItem(item);
    }

    @Override
    public int getItemCount() {
        return items.size();
    }

    private static class ItemViewHolder extends RecyclerView.ViewHolder {
        ListItemBinding binding;
```

```
        ItemViewHolder(ListItemBinding binding) {
            super(binding.getRoot());
            this.binding = binding;
        }

        void bindItem(Item item) {
            binding.setItem(item);
            binding.executePendingBindings();
        }
    }
}
```

Section 39.10: Databinding in Fragment

Data Model

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

}
```

Layout XML

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

       <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@{item.name}"/>

   </LinearLayout>
</layout>
```

Fragment

```
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
    FragmentTest binding = DataBindingUtil.inflate(inflater, R.layout.fragment_test, container,
false);
    Item item = new Item();
    item.setName("Thomas");
```

```
    binding.setItem(item);
    return binding.getRoot();
}
```

Section 39.11: DataBinding with custom variables(int,boolean)

Sometimes we need to perform basic operations like hide/show view based on single value, for that single variable we cannot create model or it is not good practice to create model for that. DataBinding supports basic datatypes to perform those oprations.

```
<layout xmlns:android="http://schemas.android.com/apk/res/android">

    <data>

        <import type="android.view.View" />

        <variable
           name="selected"
           type="Boolean" />

    </data>

    <RelativeLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent">

        <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Hello World"
           android:visibility="@{selected ? View.VISIBLE : View.GONE}" />

    </RelativeLayout>
</layout>
```

and set its value from java class.

```
binding.setSelected(true);
```

Section 39.12: Referencing classes

Data model

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Layout XML

You must import referenced classes, just as you would in Java.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
```

```
        <import type="android.view.View"/>
        <variable name="item" type="com.example.Item"/>
    </data>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <!-- We reference the View class to set the visibility of this TextView -->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{item.name}"
            android:visibility="@{item.name == null ? View.VISIBLE : View.GONE"/>

    </LinearLayout>
</layout>
```

Note: The package `java.lang.*` is imported automatically by the system. (The same is made by `JVM` for `Java`)

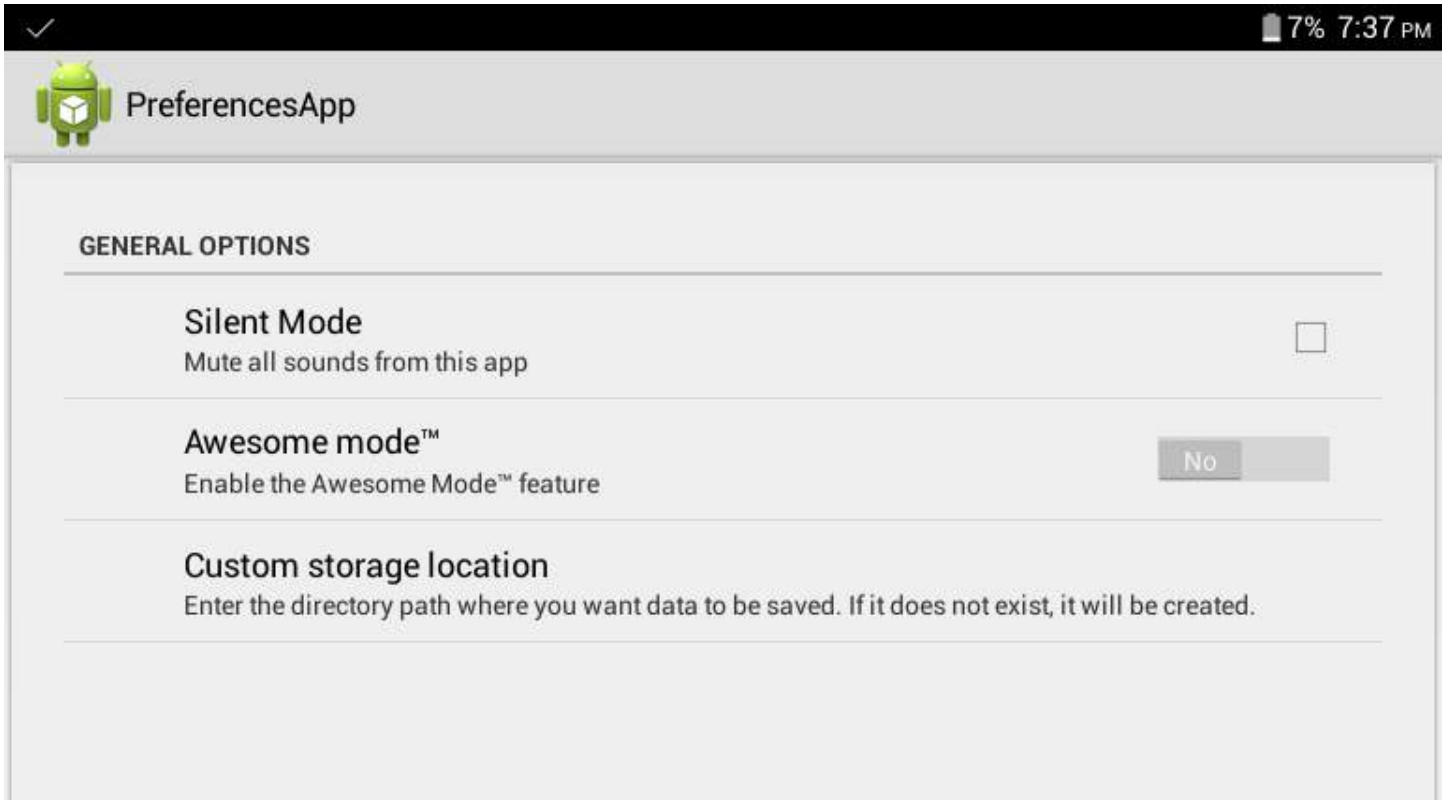
Chapter 40: SharedPreferences

Parameter	Details
key	A non-null `String` identifying the parameter. It can contain whitespace or non-printables. This is only used inside your app (and in the XML file), so it doesn't have to be namespaced, but it's a good idea to have it as a constant in your source code. Don't localize it.
defValue	All the get functions take a default value, which is returned if the given key is not present in the `SharedPreferences`. It's not returned if the key is present but the value has the wrong type: in that case you get a `ClassCastException`.

SharedPreferences provide a way to save data to disk in the form of **key-value** pairs.

Section 40.1: Implementing a Settings screen using SharedPreferences

One use of `SharedPreferences` is to implement a "Settings" screen in your app, where the user can set their preferences / options. Like this:



A PreferenceScreen saves user preferences in `SharedPreferences`. To create a PreferenceScreen, you need a few things:

An XML file to define the available options:

This goes in `/res/xml/preferences.xml`, and for the above settings screen, it looks like this:

```
<PreferenceScreen
   xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
       android:title="General options">
        <CheckBoxPreference
           android:key = "silent_mode"
           android:defaultValue="false"
           android:title="Silent Mode"
```

```
            android:summary="Mute all sounds from this app" />

        <SwitchPreference
            android:key="awesome_mode"
            android:defaultValue="false"
            android:switchTextOn="Yes"
            android:switchTextOff="No"
            android:title="Awesome mode™"
            android:summary="Enable the Awesome Mode™ feature"/>

        <EditTextPreference
            android:key="custom_storage"
            android:defaultValue="/sdcard/data/"
            android:title="Custom storage location"
            android:summary="Enter the directory path where you want data to be saved. If it does
not exist, it will be created."
            android:dialogTitle="Enter directory path (eg. /sdcard/data/ )"/>
    </PreferenceCategory>
</PreferenceScreen>
```

This defines the available options in the settings screen. There are many other types of `Preference` listed in the Android Developers documentation on the Preference Class.

Next, we need **an Activity to host our Preferences** user interface. In this case, it's quite short, and looks like this:

```
package com.example.preferences;

import android.preference.PreferenceActivity;
import android.os.Bundle;

public class PreferencesActivity extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

It extends `PreferenceActivity`, and provides the user interface for the preferences screen. It can be started just like a normal activity, in this case, with something like:

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

Don't forget to add `PreferencesActivity` to your `AndroidManifest.xml`.

Getting the values of the preferences inside your app is quite simple, just call `setDefaultValues()` first, in order to set the default values defined in your XML, and then get the default `SharedPreferences`. An example:

```
//set the default values we defined in the XML
PreferenceManager.setDefaultValues(this, R.xml.preferences, false);
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);

//get the values of the settings options
boolean silentMode = preferences.getBoolean("silent_mode", false);
boolean awesomeMode = preferences.getBoolean("awesome_mode", false);

String customStorage = preferences.getString("custom_storage", "");
```

Section 40.2: Commit vs. Apply

The `editor.apply()` method is **asynchronous**, while `editor.commit()` is **synchronous**.

Obviously, you should call either `apply()` or `commit()`.

Version ≥ 2.3

```
SharedPreferences settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean(PREF_CONST, true);
// This will asynchronously save the shared preferences without holding the current thread.
editor.apply();
```

```
SharedPreferences settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean(PREF_CONST, true);
// This will synchronously save the shared preferences while holding the current thread until done
and returning a success flag.
boolean result = editor.commit();
```

`apply()` was added in 2.3 (API 9), it commits without returning a boolean indicating success or failure.

`commit()` returns true if the save works, false otherwise.

`apply()` was added as the Android dev team noticed that almost no one took notice of the return value, so apply is faster as it is asynchronous.

Unlike `commit()`, which writes its preferences out to persistent storage synchronously, `apply()` commits its changes to the in-memory `SharedPreferences` immediately but starts an asynchronous commit to disk and you won't be notified of any failures. If another editor on this `SharedPreferences` does a regular `commit()` while a `apply()` is still outstanding, the `commit()` will block until all async commits(apply) are completed as well as any other sync commits that may be pending.

Section 40.3: Read and write values to SharedPreferences

```
public class MyActivity extends Activity {

    private static final String PREFS_FILE = "NameOfYourPreferenceFile";
    // PREFS_MODE defines which apps can access the file
    private static final int PREFS_MODE = Context.MODE_PRIVATE;
    // you can use live template "key" for quickly creating keys
    private static final String KEY_BOOLEAN = "KEY_FOR_YOUR_BOOLEAN";
    private static final String KEY_STRING = "KEY_FOR_YOUR_STRING";
    private static final String KEY_FLOAT = "KEY_FOR_YOUR_FLOAT";
    private static final String KEY_INT = "KEY_FOR_YOUR_INT";
    private static final String KEY_LONG = "KEY_FOR_YOUR_LONG";

    @Override
    protected void onStart() {
        super.onStart();

        // Get the saved flag (or default value if it hasn't been saved yet)
        SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
        // read a boolean value (default false)
        boolean booleanVal = settings.getBoolean(KEY_BOOLEAN, false);
        // read an int value (Default 0)
        int intVal = settings.getInt(KEY_INT, 0);
        // read a string value (default "my string")
        String str = settings.getString(KEY_STRING, "my string");
```

```
        // read a long value (default 123456)
        long longVal = settings.getLong(KEY_LONG, 123456);
        // read a float value (default 3.14f)
        float floatVal = settings.getFloat(KEY_FLOAT, 3.14f);
    }

    @Override
    protected void onStop() {
        super.onStop();

        // Save the flag
        SharedPreferences settings = getSharedPreferences(PREFS_FILE, PREFS_MODE);
        SharedPreferences.Editor editor = settings.edit();
        // write a boolean value
        editor.putBoolean(KEY_BOOLEAN, true);
        // write an integer value
        editor.putInt(KEY_INT, 123);
        // write a string
        editor.putString(KEY_STRING, "string value");
        // write a long value
        editor.putLong(KEY_LONG, 456876451);
        // write a float value
        editor.putFloat(KEY_FLOAT, 1.51f);
        editor.apply();
    }
}
```

`getSharedPreferences()` is a method from the `Context` class — which `Activity` extends. If you need to access the `getSharedPreferences()` method from other classes, you can use `context.getSharedPreferences()` with a `Context` Object reference from an `Activity`, `View`, or `Application`.

Section 40.4: Retrieve all stored entries from a particular SharedPreferences file

The `getAll()` method retrieves all values from the preferences. We can use it, for instance, to log the current content of the `SharedPreferences`:

```
private static final String PREFS_FILE = "MyPrefs";

public static void logSharedPreferences(final Context context) {
    SharedPreferences sharedPreferences = context.getSharedPreferences(PREFS_FILE,
Context.MODE_PRIVATE);
    Map<String, ?> allEntries = sharedPreferences.getAll();
    for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
        final String key = entry.getKey();
        final Object value = entry.getValue();
        Log.d("map values", key + ": " + value);
    }
}
```

The documentation warns you about modifying the `Collection` returned by `getAll`:

> Note that you must not modify the collection returned by this method, or alter any of its contents. The consistency of your stored data is not guaranteed if you do.

Section 40.5: Reading and writing data to SharedPreferences with Singleton

SharedPreferences Manager (Singleton) class to read and write all types of data.

```
import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;

import com.google.gson.Gson;

import java.lang.reflect.Type;

/**
 * Singleton Class for accessing SharedPreferences,
 * should be initialized once in the beginning by any application component using static
 * method initialize(applicationContext)
 */
public class SharedPrefsManager {

    private static final String TAG = SharedPrefsManager.class.getName();
    private SharedPreferences prefs;
    private static SharedPrefsManager uniqueInstance;
    public static final String PREF_NAME = "com.example.app";

    private SharedPrefsManager(Context appContext) {
        prefs = appContext.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    }

    /**
     * Throws IllegalStateException if this class is not initialized
     *
     * @return unique SharedPrefsManager instance
     */
    public static SharedPrefsManager getInstance() {
        if (uniqueInstance == null) {
            throw new IllegalStateException(
                    "SharedPrefsManager is not initialized, call initialize(applicationContext) " +
                            "static method first");
        }
        return uniqueInstance;
    }

    /**
     * Initialize this class using application Context,
     * should be called once in the beginning by any application Component
     *
     * @param appContext application context
     */
    public static void initialize(Context appContext) {
        if (appContext == null) {
            throw new NullPointerException("Provided application context is null");
        }
        if (uniqueInstance == null) {
            synchronized (SharedPrefsManager.class) {
                if (uniqueInstance == null) {
                    uniqueInstance = new SharedPrefsManager(appContext);
                }
            }
        }
    }
```

```
    private SharedPreferences getPrefs() {
        return prefs;
    }

    /**
     * Clears all data in SharedPreferences
     */
    public void clearPrefs() {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.clear();
        editor.commit();
    }

    public void removeKey(String key) {
        getPrefs().edit().remove(key).commit();
    }

    public boolean containsKey(String key) {
        return getPrefs().contains(key);
    }

    public String getString(String key, String defValue) {
        return getPrefs().getString(key, defValue);
    }

    public String getString(String key) {
        return getString(key, null);
    }

    public void setString(String key, String value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putString(key, value);
        editor.apply();
    }

    public int getInt(String key, int defValue) {
        return getPrefs().getInt(key, defValue);
    }

    public int getInt(String key) {
        return getInt(key, 0);
    }

    public void setInt(String key, int value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putInt(key, value);
        editor.apply();
    }

    public long getLong(String key, long defValue) {
        return getPrefs().getLong(key, defValue);
    }

    public long getLong(String key) {
        return getLong(key, 0L);
    }

    public void setLong(String key, long value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putLong(key, value);
        editor.apply();
    }
```

```
    public boolean getBoolean(String key, boolean defValue) {
        return getPrefs().getBoolean(key, defValue);
    }

    public boolean getBoolean(String key) {
        return getBoolean(key, false);
    }

    public void setBoolean(String key, boolean value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putBoolean(key, value);
        editor.apply();
    }

    public boolean getFloat(String key) {
        return getFloat(key, 0f);
    }

    public boolean getFloat(String key, float defValue) {
        return getFloat(key, defValue);
    }

    public void setFloat(String key, Float value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putFloat(key, value);
        editor.apply();
    }

    /**
     * Persists an Object in prefs at the specified key, class of given Object must implement Model
     * interface
     *
     * @param key         String
     * @param modelObject Object to persist
     * @param <M>         Generic for Object
     */
    public <M extends Model> void setObject(String key, M modelObject) {
        String value = createJSONStringFromObject(modelObject);
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putString(key, value);
        editor.apply();
    }

    /**
     * Fetches the previously stored Object of given Class from prefs
     *
     * @param key                String
     * @param classOfModelObject Class of persisted Object
     * @param <M>                Generic for Object
     * @return Object of given class
     */
    public <M extends Model> M getObject(String key, Class<M> classOfModelObject) {
        String jsonData = getPrefs().getString(key, null);
        if (null != jsonData) {
            try {
                Gson gson = new Gson();
                M customObject = gson.fromJson(jsonData, classOfModelObject);
                return customObject;
            } catch (ClassCastException cce) {
                Log.d(TAG, "Cannot convert string obtained from prefs into collection of type " +
                        classOfModelObject.getName() + "\n" + cce.getMessage());
            }
```

```
        }
        return null;
    }

    /**
     * Persists a Collection object in prefs at the specified key
     *
     * @param key            String
     * @param dataCollection Collection Object
     * @param <C>            Generic for Collection object
     */
    public <C> void setCollection(String key, C dataCollection) {
        SharedPreferences.Editor editor = getPrefs().edit();
        String value = createJSONStringFromObject(dataCollection);
        editor.putString(key, value);
        editor.apply();
    }

    /**
     * Fetches the previously stored Collection Object of given type from prefs
     *
     * @param key     String
     * @param typeOfC Type of Collection Object
     * @param <C>     Generic for Collection Object
     * @return Collection Object which can be casted
     */
    public <C> C getCollection(String key, Type typeOfC) {
        String jsonData = getPrefs().getString(key, null);
        if (null != jsonData) {
            try {
                Gson gson = new Gson();
                C arrFromPrefs = gson.fromJson(jsonData, typeOfC);
                return arrFromPrefs;
            } catch (ClassCastException cce) {
                Log.d(TAG, "Cannot convert string obtained from prefs into collection of type " +
                        typeOfC.toString() + "\n" + cce.getMessage());
            }
        }
        return null;
    }

    public void registerPrefsListener(SharedPreferences.OnSharedPreferenceChangeListener listener)
{
        getPrefs().registerOnSharedPreferenceChangeListener(listener);
    }

    public void unregisterPrefsListener(

            SharedPreferences.OnSharedPreferenceChangeListener listener) {
        getPrefs().unregisterOnSharedPreferenceChangeListener(listener);
    }

    public SharedPreferences.Editor getEditor() {
        return getPrefs().edit();
    }

    private static String createJSONStringFromObject(Object object) {
        Gson gson = new Gson();
        return gson.toJson(object);
    }
}
```

`Model` **`interface`** which is implemented by classes going to `Gson` to avoid proguard obfuscation.

```
public interface Model {

}
```

Proguard rules for `Model` interface:

```
-keep interface com.example.app.Model
-keep class * implements com.example.app.Model { *;}
```

Section 40.6: getPreferences(int) VS getSharedPreferences(String, int)

`getPreferences(int)`

returns the preferences saved by `Activity's class name` as described in the docs :

> Retrieve a SharedPreferences object for accessing preferences that are private to this activity. This simply calls the underlying getSharedPreferences(String, int) method by passing in this activity's class name as the preferences name.

While using getSharedPreferences (String name, int mode) method returns the prefs saved under the given `name`. As in the docs :

> Retrieve and hold the contents of the preferences file 'name', returning a SharedPreferences through which you can retrieve and modify its values.

So if the value being saved in the `SharedPreferences` has to be used across the app, one should use `getSharedPreferences (String name, int mode)` with a fixed name. As, using `getPreferences(int)` returns/saves the preferences belonging to the `Activity` calling it.

Section 40.7: Listening for SharedPreferences changes

```
SharedPreferences sharedPreferences = ...;
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);


private final SharedPreferences.OnSharedPreferenceChangeListener mOnSharedPreferenceChangeListener
= new SharedPreferences.OnSharedPreferenceChangeListener() {
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        //TODO
    }
}
```

Please note:

- The listener will fire only if value was added or changed, setting the same value won't call it;
- The listener needs to be saved in a member variable and **NOT** with an anonymous class, because `registerOnSharedPreferenceChangeListener` stores it with a weak reference, so it would be garbage collected;

- Instead of using a member variable, it can also be directly implemented by the class and then call `registerOnSharedPreferenceChangeListener(this);`
- Remember to unregister the listener when it is no more required using `unregisterOnSharedPreferenceChangeListener`.

Section 40.8: Store, Retrieve, Remove and Clear Data from SharedPreferences

Create SharedPreferences BuyyaPref

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref", MODE_PRIVATE);
Editor editor = pref.edit();
```

Storing data as KEY/VALUE pair

```
editor.putBoolean("key_name1", true);           // Saving boolean - true/false
editor.putInt("key_name2", 10);         // Saving integer
editor.putFloat("key_name3", 10.1f);    // Saving float
editor.putLong("key_name4", 1000);      // Saving long
editor.putString("key_name5", "MyString");  // Saving string

// Save the changes in SharedPreferences
editor.commit(); // commit changes
```

Get SharedPreferences data

If value for key not exist then return second param value(In this case null, this is like default value)

```
pref.getBoolean("key_name1", null);         // getting boolean
pref.getInt("key_name2", null);             // getting Integer
pref.getFloat("key_name3", null);           // getting Float
pref.getLong("key_name4", null);            // getting Long
pref.getString("key_name5", null);          // getting String
```

Deleting Key value from SharedPreferences

```
editor.remove("key_name3"); // will delete key key_name3
editor.remove("key_name4"); // will delete key key_name4

// Save the changes in SharedPreferences
editor.commit(); // commit changes
```

Clear all data from SharedPreferences

```
 editor.clear();
 editor.commit(); // commit changes
```

Section 40.9: Add filter for EditTextPreference

Create this class :

```
public class InputFilterMinMax implements InputFilter {

    private int min, max;

    public InputFilterMinMax(int min, int max) {
```

```
        this.min = min;
        this.max = max;
    }

    public InputFilterMinMax(String min, String max) {
        this.min = Integer.parseInt(min);
        this.max = Integer.parseInt(max);
    }

    @Override
    public CharSequence filter(CharSequence source, int start, int end, Spanned dest, int dstart,
int dend) {
        try {
            int input = Integer.parseInt(dest.toString() + source.toString());
            if (isInRange(min, max, input))
                return null;
        } catch (NumberFormatException nfe) { }
        return "";
    }

    private boolean isInRange(int a, int b, int c) {
        return b > a ? c >= a && c <= b : c >= b && c <= a;
    }
}
```

Use :

```
EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });
```

Section 40.10: Supported data types in SharedPreferences

`SharedPreferences` allows you to store primitive data types only (**`boolean`**, **`float`**, **`long`**, **`int`**, `String`, and `string set`). You cannot store more complex objects in `SharedPreferences`, and as such is really meant to be a place to store user settings or similar, it's not meant to be a database to keep user data (like saving a todo list a user made for example).

To store something in `SharedPreferences` you use a Key and a Value. The Key is how you can reference what you stored later and the Value data you want to store.

```
    String keyToUseToFindLater = "High Score";
    int newHighScore = 12938;
    //getting SharedPreferences & Editor objects
    SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPref.edit();
    //saving an int in the SharedPreferences file
    editor.putInt(keyToUseToFindLater, newHighScore);
    editor.commit();
```

Section 40.11: Different ways of instantiating an object of SharedPreferences

You can access SharedPreferences in several ways:

Get the default SharedPreferences file:

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Get a specific SharedPreferences file:

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

Get SharedPreferences from another app:

```
// Note that the other app must declare prefs as MODE_WORLD_WRITEABLE
final ArrayList<HashMap<String,String>> LIST = new ArrayList<HashMap<String,String>>();
Context contextOtherApp = createPackageContext("com.otherapp", Context.MODE_WORLD_WRITEABLE);
SharedPreferences prefs = contextOtherApp.getSharedPreferences("pref_file_name",
Context.MODE_WORLD_READABLE);
```

Section 40.12: Removing keys

```
private static final String MY_PREF = "MyPref";

// ...

SharedPreferences prefs = ...;

// ...

SharedPreferences.Editor editor = prefs.edit();
editor.putString(MY_PREF, "value");
editor.remove(MY_PREF);
editor.apply();
```

After the `apply()`, `prefs` contains "key" -> "value", in addition to whatever it contained already. Even though it looks like I added "key" and then removed it, the remove actually happens first. The changes in the `Editor` are all applied in one go, not in the order you added them. All removes happen before all puts.

Section 40.13: Support pre-Honeycomb with StringSet

Here's the utility class:

```
public class SharedPreferencesCompat {

    public static void putStringSet(SharedPreferences.Editor editor, String key, Set<String>
values) {
            if (Build.VERSION.SDK_INT >= 11) {
                while (true) {
                    try {
                        editor.putStringSet(key, values).apply();
                        break;
                    } catch (ClassCastException ex) {
                        // Clear stale JSON string from before system upgrade
                        editor.remove(key);
                    }
                }
            } else putStringSetToJson(editor, key, values);
    }

    public static Set<String> getStringSet(SharedPreferences prefs, String key, Set<String>
defaultReturnValue) {
```

```
        if (Build.VERSION.SDK_INT >= 11) {
            try {
                return prefs.getStringSet(key, defaultReturnValue);
            } catch (ClassCastException ex) {
                // If user upgraded from Gingerbread to something higher read the stale JSON string
                return getStringSetFromJson(prefs, key, defaultReturnValue);
            }
        } else return getStringSetFromJson(prefs, key, defaultReturnValue);
    }

    private static Set<String> getStringSetFromJson(SharedPreferences prefs, String key,
Set<String> defaultReturnValue) {
        final String input = prefs.getString(key, null);
        if (input == null) return defaultReturnValue;

        try {
            HashSet<String> set = new HashSet<>();
            JSONArray json = new JSONArray(input);
            for (int i = 0, size = json.length(); i < size; i++) {
                String value = json.getString(i);
                set.add(value);
            }
            return set;
        } catch (JSONException e) {
            e.printStackTrace();
            return defaultReturnValue;
        }
    }

    private static void putStringSetToJson(SharedPreferences.Editor editor, String key, Set<String>
values) {
        JSONArray json = new JSONArray(values);
        if (Build.VERSION.SDK_INT >= 9)
            editor.putString(key, json.toString()).apply();
        else
            editor.putString(key, json.toString()).commit();
    }

    private SharedPreferencesCompat() {}
}
```

An example to save preferences as StringSet data type is:

```
Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);
```

To retrieve them back:

```
Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());
```

Reference: Android Support Preference

Chapter 41: Intent

Parameter	Details
intent	The intent to start
requestCode	Unique number to identify the request
options	Additional options for how the Activity should be started
name	The name of the extra data
value	The value of the extra data
CHOOSE_CONTACT_REQUEST_CODE	the code of the request, to identify it on `onActivityResult` method
action	Any action to perform via this intent, ex: Intent.ACTION_VIEW
uri	data uri to be used by intent to perform specified action
packageContext	Context to use to initialize the Intent
cls	Class to be used by this intent

An Intent is a small message passed around the Android system. This message may hold information about our intention to perform a task.

It is basically a passive data structure holding an abstract description of an action to be performed.

Section 41.1: Getting a result from another Activity

By using `startActivityForResult(Intent intent, int requestCode)` you can start another `Activity` and then receive a result from that `Activity` in the `onActivityResult(int requestCode, int resultCode, Intent data)` method. The result will be returned as an `Intent`. An intent can contain data via a Bundle

In this example `MainActivity` will start a `DetailActivity` and then expect a result from it. Each request type should have its own `int` request code, so that in the *overridden* `onActivityResult(int requestCode, int resultCode, Intent data)` method in `MainActivity` , it can be determined which request to process by comparing values of `requestCode` and `REQUEST_CODE_EXAMPLE` (though in this example, there is only one).

MainActivity:

```
public class MainActivity extends Activity {

    // Use a unique request code for each use case
    private static final int REQUEST_CODE_EXAMPLE = 0x9345;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Create a new instance of Intent to start DetailActivity
        final Intent intent = new Intent(this, DetailActivity.class);

        // Start DetailActivity with the request code
        startActivityForResult(intent, REQUEST_CODE_EXAMPLE);
    }

    // onActivityResult only get called
    // when the other Activity previously started using startActivityForResult
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        // First we need to check if the requestCode matches the one we used.
        if(requestCode == REQUEST_CODE_EXAMPLE) {

            // The resultCode is set by the DetailActivity
            // By convention RESULT_OK means that whatever
            // DetailActivity did was executed successfully
            if(resultCode == Activity.RESULT_OK) {
                // Get the result from the returned Intent
                final String result = data.getStringExtra(DetailActivity.EXTRA_DATA);

                // Use the data - in this case, display it in a Toast.
                Toast.makeText(this, "Result: " + result, Toast.LENGTH_LONG).show();
            } else {
                // setResult wasn't successfully executed by DetailActivity
                // Due to some error or flow of control. No data to retrieve.
            }
        }
    }
}
```

DetailActivity:

```
public class DetailActivity extends Activity {

    // Constant used to identify data sent between Activities.
    public static final String EXTRA_DATA = "EXTRA_DATA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);

        final Button button = (Button) findViewById(R.id.button);
        // When this button is clicked we want to return a result
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Create a new Intent object as container for the result
                final Intent data = new Intent();

                // Add the required data to be returned to the MainActivity
                data.putExtra(EXTRA_DATA, "Some interesting data!");

                // Set the resultCode as Activity.RESULT_OK to
                // indicate a success and attach the Intent
                // which contains our result data
                setResult(Activity.RESULT_OK, data);

                // With finish() we close the DetailActivity to
                // return back to MainActivity
                finish();
            }
        });
    }

    @Override
    public void onBackPressed() {
        // When the user hits the back button set the resultCode
        // as Activity.RESULT_CANCELED to indicate a failure
        setResult(Activity.RESULT_CANCELED);
        super.onBackPressed();
    }
```

```
}
```

A few things you need to be aware of:

- Data is only returned once you call `finish()`. You need to call `setResult()` before calling `finish()`, otherwise, no result will be returned.
- Make sure your `Activity` is not using `android:launchMode="singleTask"`, or it will cause the `Activity` to run in a separate task and therefore you will not receive a result from it. If your `Activity` uses `singleTask` as launch mode, it will call `onActivityResult()` immediately with a result code of `Activity.RESULT_CANCELED`.
- Be careful when using `android:launchMode="singleInstance"`. On devices before Lollipop (Android 5.0, API Level 21), Activities will not return a result.
- You can use explicit or implicit intents when you call `startActivityForResult()`. When starting one of your own activities to receive a result, you should use an explicit intent to ensure that you receive the expected result. An explicit `intent` is always delivered to its target, no matter what it contains; the `filter` is not consulted. But an implicit intent is delivered to a component only if it can pass through one of the component's filters.

Section 41.2: Passing data between activities

This example illustrates sending a `String` with value as `"Some data!"` from `OriginActivity` to `DestinationActivity`.

NOTE: This is the most straightforward way of sending data between two activities. See the example on using the starter pattern for a more robust implementation.

OriginActivity

```
public class OriginActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_origin);

        // Create a new Intent object, containing DestinationActivity as target Activity.
        final Intent intent = new Intent(this, DestinationActivity.class);

        // Add data in the form of key/value pairs to the intent object by using putExtra()
        intent.putExtra(DestinationActivity.EXTRA_DATA, "Some data!");

        // Start the target Activity with the intent object
        startActivity(intent);
    }
}
```

DestinationActivity

```
public class DestinationActivity extends AppCompatActivity {

    public static final String EXTRA_DATA = "EXTRA_DATA";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_destination);

        // getIntent() returns the Intent object which was used to start this Activity
        final Intent intent = getIntent();
```

```
        // Retrieve the data from the intent object by using the same key that
        // was previously used to add data to the intent object in OriginActivity.
        final String data = intent.getStringExtra(EXTRA_DATA);
    }
}
```

It is also possible to pass other `primitive` data types as well as `arrays`, `Bundle` and `Parcelable` data. Passing `Serializable` is also possible, but should be avoided as it is more than three times slower than `Parcelable`.

Serializable is a standard Java **`interface`**. You simply mark a class as `Serializable` by implementing the `Serializable` **`interface`** and Java will automatically serialize it during required situations.

Parcelable is an Android specific **`interface`** which can be implemented on custom data types (i.e. your own objects / POJO objects), it allows your object to be flattened and reconstruct itself without the destination needing to do anything. There is a documentation example of making an object parcelable.

Once you have a `parcelable` object you can send it like a primitive type, with an intent object:

```
intent.putExtra(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

Or in a bundle / as an argument for a fragment:

```
bundle.putParcelable(DestinationActivity.EXTRA_DATA, myParcelableObject);
```

and then also read it from the intent at the destination using getParcelableExtra:

```
final MyParcelableType data = intent.getParcelableExtra(EXTRA_DATA);
```

Or when reading in a fragment from a bundle:

```
final MyParcelableType data = bundle.getParcelable(EXTRA_DATA);
```

Once you have a `Serializable` object you can put it in an intent object:

```
bundle.putSerializable(DestinationActivity.EXTRA_DATA, mySerializableObject);
```

and then also read it from the intent object at the destination as shown below:

```
final SerializableType data = (SerializableType)bundle.getSerializable(EXTRA_DATA);
```

Section 41.3: Open a URL in a browser

Opening with the default browser

This example shows how you can open a URL programmatically in the built-in web browser rather than within your application. This allows your app to open up a webpage without the need to include the `INTERNET` permission in your manifest file.

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Uri uri = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    // Verify that the intent will resolve to an activity
    if (intent.resolveActivity(getPackageManager()) != null) {
        // Here we use an intent without a Chooser unlike the next example
```

```
        startActivity(intent);
    }
}
```

Prompting the user to select a browser

Note that this example uses the `Intent.createChooser()` method:

```
public void onBrowseClick(View v) {
    String url = "http://www.google.com";
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
    // Note the Chooser below. If no applications match,
    // Android displays a system message.So here there is no need for try-catch.
    startActivity(Intent.createChooser(intent, "Browse with"));

}
```

In some cases, the URL may start with **"www"**. If that is the case you will get this exception:

> `android.content.ActivityNotFoundException` : No Activity found to handle Intent

The URL must always start with **"http://"** or **"https://"**. Your code should therefore check for it, as shown in the following code snippet:

```
if (!url.startsWith("https://") && !url.startsWith("http://")){
    url = "http://" + url;
}
Intent openUrlIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
if (openUrlIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(openUrlIntent);
}
```

Best Practices

Check if there are no apps on the device that can receive the implicit intent. Otherwise, your app will crash when it calls `startActivity()`. To first verify that an app exists to receive the intent, call `resolveActivity()` on your Intent object. If the result is non-null, there is at least one app that can handle the intent and it's safe to call `startActivity()`. If the result is null, you should not use the intent and, if possible, you should disable the feature that invokes the intent.

Section 41.4: Starter Pattern

This pattern is a more strict approach to starting an `Activity`. Its purpose is to improve code readability, while at the same time decrease code complexity, maintenance costs, and coupling of your components.

The following example implements the starter pattern, which is usually implemented as a static method on the `Activity` itself. This static method accepts all required parameters, constructs a valid `Intent` from that data, and then starts the `Activity`.

An `Intent` is an object that provides runtime binding between separate components, such as two activities. The Intent represents an app's "intent to do something." You can use intents for a wide variety of tasks, but here, your intent starts another activity.

```
public class ExampleActivity extends AppCompatActivity {
```

```
    private static final String EXTRA_DATA = "EXTRA_DATA";

    public static void start(Context context, String data) {
        Intent intent = new Intent(context, ExampleActivity.class);
        intent.putExtra(EXTRA_DATA, data);
        context.startActivity(intent);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Intent intent = getIntent();
        if(!intent.getExtras().containsKey(EXTRA_DATA)){
            throw new UnsupportedOperationException("Activity should be started using the static
start method");
        }
        String data = intent.getStringExtra(EXTRA_DATA);
    }
}
```

This pattern also allows you to force additional data to be passed with the intent.

The `ExampleActivity` can then be started like this, where `context` is an activity context:

```
ExampleActivity.start(context, "Some data!");
```

Section 41.5: Clearing an activity stack

Sometimes you may want to start a new activity while removing previous activities from the back stack, so the back button doesn't take you back to them. One example of this might be starting an app on the Login activity, taking you through to the Main activity of your application, but on logging out you want to be directed back to Login without a chance to go back. In a case like that you can set the `FLAG_ACTIVITY_CLEAR_TOP` flag for the intent, meaning if the activity being launched is already running in the current task (LoginActivity), then instead of launching a new instance of that activity, all of the other activities on top of it will be closed and this Intent will be delivered to the (now on top) old activity as a new Intent.

```
Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

It's also possible to use the flags `FLAG_ACTIVITY_NEW_TASK` along with `FLAG_ACTIVITY_CLEAR_TASK` if you want to clear all Activities on the back stack:

```
Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
// Closing all the Activities, clear the back stack.
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(intent);
```

Section 41.6: Start an activity

This example will start `DestinationActivity` from `OriginActivity`.

Here, the `Intent` constructor takes two parameters:

1. A Context as its first parameter (this is used because the Activity class is a subclass of Context)

2. The Class of the app component to which the system should deliver the Intent (in this case, the activity that should be started)

```
public class OriginActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_origin);

        Intent intent = new Intent(this, DestinationActivity.class);

        startActivity(intent);
        finish(); // Optionally, you can close OriginActivity. In this way when the user press back
from DestinationActivity he/she won't land on OriginActivity again.
    }
}
```

Another way to create the `Intent` to open `DestinationActivity` is to use the default constructor for the `Intent`, and use the `setClass()` method to tell it which Activity to open:

```
Intent i=new Intent();
i.setClass(this, DestinationActivity.class);
startActivity(intent);
finish(); // Optionally, you can close OriginActivity. In this way when the user press back from
DestinationActivity he/she won't land on OriginActivity
```

Section 41.7: Sending emails

```
// Compile a Uri with the 'mailto' schema
Intent emailIntent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts(
        "mailto","johndoe@example.com", null));
// Subject
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Hello World!");
// Body of email
emailIntent.putExtra(Intent.EXTRA_TEXT, "Hi! I am sending you a test email.");
// File attachment
emailIntent.putExtra(Intent.EXTRA_STREAM, attachedFileUri);

// Check if the device has an email client
if (emailIntent.resolveActivity(getPackageManager()) != null) {
     // Prompt the user to select a mail app
     startActivity(Intent.createChooser(emailIntent,"Choose your mail application"));
} else {
    // Inform the user that no email clients are installed or provide an alternative
}
```

This will pre-fill an email in a mail app of the user's choice.

If you need to add an attachment, you can use `Intent.ACTION_SEND` instead of `Intent.ACTION_SENDTO`. For multiple attachments you can use `ACTION_SEND_MULTIPLE`

A word of caution: not every device has a provider for `ACTION_SENDTO`, and calling `startActivity()` without checking with `resolveActivity()` first may throw an ActivityNotFoundException.

Section 41.8: CustomTabsIntent for Chrome Custom Tabs

`Version ≥ 4.0.3`

Using a `CustomTabsIntent`, it is now possible to configure Chrome custom tabs in order to customize key UI components in the browser that is opened from your app.

This is a good alternative to using a WebView for some cases. It allows loading of a web page with an Intent, with the added ability to inject some degree of the look and feel of your app into the browser.

Here is an example of how to open a url using `CustomTabsIntent`

```
String url = "https://www.google.pl/";
CustomTabsIntent intent = new CustomTabsIntent.Builder()
                    .setStartAnimations(getContext(), R.anim.slide_in_right, R.anim.slide_out_left)
                    .setExitAnimations(getContext(), android.R.anim.slide_in_left,
android.R.anim.slide_out_right)
                    .setCloseButtonIcon(BitmapFactory.decodeResource(getResources(),
R.drawable.ic_arrow_back_white_24dp))
                    .setToolbarColor(Color.parseColor("#43A047"))
                    .enableUrlBarHiding()
                    .build();
            intent.launchUrl(getActivity(), Uri.parse(url));
```

Note:

To use custom tabs, you need to add this dependency to your build.gradle

```
compile 'com.android.support:customtabs:24.1.1'
```

Section 41.9: Intent URI

This example shows, how to start intent from browser:

```
<a href="intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end">Start
intent</a>
```

This intent will start app with package `com.sample.test` or will open google play with this package.

Also this intent can be started with javascript:

```
var intent = "intent://host.com/path#Intent;package=com.sample.test;scheme=yourscheme;end";
window.location.replace(intent)
```

In activity this host and path can be obtained from intent data:

```
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Uri data = getIntent().getData(); // returns host.com/path
}
```

Intent URI syntax:

```
HOST/URI-path // Optional host
#Intent;
    package=[string];
    action=[string];
    category=[string];
    component=[string];
    scheme=[string];
```

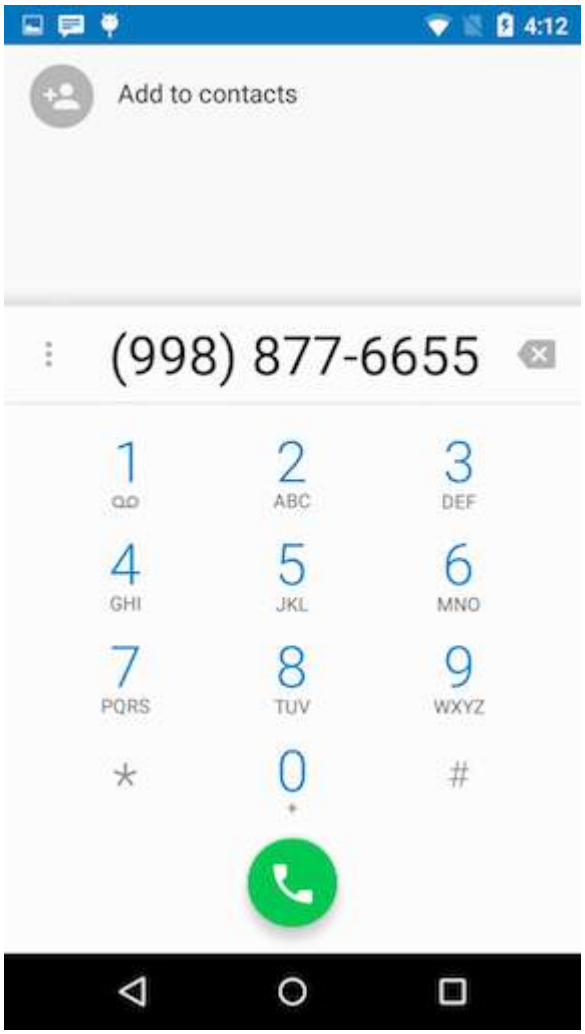
```
end;
```

Section 41.10: Start the dialer

This example shows how to open a default dialer (an app that makes regular calls) with a provided telephone number already in place:

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:9988776655")); //Replace with valid phone number. Remember to add the
tel: prefix, otherwise it will crash.
startActivity(intent);
```

Result from running the code above:



Section 41.11: Broadcasting Messages to Other Components

Intents can be used to broadcast messages to other components of your application (such as a running background service) or to the entire Android system.

To send a broadcast **within your application**, use the `LocalBroadcastManager` class:

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // the intent action
intent.putExtra("key", "value"); // data to be passed with your broadcast

LocalBroadcastManager manager = LocalBroadcastManager.getInstance(context);
manager.sendBroadcast(intent);
```

To send a broadcast to components outside of your application, use the `sendBroadcast()` method on a `Context` object.

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // the intent action
intent.putExtra("key", "value"); // data to be passed with your broadcast

context.sendBroadcast(intent);
```

Information about *receiving* broadcasts can be found here: Broadcast Receiver

Section 41.12: Passing custom object between activities

It is also possible to pass your custom object to other activities using the `Bundle` class.

There are two ways:

- `Serializable` interface—for Java and Android
- `Parcelable` interface—memory efficient, only for Android (recommended)

Parcelable

Parcelable processing is much faster than serializable. One of the reasons for this is that we are being explicit about the serialization process instead of using reflection to infer it. It also stands to reason that the code has been heavily optimized for this purpose.

```
public class MyObjects implements Parcelable {

    private int age;
    private String name;

    private ArrayList<String> address;

    public MyObjects(String name, int age, ArrayList<String> address) {
        this.name = name;
        this.age = age;
        this.address = address;

    }

    public MyObjects(Parcel source) {
        age = source.readInt();
        name = source.readString();
        address = source.createStringArrayList();
    }

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeInt(age);
        dest.writeString(name);
        dest.writeStringList(address);
    }

    public int getAge() {
        return age;
```

```
    }

    public String getName() {
        return name;
    }

    public ArrayList<String> getAddress() {
        if (!(address == null))
            return address;
        else
            return new ArrayList<String>();
    }

    public static final Creator<MyObjects> CREATOR = new Creator<MyObjects>() {
        @Override
        public MyObjects[] newArray(int size) {
            return new MyObjects[size];
        }

        @Override
        public MyObjects createFromParcel(Parcel source) {
            return new MyObjects(source);
        }
    };
}
```

Sending Activity Code

```
MyObject mObject = new MyObject("name","age","Address array here");

//Passing MyOject
Intent mIntent = new Intent(FromActivity.this, ToActivity.class);
mIntent.putExtra("UniqueKey", mObject);
startActivity(mIntent);
```

Receiving the object in destination activity.

```
//Getting MyObjects
Intent mIntent = getIntent();
MyObjects workorder = (MyObjects) mIntent.getParcelable("UniqueKey");
```

You can pass Arraylist of Parceble object as below

```
//Array of MyObjects
ArrayList<MyObject> mUsers;

//Passing MyObject List
Intent mIntent = new Intent(FromActivity.this, ToActivity.class);
mIntent.putParcelableArrayListExtra("UniqueKey", mUsers);
startActivity(mIntent);

//Getting MyObject List
Intent mIntent = getIntent();
ArrayList<MyObjects> mUsers = mIntent.getParcelableArrayList("UniqueKey");
```

> **Note:** There are Android Studio plugins such as this one available to generate Parcelable code

Serializable

Sending Activity Code

```
Product product = new Product();
Bundle bundle = new Bundle();
bundle.putSerializable("product", product);
Intent cartIntent = new Intent(mContext, ShowCartActivity.class);
cartIntent.putExtras(bundle);
mContext.startActivity(cartIntent);
```

Receiving the object in destination activity.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Bundle bundle = this.getIntent().getExtras();
    Product product = null;
    if (bundle != null) {
        product = (Product) bundle.getSerializable("product");
    }
```

`Arraylist` of Serializable object: same as single object passing

Custom object should implement the `Serializable` interface.

Section 41.13: Open Google map with specified latitude, longitude

You can pass latitude, longitude from your app to Google map using Intent

```
String uri = String.format(Locale.ENGLISH, "http://maps.google.com/maps?q=loc:%f,%f",
28.43242324,77.8977673);
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
startActivity(intent);
```

Section 41.14: Passing different data through Intent in Activity

1. Passing integer data:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("intVariableName", intValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
int intValue = mIntent.getIntExtra("intVariableName", 0); // set 0 as the default value if no value
for intVariableName found
```

2. Passing double data:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("doubleVariableName", doubleValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
double doubleValue = mIntent.getDoubleExtra("doubleVariableName", 0.00); // set 0.00 as the default
value if no value for doubleVariableName found
```

3. Passing String data:

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("stringVariableName", stringValue);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
String stringValue = mIntent.getExtras().getString("stringVariableName");
```

or

```
Intent mIntent = getIntent();
String stringValue = mIntent.getStringExtra("stringVariableName");
```

4. Passing ArrayList data :

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putStringArrayListExtra("arrayListVariableName", arrayList);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
arrayList = mIntent.getStringArrayListExtra("arrayListVariableName");
```

5. Passing Object data :

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("ObjectVariableName", yourObject);
startActivity(myIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
yourObj = mIntent.getSerializableExtra("ObjectVariableName");
```

> **Note :** Keep in mind your custom Class must implement the `Serializable` interface.

6. Passing HashMap<String, String> data :

SenderActivity

HashMap<String, String> hashMap;

```
Intent mIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
mIntent.putExtra("hashMap", hashMap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
HashMap<String, String> hashMap = (HashMap<String, String>)
mIntent.getSerializableExtra("hashMap");
```

7. Passing Bitmap data :

SenderActivity

```
Intent myIntent = new Intent(SenderActivity.this, ReceiverActivity.class);
myIntent.putExtra("image",bitmap);
startActivity(mIntent);
```

ReceiverActivity

```
Intent mIntent = getIntent();
Bitmap bitmap = mIntent.getParcelableExtra("image");
```

Section 41.15: Share intent

Share simple information with differents apps.

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "This is my text to send.");
sendIntent.setType("text/plain");
startActivity(Intent.createChooser(sendIntent, getResources().getText(R.string.send_to)));
```

Share an image with differents apps.

```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

Section 41.16: Showing a File Chooser and Reading the Result

Starting a File Chooser Activity

```
public void showFileChooser() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);

    // Update with mime types
    intent.setType("*/*");

    // Update with additional mime types here using a String[].
    intent.putExtra(Intent.EXTRA_MIME_TYPES, mimeTypes);
```

```
    // Only pick openable and local files. Theoretically we could pull files from google drive
    // or other applications that have networked files, but that's unnecessary for this example.
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    intent.putExtra(Intent.EXTRA_LOCAL_ONLY, true);

    // REQUEST_CODE = <some-integer>
    startActivityForResult(intent, REQUEST_CODE);
}
```

Reading the Result

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // If the user doesn't pick a file just return
    if (requestCode != REQUEST_CODE || resultCode != RESULT_OK) {
        return;
    }

    // Import the file
    importFile(data.getData());
}

public void importFile(Uri uri) {
    String fileName = getFileName(uri);

    // The temp file could be whatever you want
    File fileCopy = copyToTempFile(uri, File tempFile)

    // Done!
}

/**
 * Obtains the file name for a URI using content resolvers. Taken from the following link
 * https://developer.android.com/training/secure-file-sharing/retrieve-info.html#RetrieveFileInfo
 *
 * @param uri a uri to query
 * @return the file name with no path
 * @throws IllegalArgumentException if the query is null, empty, or the column doesn't exist
 */
private String getFileName(Uri uri) throws IllegalArgumentException {
    // Obtain a cursor with information regarding this uri
    Cursor cursor = getContentResolver().query(uri, null, null, null, null);

    if (cursor.getCount() <= 0) {
        cursor.close();
        throw new IllegalArgumentException("Can't obtain file name, cursor is empty");
    }

    cursor.moveToFirst();

    String fileName = cursor.getString(cursor.getColumnIndexOrThrow(OpenableColumns.DISPLAY_NAME));

    cursor.close();

    return fileName;
}

/**
 * Copies a uri reference to a temporary file
 *
 * @param uri      the uri used as the input stream
 * @param tempFile the file used as an output stream
```

```
 * @return the input tempFile for convenience
 * @throws IOException if an error occurs
 */
private File copyToTempFile(Uri uri, File tempFile) throws IOException {
    // Obtain an input stream from the uri
    InputStream inputStream = getContentResolver().openInputStream(uri);

    if (inputStream == null) {
        throw new IOException("Unable to obtain input stream from URI");
    }

    // Copy the stream to the temp file
    FileUtils.copyInputStreamToFile(inputStream, tempFile);

    return tempFile;
}
```

Section 41.17: Sharing Multiple Files through Intent

The String List passed as a parameter to the `share()` method contains the paths of all the files you want to share.

It basically loops through the paths, adds them to Uri, and starts the Activity which can accept Files of this type.

```
  public static void share(AppCompatActivity context,List<String> paths) {

        if (paths == null || paths.size() == 0) {
            return;
        }
        ArrayList<Uri> uris = new ArrayList<>();
        Intent intent = new Intent();
        intent.setAction(android.content.Intent.ACTION_SEND_MULTIPLE);
        intent.setType("*/*");
        for (String path : paths) {
                File file = new File(path);
                uris.add(Uri.fromFile(file));
        }
        intent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, uris);
        context.startActivity(intent);
    }
```

Section 41.18: Start Unbound Service using an Intent

A Service is a component which runs in the background (on the UI thread) without direct interaction with the user. An unbound Service is just started, and is not bound to the lifecycle of any Activity.

To start a Service you can do as shown in the example below:

```
// This Intent will be used to start the service
Intent i= new Intent(context, ServiceName.class);
// potentially add data to the intent extras
i.putExtra("KEY1", "Value to be used by the service");
context.startService(i);
```

You can use any extras from the intent by using an `onStartCommand()` override:

```
public class MyService extends Service {
    public MyService() {
    }
```

```
    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
    {
        if (intent != null) {
            Bundle extras = intent.getExtras();
            String key1 = extras.getString("KEY1", "");
            if (key1.equals("Value to be used by the service")) {
                //do something
            }
        }
        return START_STICKY;
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

}
```

Section 41.19: Getting a result from Activity to Fragment

Like Getting a result from another Activity you need to call the `Fragment`'s method `startActivityForResult(Intent intent, int requestCode)`. note that you should not call `getActivity().startActivityForResult()` as this will take the result back to the `Fragment`'s parent `Activity`.

Receiving the result can be done using the `Fragment`'s method `onActivityResult()`. You need to make sure that the Fragment's parent Activity also overrides `onActivityResult()` and calls it's **`super`** implementation.

In the following example `ActivityOne` contains `FragmentOne`, which will start `ActivityTwo` and expect a result from it.

ActivityOne

```
public class ActivityOne extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one);
    }

    // You must override this method as the second Activity will always send its results to this
Activity and then to the Fragment
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
       super.onActivityResult(requestCode, resultCode, data);
    }
}
```

activity_one.xml

```
<fragment android:name="com.example.FragmentOne"
   android:id="@+id/fragment_one"
   android:layout_width="match_parent"
   android:layout_height="match_parent" />
```

FragmentOne

```
public class FragmentOne extends Fragment {
    public static final int REQUEST_CODE = 11;
    public static final int RESULT_CODE = 12;
    public static final String EXTRA_KEY_TEST = "testKey";

    // Initializing and starting the second Activity
    private void startSecondActivity() {
        Intent intent = new Intent(getActivity(), ActivityTwo.class);
        startActivityForResult(REQUEST_CODE, intent);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE && resultCode == RESULT_CODE) {
            String testResult = data.getStringExtra(EXTRA_KEY_TEST);
            // TODO: Do something with your extra data
        }
    }
}
```

ActivityTwo

```
public class ActivityTwo extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_two);
    }

    private void closeActivity() {
        Intent intent = new Intent();
        intent.putExtra(FragmentOne.EXTRA_KEY_TEST, "Testing passing data back to ActivityOne");
        setResult(FragmentOne.RESULT_CODE, intent); // You can also send result without any data
using setResult(int resultCode)
        finish();
    }
}
```

Chapter 42: Fragments

Introduction about Fragments and their intercommunication mechanism

Section 42.1: Pass data from Activity to Fragment using Bundle

All fragments should have an empty constructor (i.e. a constructor method having no input arguments). Therefore, in order to pass your data to the Fragment being created, you should use the `setArguments()` method. This methods gets a bundle, which you store your data in, and stores the Bundle in the arguments. Subsequently, this Bundle can then be retrieved in `onCreate()` and `onCreateView()` call backs of the Fragment.

Activity:

```
Bundle bundle = new Bundle();
String myMessage = "Stack Overflow is cool!";
bundle.putString("message", myMessage );
FragmentClass fragInfo = new FragmentClass();
fragInfo.setArguments(bundle);
transaction.replace(R.id.fragment_single, fragInfo);
transaction.commit();
```

Fragment:

```
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
{
    String myValue = this.getArguments().getString("message");
    ...
 }
```

Section 42.2: The newInstance() pattern

Although it is possible to create a fragment constructor with parameters, Android internally calls the zero-argument constructor when recreating fragments (for example, if they are being restored after being killed for Android's own reasons). For this reason, it is not advisable to rely on a constructor that has parameters.

To ensure that your expected fragment arguments are always present you can use a static `newInstance()` method to create the fragment, and put whatever parameters you want in to a bundle that will be available when creating a new instance.

```
import android.os.Bundle;
import android.support.v4.app.Fragment;

public class MyFragment extends Fragment
{
  // Our identifier for obtaining the name from arguments
  private static final String NAME_ARG = "name";

  private String mName;

  // Required
  public MyFragment(){}

  // The static constructor.  This is the only way that you should instantiate
  // the fragment yourself
```

```
  public static MyFragment newInstance(final String name) {
    final MyFragment myFragment = new MyFragment();
    // The 1 below is an optimization, being the number of arguments that will
    // be added to this bundle.  If you know the number of arguments you will add
    // to the bundle it stops additional allocations of the backing map.  If
    // unsure, you can construct Bundle without any arguments
    final Bundle args = new Bundle(1);

    // This stores the argument as an argument in the bundle.  Note that even if
    // the 'name' parameter is NULL then this will work, so you should consider
    // at this point if the parameter is mandatory and if so check for NULL and
    // throw an appropriate error if so
    args.putString(NAME_ARG, name);

    myFragment.setArguments(args);
    return myFragment;
  }

  @Override
  public void onCreate(final Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    final Bundle arguments = getArguments();
    if (arguments == null || !arguments.containsKey(NAME_ARG)) {
      // Set a default or error as you see fit
    } else {
      mName = arguments.getString(NAME_ARG);
    }
  }
}
```

Now, in the Activity:

```
FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
MyFragment mFragment = MyFragment.newInstance("my name");
ft.replace(R.id.placeholder, mFragment);
//R.id.placeholder is where we want to load our fragment
ft.commit();
```

This pattern is a best practice to ensure that all the needed arguments will be passed to fragments on creation. Note that when the system destroys the fragment and re-creates it later, it will automatically restore its state - but you must provide it with an `onSaveInstanceState(Bundle)` implementation.

Section 42.3: Navigation between fragments using backstack and static fabric pattern

First of all, we need to add our first `Fragment` at the beginning, we should do it in the `onCreate()` method of our Activity:

```
if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
      .addToBackStack("fragmentA")
      .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
      .commit();
}
```

Next, we need to manage our backstack. The easiest way is using a function added in our activity that is used for all FragmentTransactions.

```
public void replaceFragment(Fragment fragment, String tag) {
    //Get current fragment placed in container
    Fragment currentFragment = getSupportFragmentManager().findFragmentById(R.id.container);

    //Prevent adding same fragment on top
    if (currentFragment.getClass() == fragment.getClass()) {
        return;
    }

    //If fragment is already on stack, we can pop back stack to prevent stack infinite growth
    if (getSupportFragmentManager().findFragmentByTag(tag) != null) {
        getSupportFragmentManager().popBackStack(tag, FragmentManager.POP_BACK_STACK_INCLUSIVE);
    }

    //Otherwise, just replace fragment
    getSupportFragmentManager()
            .beginTransaction()
            .addToBackStack(tag)
            .replace(R.id.container, fragment, tag)
            .commit();
}
```

Finally, we should override `onBackPressed()` to exit the application when going back from the last Fragment available in the backstack.

```
@Override
public void onBackPressed() {
    int fragmentsInStack = getSupportFragmentManager().getBackStackEntryCount();
    if (fragmentsInStack > 1) { // If we have more than one fragment, pop back stack
        getSupportFragmentManager().popBackStack();
    } else if (fragmentsInStack == 1) { // Finish activity, if only one fragment left, to prevent
leaving empty screen
        finish();
    } else {
        super.onBackPressed();
    }
}
```

Execution in activity:

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Execution outside activity (assuming `MainActivity` is our activity):

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Section 42.4: Sending events back to an activity with callback interface

If you need to send events from fragment to activity, one of the possible solutions is to define callback interface and require that the host activity implement it.

Example

Send callback to an activity, when fragment's button clicked

First of all, define callback interface:

```
public interface SampleCallback {
    void onButtonClicked();
```

```
}
```

Next step is to assign this callback in fragment:

```
public final class SampleFragment extends Fragment {

    private SampleCallback callback;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof SampleCallback) {
            callback = (SampleCallback) context;
        } else {
            throw new RuntimeException(context.toString()
                    + " must implement SampleCallback");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        callback = null;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        final View view = inflater.inflate(R.layout.sample, container, false);
        // Add button's click listener
        view.findViewById(R.id.actionButton).setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                callback.onButtonClicked(); // Invoke callback here
            }
        });
        return view;
    }
}
```

And finally, implement callback in activity:

```
public final class SampleActivity extends Activity implements SampleCallback {

    // ... Skipped code with settings content view and presenting the fragment

    @Override
    public void onButtonClicked() {
        // Invoked when fragment's button has been clicked
    }
}
```

Section 42.5: Animate the transition between fragments

To animate the transition between fragments, or to animate the process of showing or hiding a fragment you use the `FragmentManager` to create a `FragmentTransaction`.

For a single `FragmentTransaction`, there are two different ways to perform animations: you can use a standard animation or you can supply your own custom animations.

Standard animations are specified by calling `FragmentTransaction.setTransition(int transit)`, and using one of the pre-defined constants available in the `FragmentTransaction` class. At the time of writing, these constants are:

```
FragmentTransaction.TRANSIT_NONE
FragmentTransaction.TRANSIT_FRAGMENT_OPEN
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

The complete transaction might look something like this:

```
getSupportFragmentManager()
    .beginTransaction()
    .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Custom animations are specified by calling either `FragmentTransaction.setCustomAnimations(int enter, int exit)` or `FragmentTransaction.setCustomAnimations(int enter, int exit, int popEnter, int popExit)`.

The `enter` and `exit` animations will be played for `FragmentTransactions` that do not involve popping fragments off of the back stack. The `popEnter` and `popExit` animations will be played when popping a fragment off of the back stack.

The following code shows how you would replace a fragment by sliding out one fragment and sliding the other one in it's place.

```
getSupportFragmentManager()
    .beginTransaction()
    .setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

The XML animation definitions would use the `objectAnimator` tag. An example of **slide_in_left.xml** might look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<set>
  <objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
   android:propertyName="x"
   android:valueType="floatType"
   android:valueFrom="-1280"
   android:valueTo="0"
   android:duration="500"/>
</set>
```

Section 42.6: Communication between Fragments

All communications between Fragments must go via an Activity. Fragments **CANNOT** communicate with each other without an Activity.

Additional Resources

- How to implement OnFragmentInteractionListener
- Android | Communicating With Other Fragments

In this sample, we have a `MainActivity` that hosts two fragments, `SenderFragment` and `ReceiverFragment`, for

sending and receiving a `message` (a simple String in this case) respectively.

A Button in `SenderFragment` initiates the process of sending the message. A TextView in the `ReceiverFragment` is updated when the message is received by it.

Following is the snippet for the MainActivity with comments explaining the important lines of code:

```
// Our MainActivity implements the interface defined by the SenderFragment. This enables
// communication from the fragment to the activity
public class MainActivity extends AppCompatActivity implements SenderFragment.SendMessageListener {

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}

/**
 * This method is called when we click on the button in the SenderFragment
 * @param message The message sent by the SenderFragment
 */
@Override
public void onSendMessage(String message) {
    // Find our ReceiverFragment using the SupportFragmentManager and the fragment's id
    ReceiverFragment receiverFragment = (ReceiverFragment)
            getSupportFragmentManager().findFragmentById(R.id.fragment_receiver);

    // Make sure that such a fragment exists
    if (receiverFragment != null) {
        // Send this message to the ReceiverFragment by calling its public method
        receiverFragment.showMessage(message);
    }
}
}
```

The layout file for the `MainActivity` hosts two fragments inside a LinearLayout :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.naru.fragmentcommunication.MainActivity">

<fragment
   android:id="@+id/fragment_sender"
   android:name="com.naru.fragmentcommunication.SenderFragment"
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="1"
   tools:layout="@layout/fragment_sender" />

<fragment
   android:id="@+id/fragment_receiver"
   android:name="com.naru.fragmentcommunication.ReceiverFragment"
```

```
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    tools:layout="@layout/fragment_receiver" />
</LinearLayout>
```

The `SenderFragment` exposes an interface `SendMessageListener` that helps the `MainActivity` know when Button in the `SenderFragment` was clicked.

Following is the code snippet for the `SenderFragment` explaining the important lines of code:

```
public class SenderFragment extends Fragment {

private SendMessageListener commander;

/**
 * This interface is created to communicate between the activity and the fragment. Any activity
 * which implements this interface will be able to receive the message that is sent by this
 * fragment.
 */
public interface SendMessageListener {
    void onSendMessage(String message);
}

/**
 * API LEVEL >= 23
 * <p>
 * This method is called when the fragment is attached to the activity. This method here will
 * help us to initialize our reference variable, 'commander' , for our interface
 * 'SendMessageListener'
 *
 * @param context
 */
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    // Try to cast the context to our interface SendMessageListener i.e. check whether the
    // activity implements the SendMessageListener. If not a ClassCastException is thrown.
    try {
        commander = (SendMessageListener) context;
    } catch (ClassCastException e) {
        throw new ClassCastException(context.toString()
                + "must implement the SendMessageListener interface");
    }
}

/**
 * API LEVEL < 23
 * <p>
 * This method is called when the fragment is attached to the activity. This method here will
 * help us to initialize our reference variable, 'commander' , for our interface
 * 'SendMessageListener'
 *
 * @param activity
 */
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    // Try to cast the context to our interface SendMessageListener i.e. check whether the
    // activity implements the SendMessageListener. If not a ClassCastException is thrown.
    try {
```

```
        commander = (SendMessageListener) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString()
                + "must implement the SendMessageListener interface");
    }
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {
    // Inflate view for the sender fragment.
    View view = inflater.inflate(R.layout.fragment_receiver, container, false);

    // Initialize button and a click listener on it
    Button send = (Button) view.findViewById(R.id.bSend);
    send.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            // Sanity check whether we were able to properly initialize our interface reference
            if (commander != null) {

                // Call our interface method. This enables us to call the implemented method
                // in the activity, from where we can send the message to the ReceiverFragment.
                commander.onSendMessage("HELLO FROM SENDER FRAGMENT!");
            }
        }
    });

    return view;
}
}
```

The layout file for the `SenderFragment`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:gravity="center"
   android:orientation="vertical">

<Button
   android:id="@+id/bSend"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="SEND"
   android:layout_gravity="center_horizontal" />
</LinearLayout>
```

The `ReceiverFragment` is simple and exposes a simple public method to updates its TextView. When the `MainActivity` receives the message from the `SenderFragment` it calls this public method of the `ReceiverFragment`

Following is the code snippet for the `ReceiverFragment` with comments explaining the important lines of code :

```
public class ReceiverFragment extends Fragment {
TextView tvMessage;

@Nullable
```

```
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {
    // Inflate view for the sender fragment.
    View view = inflater.inflate(R.layout.fragment_receiver, container, false);

    // Initialize the TextView
    tvMessage = (TextView) view.findViewById(R.id.tvReceivedMessage);

    return view;
}


/**
 * Method that is called by the MainActivity when it receives a message from the SenderFragment.
 * This method helps update the text in the TextView to the message sent by the SenderFragment.
 * @param message Message sent by the SenderFragment via the MainActivity.
 */
public void showMessage(String message) {
    tvMessage.setText(message);
}
}
```

The layout file for the `ReceiverFragment` :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:gravity="center"
   android:orientation="vertical">
<TextView
   android:id="@+id/tvReceivedMessage"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Waiting for message!" />
</LinearLayout>
```

Chapter 43: Button

Section 43.1: Using the same click event for one or more Views in the XML

When we create any View in layout, we can use the android:onClick attribute to reference a method in the associated activity or fragment to handle the click events.

XML Layout

```
<Button android:id="@+id/button"
    ...
    // onClick should reference the method in your activity or fragment
    android:onClick="doSomething" />

// Note that this works with any class which is a subclass of View, not just Button
<ImageView android:id="@+id/image"
    ...
    android:onClick="doSomething" />
```

Activity/fragment code

In your **code**, create the method you named, where v will be the view that was touched, and do something for each view that calls this method.

```
public void doSomething(View v) {
    switch(v.getId()) {
        case R.id.button:
            // Button was clicked, do something.
            break;
        case R.id.image:
            // Image was clicked, do something else.
            break;
    }
}
```

If you want, you can also use different method for each View (in this case, of course, you don't have to check for the ID).

Section 43.2: Defining external Listener

When should I use it

- When the code inside an inline listener is too big and your method / class becomes ugly and hard to read
- You want to perform same action in various elements (view) of your app

To achieve this you need to create a class implementing one of the listeners in the View API.

For example, give help when long click on any element:

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
```

```
    public void onLongClick(View v) {
        // show help toast or popup
    }
}
```

Then you just need to have an attribute or variable in your `Activity` to use it:

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);

button1.setOnClickListener(helpListener);
button2.setOnClickListener(helpListener);
label.setOnClickListener(helpListener);
button1.setOnClickListener(helpListener);
```

NOTE: defining listeners in separated class has one disadvantage, it cannot access class fields directly, so you need to pass data (context, view) through constructor unless you make attributes public or define geters.

Section 43.3: inline onClickListener

Say we have a button (we can create it programmatically, or bind it from a view using findViewbyId(), etc...)

```
Button btnOK = (...)
```

Now, create an anonymous class and set it inline.

```
btnOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Do stuff here...
    }
});
```

Section 43.4: Customizing Button style

There are many possible ways of customizing the look of a Button. This example presents several options:

Option 0: Use ThemeOverlay (currently the easiest/quickest way)

Create a new style in your styles file:

styles.xml

```
<resources>
    <style name="mybutton" parent="ThemeOverlay.AppCompat.Ligth">
        <!-- customize colorButtonNormal for the disable color -->
        <item name="colorButtonNormal">@color/colorbuttonnormal</item>
        <!-- customize colorAccent for the enabled color -->
        <item name="colorButtonNormal">@color/coloraccent</item>
    </style>
</resources>
```

Then in the layout where you place your button (e.g. MainActivity):

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:theme="@style/mybutton"
                style="@style/Widget.AppCompat.Button.Colored"/>

</LinearLayout>
```

Option 1: Create your own button style

In values/styles.xml, create a new style for your button:

styles.xml

```
<resources>
        <style name="mybuttonstyle" parent="@android:style/Widget.Button">
            <item name="android:gravity">center_vertical|center_horizontal</item>
            <item name="android:textColor">#FFFFFFFF</item>
            <item name="android:shadowColor">#FF000000</item>
            <item name="android:shadowDx">0</item>
            <item name="android:shadowDy">-1</item>
            <item name="android:shadowRadius">0.2</item>
            <item name="android:textSize">16dip</item>
            <item name="android:textStyle">bold</item>
            <item name="android:background">@drawable/button</item>
        </style>
    </resources>
```

Then in the layout where you place your button (e.g. in MainActivity):

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">
```

```
            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:theme="@style/mybuttonstyle"/>

</LinearLayout>
```

Option 2: Assign a drawable for each of your button states

Create an xml file into drawable folder called 'mybuttondrawable.xml' to define the drawable resource of each of your button states:

drawable/mybutton.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:state_enabled="false"
        android:drawable="@drawable/mybutton_disabled" />
    <item
        android:state_pressed="true"
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_pressed" />
    <item
        android:state_focused="true"
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_focused" />
    <item
        android:state_enabled="true"
        android:drawable="@drawable/mybutton_enabled" />
</selector>
```

Each of those drawables may be images (e.g. mybutton_disabled.png) or xml files defined by you and stored in the drawables folder. For instance:

drawable/mybutton_disabled.xml

```
<?xml version="1.0" encoding="utf-8"?>

    <shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">
        <gradient
            android:startColor="#F2F2F2"
            android:centerColor="#A4A4A4"
            android:endColor="#F2F2F2"
            android:angle="90"/>
        <padding android:left="7dp"
            android:top="7dp"
            android:right="7dp"
            android:bottom="7dp" />
        <stroke
            android:width="2dip"
            android:color="#FFFFFF" />
        <corners android:radius= "8dp" />
    </shape>
```

Then in the layout where you place your button (e.g. MainActivity):

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

            <Button
                android:id="@+id/mybutton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello"
                android:background="@drawable/mybuttondrawable"/>

</LinearLayout>
```

Option 3: Add your button style to your App theme

You can override the default android button style in the definition of your app theme (in values/styles.xml).

styles.xml

```
<resources>
     <style name="AppTheme" parent="android:Theme">
            <item name="colorPrimary">@color/colorPrimary</item>
            <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
            <item name="colorAccent">@color/colorAccent</item>
              <item name="android:button">@style/mybutton</item>
     </style>

     <style name="mybutton" parent="android:style/Widget.Button">
             <item name="android:gravity">center_vertical|center_horizontal</item>
                <item name="android:textColor">#FFFFFFFF</item>
                <item name="android:shadowColor">#FF000000</item>
                <item name="android:shadowDx">0</item>
                <item name="android:shadowDy">-1</item>
                <item name="android:shadowRadius">0.2</item>
                <item name="android:textSize">16dip</item>
                <item name="android:textStyle">bold</item>
                <item name="android:background">@drawable/anydrawable</item>
     </style>
</resources>
```

Option 4: Overlay a color on the default button style programmatically

Just find you button in your activity and apply a color filter:

```
Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)
```

You can check different blending modes here and nice examples here.

Section 43.5: Custom Click Listener to prevent multiple fast clicks

In order to **prevent a button from firing multiple times within a short period of time** (let's say 2 clicks within 1 second, which may cause serious problems if the flow is not controlled), one can implement a custom ***SingleClickListener***.

This ClickListener sets a specific time interval as threshold (for instance, 1000ms).
When the button is clicked, a check will be ran to see if the trigger was executed in the past amount of time you defined, and if not it will trigger it.

```
public class SingleClickListener implements View.OnClickListener {

    protected int defaultInterval;
    private long lastTimeClicked = 0;

    public SingleClickListener() {
        this(1000);
    }

    public SingleClickListener(int minInterval) {
        this.defaultInterval = minInterval;
    }

    @Override
    public void onClick(View v) {
        if (SystemClock.elapsedRealtime() - lastTimeClicked < defaultInterval) {
            return;
        }
        lastTimeClicked = SystemClock.elapsedRealtime();
        performClick(v);
    }

    public abstract void performClick(View v);

}
```

And in the class, the SingleClickListener is associated to the Button at stake

```
myButton = (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(new SingleClickListener() {
    @Override
    public void performClick(View view) {
        // do stuff
    }
});
```

Section 43.6: Using the layout to define a click action

When we create a button in layout, we can use the `android:onClick` attribute to reference a method in code to handle clicks.

Button

```
<Button
   android:width="120dp"
   android:height="wrap_content"
   android:text="Click me"
```

```
    android:onClick="handleClick" />
```

Then in your activity, create the `handleClick` method.

```
public void handleClick(View v) {
    // Do whatever.
}
```

Section 43.7: Listening to the long click events

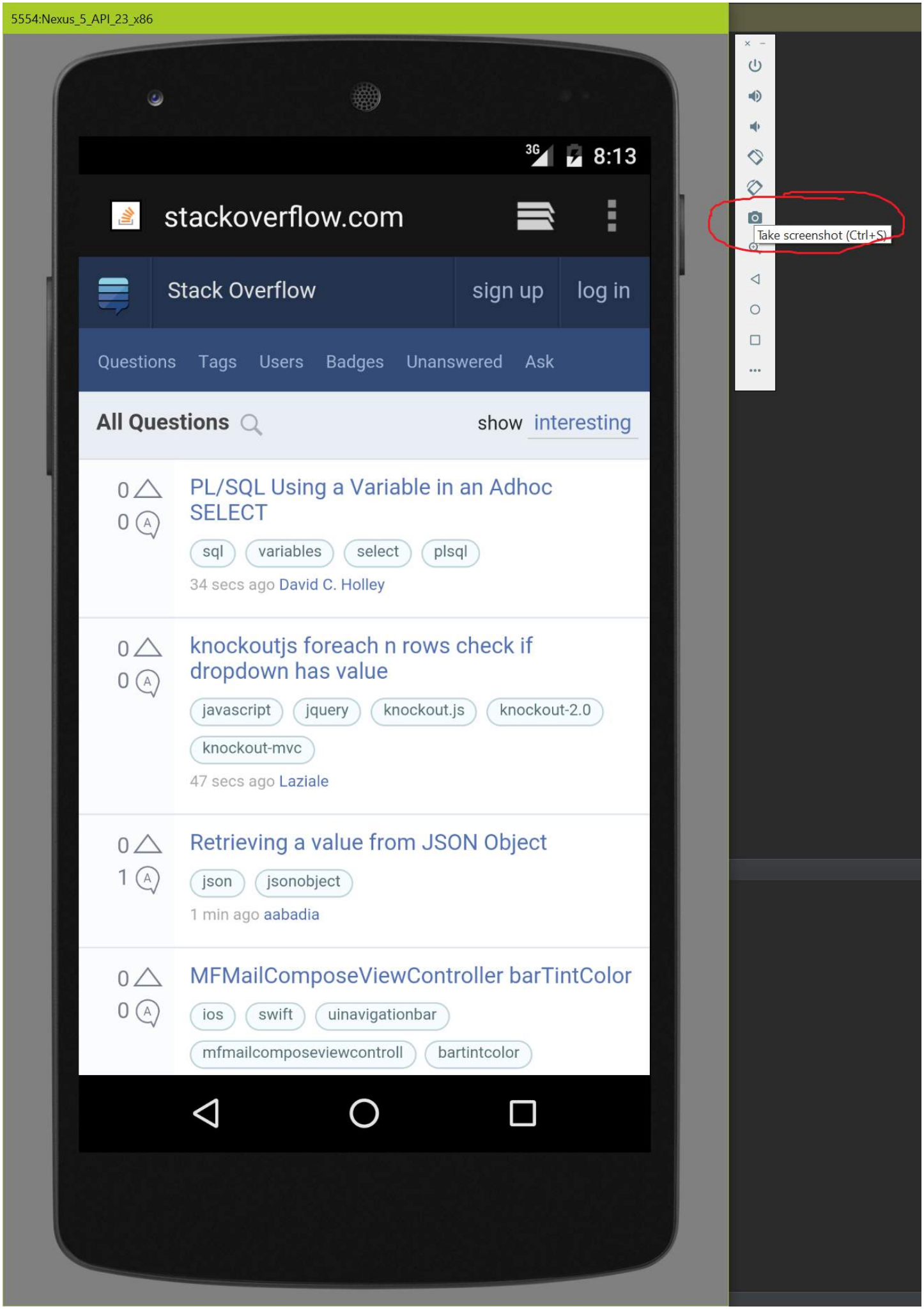
To catch a long click and use it you need to provide appropriate listener to button:

```
View.OnLongClickListener listener = new View.OnLongClickListener() {
    public boolean onLongClick(View v) {
        Button clickedButton = (Button) v;
        String buttonText = clickedButton.getText().toString();
        Log.v(TAG, "button long pressed --> " + buttonText);
        return true;
    }
};

button.setOnLongClickListener(listener);
```

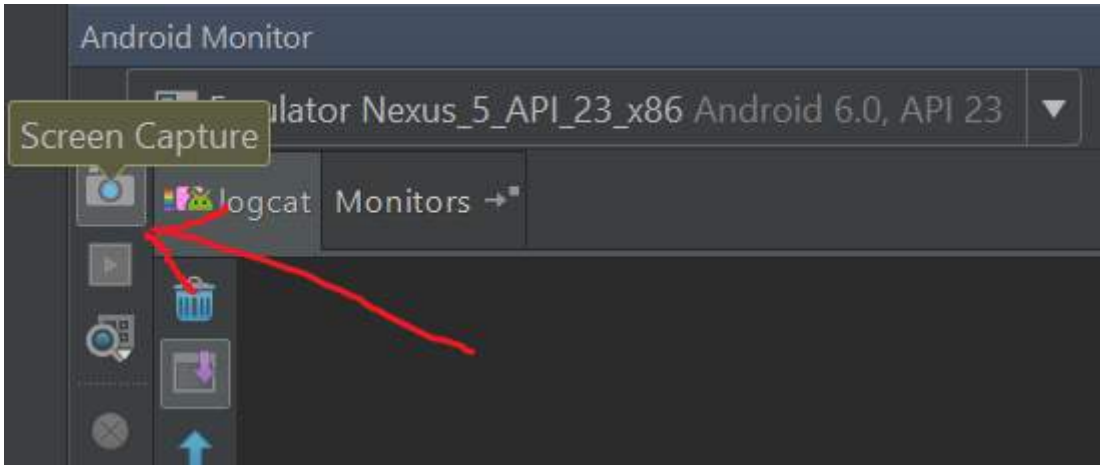
Chapter 44: Emulator

Section 44.1: Taking screenshots

If you want to take a screenshot from the Android Emulator (2.0), then you just need to press `Ctrl` + `S` or you click on the camera icon on the side bar:



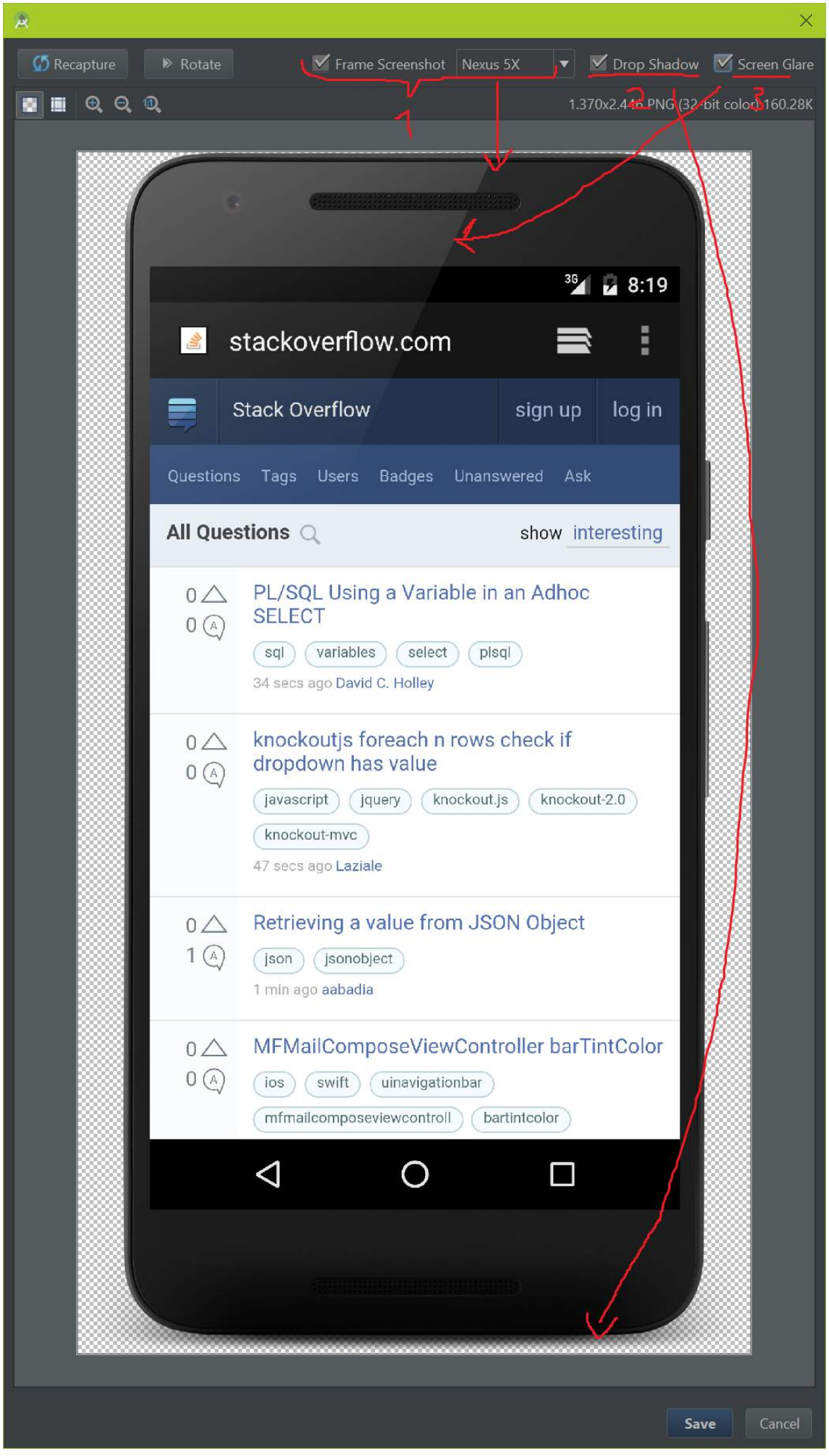
If you use an older version of the Android Emulator or you want to take a screenshot from a real device, then you need to click on the camera icon in the Android Monitor:



Double check that you have selected the right device, because this is a common pitfall.

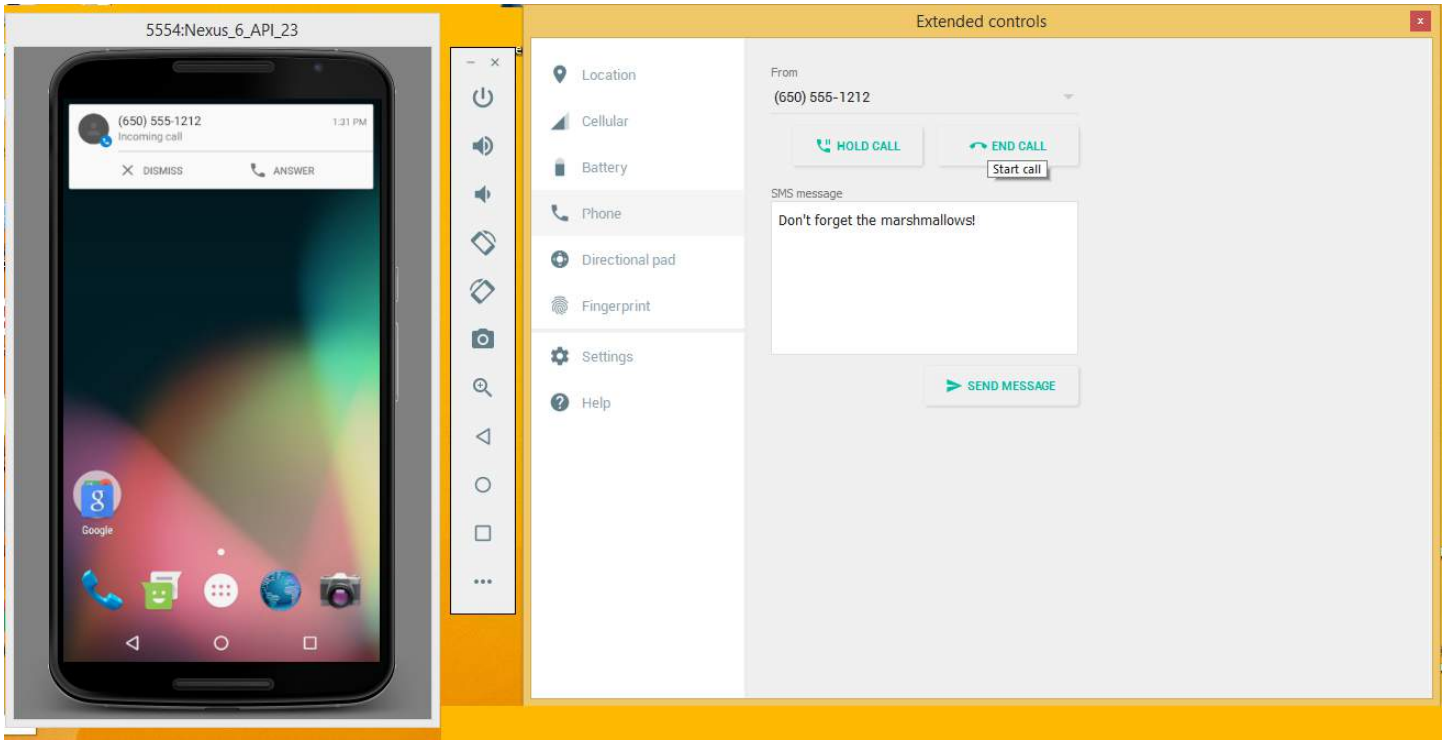
After taking a screenshot, you can optionally add the following decorations to it (also see the image below):

1. A device frame around the screenshot.
2. A drop shadow below the device frame.
3. A screen glare across device frame and screenshot.



Section 44.2: Simulate call

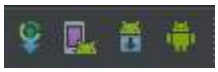
To simulate a phone call, press the 'Extended controls' button indicated by three dots, choose 'Phone' and select 'Call'. You can also optionally change the phone number.



Section 44.3: Open the AVD Manager

Once the SDK installed, you can open the AVD Manager from the command line using `android avd`.

You can also access AVD Manager from Android studio using `Tools` > `Android` > `AVD Manager` or by clicking on the AVD Manager icon in the toolbar which is the second in the screenshot below.



Section 44.4: Resolving Errors while starting emulator

First of all, ensure that you've enabled the '**Virtualization**' in your BIOS setup.

Start the **Android SDK Manager**, select **Extras** and then select **Intel Hardware Accelerated Execution Manager** and wait until your download completes. If it still doesn't work, open your SDK folder and run `/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe`.

Follow the on-screen instructions to complete installation.

Or for OS X you can do it without onscreen prompts like this:
`/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

> If your CPU does not support VT-x or SVM, you can not use x86-based Android images. Please use ARM-based images instead.

After installation completed, confirm that the virtualization driver is operating correctly by opening a command prompt window and running the following command: `sc query intelhaxm`

To run an x86-based emulator with VM acceleration: If you are running the emulator from the command line, just specify an x86-based AVD: `emulator -avd <avd_name>`

If you follow all the steps mentioned above correctly, then surely you should be able to see your AVD with HAXM coming up normally.

Chapter 45: Service

A Service runs in **background** to perform long-running operations or to perform work for remote processes. A service does not provide any user interface it runs only in background with User's input. For example a service can play music in the background while the user is in a different App, or it might download data from the internet without blocking user's interaction with the Android device.

Section 45.1: Lifecycle of a Service

The services lifecycle has the following callbacks

- `onCreate()` :

Executed when the service is first created in order to set up the initial configurations you might need. This method is executed only if the service is not already running.

- `onStartCommand()` :

Executed every time `startService()` is invoked by another component, like an Activity or a BroadcastReceiver. When you use this method, the Service will run until you call `stopSelf()` or `stopService()`. Note that regardless of how many times you call `onStartCommand()`, the methods `stopSelf()` and `stopService()` must be invoked only once in order to stop the service.

- `onBind()` :

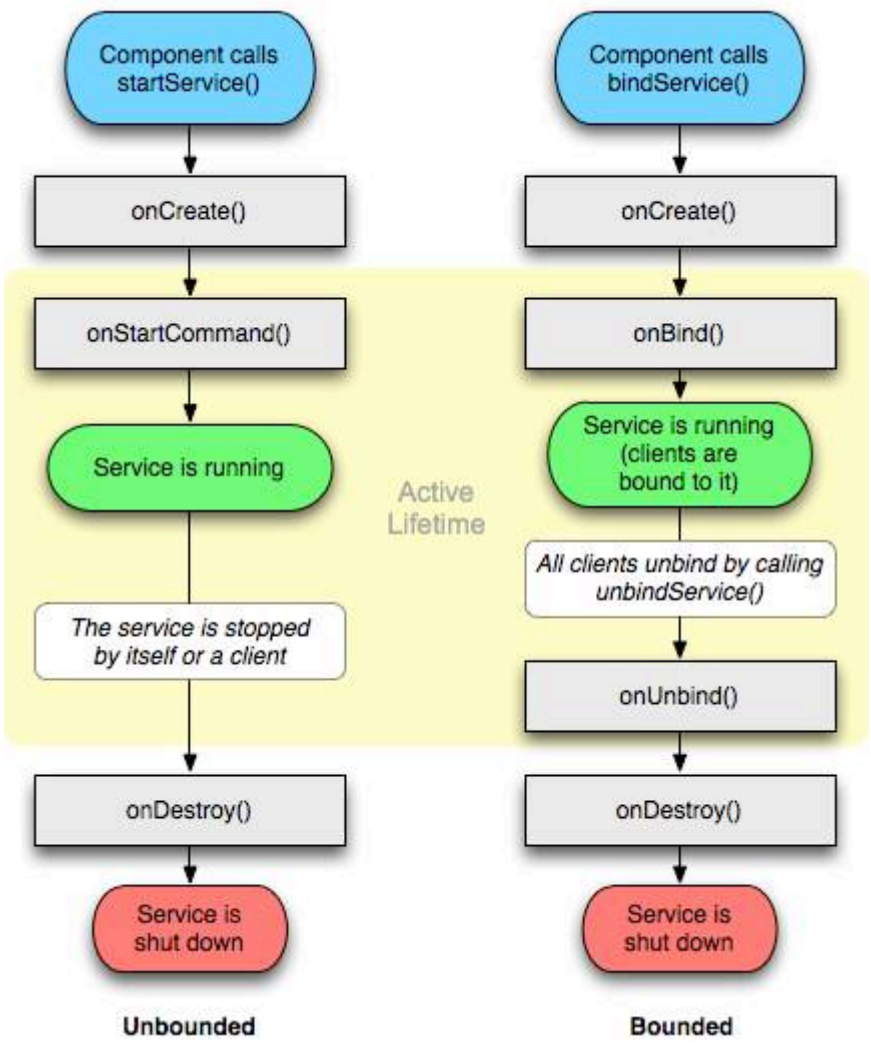
Executed when a component calls `bindService()` and returns an instance of IBInder, providing a communication channel to the Service. A call to `bindService()` will keep the service running as long as there are clients bound to it.

- `onDestroy()` :

Executed when the service is no longer in use and allows for disposal of resources that have been allocated.

It is important to note that during the lifecycle of a service other callbacks might be invoked such as `onConfigurationChanged()` and `onLowMemory()`

https://developer.android.com/guide/components/services.html



Section 45.2: Defining the process of a service

The `android:process` field defines the name of the process where the service is to run. Normally, all components of an application run in the default process created for the application. However, a component can override the default with its own process attribute, allowing you to spread your application across multiple processes.

If the name assigned to this attribute begins with a colon (':'), the service will run in its own separate process.

```
<service
 android:name="com.example.appName"
 android:process=":externalProcess" />
```

If the process name begins with a lowercase character, the service will run in a global process of that name, provided that it has permission to do so. This allows components in different applications to share a process, reducing resource usage.

Section 45.3: Creating an unbound service

The first thing to do is to add the service to `AndroidManifest.xml`, inside the **`<application>`** tag:

```
<application ...>

    ...

    <service
        android:name=".RecordingService"
```

```
        <!--"enabled" tag specifies Whether or not the service can be instantiated by the system —
"true" -->
        <!--if it can be, and "false" if not. The default value is "true".-->
        android:enabled="true"
        <!--exported tag specifies Whether or not components of other applications can invoke the -
->
        <!--service or interact with it — "true" if they can, and "false" if not. When the value-
->
        <!--is "false", only components of the same application or applications with the same user
-->
        <!--ID can start the service or bind to it.-->
        android:exported="false" />

</application>
```

If your intend to manage your service class in a separate package (eg: .AllServices.RecordingService) then you will need to specify where your service is located. So, in above case we will modify:

```
android:name=".RecordingService"
```

to

```
android:name=".AllServices.RecordingService"
```

or the easiest way of doing so is to specify the full package name.

Then we create the actual service class:

```
public class RecordingService extends Service {
    private int NOTIFICATION = 1; // Unique identifier for our notification

    public static boolean isRunning = false;
    public static RecordingService instance = null;


    private NotificationManager notificationManager = null;


    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate(){
        instance = this;
        isRunning = true;

        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        // The PendingIntent to launch our activity if the user selects this notification
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0, new Intent(this,
MainActivity.class), 0);

        // Set the info for the views that show in the notification panel.
```

```
        Notification notification = new NotificationCompat.Builder(this)
                .setSmallIcon(R.mipmap.ic_launcher)        // the status icon
                .setTicker("Service running...")           // the status text
                .setWhen(System.currentTimeMillis())       // the time stamp
                .setContentTitle("My App")                 // the label of the entry
                .setContentText("Service running...")      // the content of the entry
                .setContentIntent(contentIntent)           // the intent to send when the entry is
clicked
                .setOngoing(true)                          // make persistent (disable swipe-away)
                .build();

        // Start service in foreground mode
        startForeground(NOTIFICATION, notification);

        return START_STICKY;
    }


    @Override
    public void onDestroy(){
        isRunning = false;
        instance = null;

        notificationManager.cancel(NOTIFICATION); // Remove notification

        super.onDestroy();
    }


    public void doSomething(){
        Toast.makeText(getApplicationContext(), "Doing stuff from service...",
Toast.LENGTH_SHORT).show();
    }

}
```

All this service does is show a notification when it's running, and it can display toasts when its `doSomething()` method is called.

As you'll notice, it's implemented as a singleton, keeping track of its own instance - but without the usual static singleton factory method because services are naturally singletons and are created by intents. The instance is useful to the outside to get a "handle" to the service when it's running.

Last, we need to start and stop the service from an activity:

```
public void startOrStopService(){
    if( RecordingService.isRunning ){
        // Stop service
        Intent intent = new Intent(this, RecordingService.class);
        stopService(intent);
    }
    else {
        // Start service
        Intent intent = new Intent(this, RecordingService.class);
        startService(intent);
    }
}
```

In this example, the service is started and stopped by the same method, depending on it's current state.

We can also invoke the `doSomething()` method from our activity:

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

Section 45.4: Starting a Service

Starting a service is very easy, just call `startService` with an intent, from within an Activity:

```
Intent intent = new Intent(this, MyService.class);  //substitute MyService with the name of your
service
intent.putExtra(Intent.EXTRA_TEXT, "Some text"); //add any extra data to pass to the service

startService(intent); //Call startService to start the service.
```

Section 45.5: Creating Bound Service with help of Binder

Create a class which extends `Service` class and in overridden method `onBind` return your local binder instance:

```
public class LocalService extends Service {
    // Binder given to clients
    private final IBinder mBinder = new LocalBinder();

    /**
     * Class used for the client Binder.  Because we know this service always
     * runs in the same process as its clients, we don't need to deal with IPC.
     */
    public class LocalBinder extends Binder {
        LocalService getService() {
            // Return this instance of LocalService so clients can call public methods
            return LocalService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
}
```

Then in your activity bind to service in `onStart` callback, using `ServiceConnection` instance and unbind from it in `onStop`:

```
public class BindingActivity extends Activity {
    LocalService mService;
    boolean mBound = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected void onStart() {
        super.onStart();
```

```
        // Bind to LocalService
        Intent intent = new Intent(this, LocalService.class);
        bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        // Unbind from the service
        if (mBound) {
            unbindService(mConnection);
            mBound = false;
        }
    }

    /** Defines callbacks for service binding, passed to bindService() */
    private ServiceConnection mConnection = new ServiceConnection() {

        @Override
        public void onServiceConnected(ComponentName className,
                IBinder service) {
            // We've bound to LocalService, cast the IBinder and get LocalService instance
            LocalBinder binder = (LocalBinder) service;
            mService = binder.getService();
            mBound = true;
        }

        @Override
        public void onServiceDisconnected(ComponentName arg0) {
            mBound = false;
        }
    };
}
```

Section 45.6: Creating Remote Service (via AIDL)

Describe your service access interface through `.aidl` file:

```
// IRemoteService.aidl
package com.example.android;

// Declare any non-default types here with import statements

/** Example service interface */
interface IRemoteService {
    /** Request the process ID of this service, to do evil things with it. */
    int getPid();
}
```

Now after build application, sdk tools will generate appropriate `.java` file. This file will contain `Stub` class which implements our aidl interface, and which we need to extend:

```
public class RemoteService extends Service {

    private final IRemoteService.Stub binder = new IRemoteService.Stub() {
        @Override
        public int getPid() throws RemoteException {
            return Process.myPid();
        }
    };
```

```
    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return binder;
    }
}
```

Then in activity:

```
public class MainActivity extends AppCompatActivity {
    private final ServiceConnection connection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder iBinder) {
            IRemoteService service = IRemoteService.Stub.asInterface(iBinder);
            Toast.makeText(this, "Activity process: " + Process.myPid + ", Service process: " +
getRemotePid(service), LENGTH_SHORT).show();
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {}
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, RemoteService.class);
        bindService(intent, connection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onStop() {
        super.onStop();
        unbindService(connection);
    }

    private int getRemotePid(IRemoteService service) {
        int result = -1;

        try {
            result = service.getPid();
        } catch (RemoteException e) {
            e.printStackTrace();
        }

        return result;
    }
}
```

Chapter 46: The Manifest File

The Manifest is an obligatory file named exactly "AndroidManifest.xml" and located in the app's root directory. It specifies the app name, icon, Java package name, version, declaration of Activities, Services, app permissions and other information.

Section 46.1: Declaring Components

The primary task of the manifest is to inform the system about the app's components. For example, a manifest file can declare an activity as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

In the **<application>** element, the `android:icon` attribute points to resources for an icon that identifies the app.

In the element, the `android:name` attribute specifies the fully qualified class name of the Activity subclass and the android:label attribute specifies a string to use as the user-visible label for the activity.

You must declare all app components this way:

-**<activity>** elements for activities

-**<service>** elements for services

-**<receiver>** elements for broadcast receivers

-**<provider>** elements for content providers

Activities, services, and content providers that you include in your source but do not declare in the manifest are not visible to the system and, consequently, can never run. However, broadcast receivers can be either declared in the manifest or created dynamically in code (as `BroadcastReceiver` objects) and registered with the system by calling `registerReceiver()`.

For more about how to structure the manifest file for your app, see The AndroidManifest.xml File documentation.

Section 46.2: Declaring permissions in your manifest file

Any permission required by your application to access a protected part of the API or to interact with other applications must be declared in your `AndroidManifest.xml` file. This is done using the **<uses-permission />** tag.

Syntax

```
<uses-permission android:name="string"
   android:maxSdkVersion="integer"/>
```

android:name: This is the name of the required permission

android:maxSdkVersion: The highest API level at which this permission should be granted to your app. Setting this permission is optional and should only be set if the permission your app requires is no longer needed at a certain API level.

Sample AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.samplepackage">

    <!-- request internet permission -->
    <uses-permission android:name="android.permission.INTERNET" />

    <!-- request camera permission -->
    <uses-permission android:name="android.permission.CAMERA"/>

    <!-- request permission to write to external storage -->
    <uses-permission
         android:name="android.permission.WRITE_EXTERNAL_STORAGE"
         android:maxSdkVersion="18" />

    <application>....</application>
</manifest>
```

* Also see the Permissions topic.

Chapter 47: Gradle for Android

Gradle is a JVM-based build system that enables developers to write high-level scripts that can be used to automate the process of compilation and application production. It is a flexible plugin-based system, which allows you to automate various aspects of the build process; including compiling and signing a `.jar`, downloading and managing external dependencies, injecting fields into the `AndroidManifest` or utilising specific SDK versions.

Section 47.1: A basic build.gradle file

This is an example of a default `build.gradle` file in a module.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion '25.0.3'

    signingConfigs {
        applicationName {
            keyAlias 'applicationName'
            keyPassword 'password'
            storeFile file('../key/applicationName.jks')
            storePassword 'keystorePassword'
        }
    }
    defaultConfig {
        applicationId 'com.company.applicationName'
        minSdkVersion 14
        targetSdkVersion 25
        versionCode 1
        versionName '1.0'
        signingConfig signingConfigs.applicationName
    }
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])

    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'

    testCompile 'junit:junit:4.12'
}
```

DSL (domain-specific language)

Each block in the file above is called a `DSL` (domain-specific language).

Plugins

The first line, `apply plugin: 'com.android.application'`, applies the Android plugin for Gradle to the build and

makes the `android {}` block available to declare Android-specific build options.

For an **Android Application**:

```
apply plugin: 'com.android.application'
```

For an **Android Library**:

```
apply plugin: 'com.android.library'
```

Understanding the DSLs in the sample above

The second part, The `android {...}` block, is the Android `DSL` which contains information about your project.

For example, you can set the `compileSdkVersion` which specifies the Android API level , Which should be used by Gradle to compile your app.
The sub-block `defaultConfig` holds the defaults for your manifest. You can `override` them with Product Flavors.

You can find more info in these examples:

- DSL for the app module
- Build Types
- Product Flavors
- Signing settings

Dependencies

The `dependencies` block is defined outside the `android` block `{...}` : This means it's not defined by the Android plugin but it's standard Gradle.
The `dependencies` block specifies what external libraries (typically Android libraries, but Java libraries are also valid) you wish to include in your app. Gradle will automatically download these dependencies for you (if there is no local copy available), you just need to add similar `compile` lines when you wish to add another library.

Let's look at one of the lines present here:

> `compile 'com.android.support:design:25.3.1'`

This line basically says

> add a dependency on the Android support design library to my project.

Gradle will ensure that the library is downloaded and present so that you can use it in your app, and its code will also be included in your app.

If you're familiar with Maven, this syntax is the *GroupId*, a colon, *ArtifactId*, another colon, then the version of the dependency you wish to include, giving you full control over versioning.

While it is possible to specify artifact versions using the plus (+) sign, best practice is to avoid doing so; it can lead to issues if the library gets updated with breaking changes without your knowledge, which would likely lead to crashes in your app.

You can add different kind of dependencies:

- local binary dependencies
- module dependencies
- remote dependencies

A particular attention should be dedicated to the aar flat dependencies.

You can find more details in this topic.

Note about the **-v7 in *appcompat-v7***

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

This simply means that this **library** (`appcompat`) is compatible with the Android API level 7 and forward.

Note about the **junit:junit:4.12**

This is Testing dependency for Unit testing.

Specifying dependencies specific to different build configurations

You can specify that a dependency should only be used for a certain build configuration or you can define different dependencies for the build types or the product flavors (e.g., debug, test or release) by using `debugCompile`, `testCompile` or `releaseCompile` instead of the usual `compile`.

This is helpful for keeping test- and debug- related dependencies out of your release build, which will keep your release `APK` as slim as possible and help to ensure that any debug information cannot be used to obtain internal information about your app.

signingConfig

The `signingConfig` allows you to configure your Gradle to include `keystore` information and ensure that the APK built using these configurations are signed and ready for Play Store release.

Here you can find a dedicated topic.

Note: It's not recommended though to keep the signing credentials inside your Gradle file. To remove the signing configurations, just omit the `signingConfigs` portion.
You can specify them in different ways:

- storing in an external file
- storing them in setting environment variables.

See this topic for more details : Sign APK without exposing keystore password.

You can find further information about Gradle for Android in the dedicated Gradle topic.

Section 47.2: Define and use Build Configuration Fields

BuildConfigField

Gradle allows `buildConfigField` lines to define constants. These constants will be accessible at runtime as static fields of the `BuildConfig` class. This can be used to create flavors by defining all fields within the `defaultConfig` block, then overriding them for individual build flavors as needed.

This example defines the build date and flags the build for production rather than test:

```
android {
    ...
    defaultConfig {
        ...
        // defining the build date
        buildConfigField "long", "BUILD_DATE", System.currentTimeMillis() + "L"
        // define whether this build is a production build
        buildConfigField "boolean", "IS_PRODUCTION", "false"
        // note that to define a string you need to escape it
        buildConfigField "String", "API_KEY", "\"my_api_key\""
    }

    productFlavors {
        prod {
            // override the productive flag for the flavor "prod"
            buildConfigField "boolean", "IS_PRODUCTION", "true"
            resValue 'string', 'app_name', 'My App Name'
        }
        dev {
            // inherit default fields
            resValue 'string', 'app_name', 'My App Name - Dev'
        }
    }
}
```

The automatically-generated <package_name>.`BuildConfig`.java in the gen folder contains the following fields based on the directive above:

```
public class BuildConfig {
    // ... other generated fields ...
    public static final long BUILD_DATE = 1469504547000L;
    public static final boolean IS_PRODUCTION = false;
    public static final String API_KEY = "my_api_key";
}
```

The defined fields can now be used within the app at runtime by accessing the generated `BuildConfig` class:

```
public void example() {
    // format the build date
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
    String buildDate = dateFormat.format(new Date(BuildConfig.BUILD_DATE));
    Log.d("build date", buildDate);

    // do something depending whether this is a productive build
    if (BuildConfig.IS_PRODUCTION) {
        connectToProductionApiEndpoint();
    } else {
        connectToStagingApiEndpoint();
    }
}
```

ResValue

The `resValue` in the `productFlavors` creates a resource value. It can be any type of resource (`string`, `dimen`, `color`, etc.). This is similar to defining a resource in the appropriate file: e.g. defining string in a `strings.xml` file. The advantage being that the one defined in gradle can be modified based on your productFlavor/buildVariant. To access the value, write the same code as if you were accessing a res from the resources file:

```
getResources().getString(R.string.app_name)
```

The important thing is that resources defined this way cannot modify existing resources defined in files. They can only create new resource values.

Some libraries (such as the Google Maps Android API) require an API key provided in the Manifest as a `meta-data` tag. If different keys are needed for debugging and production builds, specify a manifest placeholder filled in by Gradle.

In your `AndroidManifest.xml` file:

```
<meta-data
   android:name="com.google.android.geo.API_KEY"
   android:value="${MAPS_API_KEY}"/>
```

And then set the field accordingly in your `build.gradle` file:

```
android {
    defaultConfig {
        ...
        // Your development key
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
    }

    productFlavors {
        prod {
            // Your production key
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
        }
    }
}
```

The Android build system generates a number of fields automatically and places them in `BuildConfig.java`. These fields are:

Field	Description
`DEBUG`	a `Boolean` stating if the app is in debug or release mode
`APPLICATION_ID`	a `String` containing the ID of the application (e.g. `com.example.app`)
`BUILD_TYPE`	a `String` containing the build type of the application (usually either `debug` or `release`)
`FLAVOR`	a `String` containing the particular flavor of the build
`VERSION_CODE`	an **`int`** containing the version (build) number. This is the same as `versionCode` in `build.gradle` or `versionCode` in `AndroidManifest.xml`
`VERSION_NAME`	a `String` containing the version (build) name. This is the same as `versionName` in `build.gradle` or `versionName` in `AndroidManifest.xml`

In addition to the above, if you have defined multiple dimensions of flavor then each dimension will have its own value. For example, if you had two dimensions of flavor for `color` and `size` you will also have the following variables:

Field	Description

`FLAVOR_color` a `String` containing the value for the 'color' flavor.

`FLAVOR_size` a `String` containing the value for the 'size' flavor.

Section 47.3: Centralizing dependencies via "dependencies.gradle" file

When working with multi-module projects, it is helpful to centralize dependencies in a single location rather than having them spread across many build files, especially for common libraries such as the Android support libraries and the Firebase libraries.

One recommended way is to separate the Gradle build files, with one `build.gradle` per module, as well as one in the project root and another one for the dependencies, for example:

```
root
  +- gradleScript/
  |     dependencies.gradle
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
```

Then, all of your dependencies can be located in `gradleScript/dependencies.gradle`:

```
ext {
    // Version
    supportVersion = '24.1.0'

    // Support Libraries dependencies
    supportDependencies = [
            design:            "com.android.support:design:${supportVersion}",
            recyclerView:      "com.android.support:recyclerview-v7:${supportVersion}",
            cardView:          "com.android.support:cardview-v7:${supportVersion}",
            appCompat:         "com.android.support:appcompat-v7:${supportVersion}",
            supportAnnotation: "com.android.support:support-annotations:${supportVersion}",
    ]

    firebaseVersion = '9.2.0';

    firebaseDependencies = [
            core:         "com.google.firebase:firebase-core:${firebaseVersion}",
            database:     "com.google.firebase:firebase-database:${firebaseVersion}",
            storage:      "com.google.firebase:firebase-storage:${firebaseVersion}",
            crash:        "com.google.firebase:firebase-crash:${firebaseVersion}",
            auth:         "com.google.firebase:firebase-auth:${firebaseVersion}",
            messaging:    "com.google.firebase:firebase-messaging:${firebaseVersion}",
            remoteConfig: "com.google.firebase:firebase-config:${firebaseVersion}",
            invites:      "com.google.firebase:firebase-invites:${firebaseVersion}",
            adMod:        "com.google.firebase:firebase-ads:${firebaseVersion}",
            appIndexing:  "com.google.android.gms:play-services-appindexing:${firebaseVersion}",
    ];
}
```

Which can then be applied from that file in the top level file `build.gradle` like so:

```
// Load dependencies
apply from: 'gradleScript/dependencies.gradle'
```

and in the `module1/build.gradle` like so:

```
// Module build file
dependencies {
    // ...
    compile supportDependencies.appCompat
    compile supportDependencies.design
    compile firebaseDependencies.crash
}
```

Another approach

A less verbose approach for centralizing library dependencies versions can be achieved by declaring the version number as a variable once, and using it everywhere.

In the workspace root `build.gradle` add this:

```
ext.v = [
    supportVersion:'24.1.1',
]
```

And in every module that uses the same library add the needed libraries

```
compile "com.android.support:support-v4:${v.supportVersion}"
compile "com.android.support:recyclerview-v7:${v.supportVersion}"
compile "com.android.support:design:${v.supportVersion}"
compile "com.android.support:support-annotations:${v.supportVersion}"
```

Section 47.4: Sign APK without exposing keystore password

You can define the signing configuration to sign the apk in the `build.gradle` file using these properties:

- `storeFile` : the keystore file
- `storePassword`: the keystore password
- `keyAlias`: a key alias name
- `keyPassword`: A key alias password

In many case you may need to avoid this kind of info in the `build.gradle` file.

Method A: Configure release signing using a keystore.properties file

It's possible to configure your app's `build.gradle` so that it will read your signing configuration information from a properties file like `keystore.properties`.

Setting up signing like this is beneficial because:

- Your signing configuration information is separate from your `build.gradle` file
- You do not have to intervene during the signing process in order to provide passwords for your keystore file
- You can easily exclude the `keystore.properties` file from version control

First, create a file called `keystore.properties` in the root of your project with content like this (replacing the values with your own):

```
storeFile=keystore.jks
storePassword=storePassword
keyAlias=keyAlias
```

```
keyPassword=keyPassword
```

Now, in your app's `build.gradle` file, set up the `signingConfigs` block as follows:

```
android {
...

    signingConfigs {
        release {
            def propsFile = rootProject.file('keystore.properties')
            if (propsFile.exists()) {
                def props = new Properties()
                props.load(new FileInputStream(propsFile))
                storeFile = file(props['storeFile'])
                storePassword = props['storePassword']
                keyAlias = props['keyAlias']
                keyPassword = props['keyPassword']
            }
        }
    }
}
```

That's really all there is to it, **but don't forget to exclude both your keystore file and your `keystore.properties` file from version control**.

A couple of things to note:

- The `storeFile` path specified in the `keystore.properties` file should be relative to your app's `build.gradle` file. This example assumes that the keystore file is in the same directory as the app's `build.gradle` file.
- This example has the `keystore.properties` file in the root of the project. If you put it somewhere else, be sure to change the value in `rootProject.file('keystore.properties')` to the location of yours, relative to the root of your project.

Method B: By using an environment variable

The same can be achieved also without a properties file, making the password harder to find:

```
android {

  signingConfigs {
    release {
        storeFile file('/your/keystore/location/key')
        keyAlias 'your_alias'
        String ps = System.getenv("ps")
        if (ps == null) {
             throw new GradleException('missing ps env variable')
        }
        keyPassword ps
        storePassword ps
    }
}
```

The `"ps"` environment variable can be global, but a safer approach can be by adding it to the shell of Android Studio only.
In linux this can be done by editing Android Studio's `Desktop Entry`

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

You can find more details in this topic.

Section 47.5: Adding product flavor-specific dependencies

Dependencies can be added for a specific product flavor, similar to how they can be added for specific build configurations.

For this example, assume that we have already defined two product flavors called `free` and `paid` (more on defining flavors here).
We can then add the AdMob dependency for the `free` flavor, and the Picasso library for the `paid` one like so:

```
android {
    ...

    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}

...
dependencies {
    ...
    // Add AdMob only for free flavor
    freeCompile 'com.android.support:appcompat-v7:23.1.1'
    freeCompile 'com.google.android.gms:play-services-ads:8.4.0'
    freeCompile 'com.android.support:support-v4:23.1.1'

    // Add picasso only for paid flavor
    paidCompile 'com.squareup.picasso:picasso:2.5.2'
}
...
```

Section 47.6: Specifying different application IDs for build types and product flavors

You can specify different application IDs or package names for each `buildType` or `productFlavor` using the **applicationIdSuffix** configuration attribute:

Example of suffixing the `applicationId` for each `buildType`:

```
defaultConfig {
    applicationId "com.package.android"
    minSdkVersion 17
    targetSdkVersion 23
    versionCode 1
    versionName "1.0"
}

buildTypes {
    release {
        debuggable false
```

```
    }

    development {
        debuggable true
        applicationIdSuffix ".dev"
    }

    testing {
        debuggable true
        applicationIdSuffix ".qa"
    }
}
```

Our resulting `applicationIds` would now be:

- com.package.android for `release`
- com.package.android.**dev** for `development`
- com.package.android.**qa** for `testing`

This can be done for `productFlavors` as well:

```
productFlavors {
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
    }
}
```

The resulting `applicationIds` would be:

- com.package.android.**free** for the `free` flavor
- com.package.android.**paid** for the `paid` flavor

Section 47.7: Versioning your builds via "version.properties" file

You can use Gradle to auto-increment your package version each time you build it. To do so create a `version.properties` file in the same directory as your `build.gradle` with the following contents:

```
VERSION_MAJOR=0
VERSION_MINOR=1
VERSION_BUILD=1
```

(Changing the values for major and minor as you see fit). Then in your `build.gradle` add the following code to the `android` section:

```
// Read version information from local file and increment as appropriate
def versionPropsFile = file('version.properties')
if (versionPropsFile.canRead()) {
  def Properties versionProps = new Properties()

  versionProps.load(new FileInputStream(versionPropsFile))

  def versionMajor = versionProps['VERSION_MAJOR'].toInteger()
  def versionMinor = versionProps['VERSION_MINOR'].toInteger()
```

```
  def versionBuild = versionProps['VERSION_BUILD'].toInteger() + 1

  // Update the build number in the local file
  versionProps['VERSION_BUILD'] = versionBuild.toString()
  versionProps.store(versionPropsFile.newWriter(), null)

  defaultConfig {
    versionCode versionBuild
    versionName "${versionMajor}.${versionMinor}." + String.format("%05d", versionBuild)
  }
}
```

The information can be accessed in Java as a string `BuildConfig.VERSION_NAME` for the complete {major}.{minor}.{build} number and as an integer `BuildConfig.VERSION_CODE` for just the build number.

Section 47.8: Defining product flavors

Product flavors are defined in the `build.gradle` file inside the `android { ... }` block as seen below.

```
...
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}
```

By doing this, we now have two additional product flavors: `free` and `paid`. Each can have its own specific configuration and attributes. For example, both of our new flavors has a separate `applicationId` and `versionName` than our existing `main` flavor (available by default, so not shown here).

Section 47.9: Changing output apk name and add version name:

This is the code for changing output application file name (.apk). The name can be configured by assigning a different value to `newName`

```
android {

    applicationVariants.all { variant ->
        def newName = "ApkName";
        variant.outputs.each { output ->
            def apk = output.outputFile;

            newName += "-v" + defaultConfig.versionName;
            if (variant.buildType.name == "release") {
                newName += "-release.apk";
            } else {
                newName += ".apk";
            }
            if (!output.zipAlign) {
```

```
                newName = newName.replace(".apk", "-unaligned.apk");
            }

            output.outputFile = new File(apk.parentFile, newName);
            logger.info("INFO: Set outputFile to "
                        + output.outputFile
                        + " for [" + output.name + "]");
        }
    }
}
```

Section 47.10: Adding product flavor-specific resources

Resources can be added for a specific product flavor.

For this example, assume that we have already defined two product flavors called `free` and `paid`. In order to add product flavor-specific resources, we create additional resource folders alongside the `main/res` folder, which we can then add resources to like usual. For this example, we'll define a string, `status`, for each product flavor:

*/src/**main**/res/values/strings.xml*

```
<resources>
    <string name="status">Default</string>
</resources>
```

*/src/**free**/res/values/strings.xml*

```
<resources>
    <string name="status">Free</string>
</resources>
```

*/src/**paid**/res/values/strings.xml*

```
<resources>
    <string name="status">Paid</string>
</resources>
```

The product flavor-specific `status` strings will override the value for `status` in the `main` flavor.

Section 47.11: Why are there two build.gradle files in an Android Studio project?

`<PROJECT_ROOT>\app\build.gradle` is specific for **app module**.

`<PROJECT_ROOT>\build.gradle` is a **"Top-level build file"** where you can add configuration options common to all sub-projects/modules.

If you use another module in your project, as a local library you would have another `build.gradle` file:
`<PROJECT_ROOT>\module\build.gradle`

In the top level file you can specify common properties as the buildscript block or some common properties.

```
buildscript {
    repositories {
        mavenCentral()
    }
```

```
    dependencies {
       classpath 'com.android.tools.build:gradle:2.2.0'
       classpath 'com.google.gms:google-services:3.0.0'
    }
}

ext {
    compileSdkVersion = 23
    buildToolsVersion = "23.0.1"
}
```

In the `app\build.gradle` you define only the properties for the module:

```
apply plugin: 'com.android.application'


android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //.....
}
```

Section 47.12: Directory structure for flavor-specific resources

Different flavors of application builds can contain different resources. To create a flavor-specific resource make a directory with the lower-case name of your flavor in the `src` directory and add your resources in the same way you would normally.

For example, if you had a flavour `Development` and wanted to provide a distinct launcher icon for it you would create a directory `src/development/res/drawable-mdpi` and inside that directory create an `ic_launcher.png` file with your development-specific icon.

The directory structure will look like this:

```
src/
  main/
    res/
      drawable-mdpi/
        ic_launcher.png  <-- the default launcher icon
  development/
    res/
      drawable-mdpi/
        ic_launcher.png  <-- the launcher icon used when the product flavor is 'Development'
```

(Of course, in this case you would also create icons for drawable-hdpi, drawable-xhdpi *etc*).

Section 47.13: Enable Proguard using gradle

For enabling Proguard configurations for your application you need to enable it in your module-level gradle file. You need to set the value of `minifyEnabled` to **`true`**.

```
buildTypes {
        release {
```

```
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
```

The above code will apply your Proguard configurations contained in the default Android SDK combined with the "proguard-rules.pro" file on your module to your released apk.

Section 47.14: Ignoring build variant

For some reasons you may want to ignore your build variants. For example: you have 'mock' product flavour and you use it only for debug purposes, such as unit/instrumentation tests.

Let's ignore **mockRelease** variant from our project. Open **build.gradle** file and write:

```
    // Remove mockRelease as it's not needed.
    android.variantFilter { variant ->
        if (variant.buildType.name.equals('release') &&
variant.getFlavors().get(0).name.equals('mock')) {
            variant.setIgnore(true);
        }
    }
```

Section 47.15: Enable experimental NDK plugin support for Gradle and AndroidStudio

Enable and configure the experimental Gradle plugin to improve AndroidStudio's NDK support. Check that you fulfill the following requirements:

- Gradle 2.10 (for this example)
- Android NDK r10 or later
- Android SDK with build tools v19.0.0 or later

Configure MyApp/build.gradle file

Edit the dependencies.classpath line in build.gradle from e.g.

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

to

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

(v0.7.2 was the latest version at the time of writing. Check the latest version yourself and adapt your line accordingly)

The build.gradle file should look similar to this:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.2'
    }
}
```

```
allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Configure MyApp/app/build.gradle file

Edit the build.gradle file to look similar to the following example. Your version numbers may look different.

```
apply plugin: 'com.android.model.application'

model {
    android {
        compileSdkVersion 19
        buildToolsVersion "24.0.1"

        defaultConfig {
            applicationId "com.example.mydomain.myapp"
            minSdkVersion.apiLevel 19
            targetSdkVersion.apiLevel 19
            versionCode 1
            versionName "1.0"
        }
        buildTypes {
            release {
                minifyEnabled false
                proguardFiles.add(file('proguard-android.txt'))
            }
        }
        ndk {
            moduleName "myLib"

            /* The following lines are examples of a some optional flags that
               you may set to configure your build environment
            */
            cppFlags.add("-I${file("path/to/my/includes/dir")}".toString())
            cppFlags.add("-std=c++11")
            ldLibs.addAll(['log', 'm'])
            stl = "c++_static"
            abiFilters.add("armeabi-v7a")
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

Sync and check that there are no errors in the Gradle files before proceeding.

Test if plugin is enabled

First make sure you have downloaded the Android NDK module. Then create an new app in AndroidStudio and add the following to the ActivityMain file:

```
public class MainActivity implements Activity {
```

```
    onCreate() {
        // Pregenerated code. Not important here
    }
    static {
        System.loadLibrary("myLib");
    }
    public static native String getString();
}
```

The `getString()` part should be highlighted red saying that the corresponding JNI function could not be found. Hover your mouse over the function call until a red lightbulb appears. Click the bulb and select `create function JNI_...`. This should generate a myLib.c file in the myApp/app/src/main/jni directory with the correct JNI function call. It should look similar to this:

```
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}
```

If it doesn't look like this, then the plugin has not correctly been configured or the NDK has not been downloaded

Section 47.16: Display signing information

In some circumstances (for example obtaining a Google API key) you need to find your keystore fingerprint. Gradle has a convenient task that display all the signing information, including keystore fingerprints:

```
./gradlew signingReport
```

This is a sample output:

```
:app:signingReport
Variant: release
Config: none
----------
Variant: debug
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044
----------
Variant: debugAndroidTest
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044
----------
Variant: debugUnitTest
Config: debug
Store: /Users/user/.android/debug.keystore
Alias: AndroidDebugKey
```

```
MD5: 25:08:76:A9:7C:0C:19:35:99:02:7B:00:AA:1E:49:CA
SHA1: 26:BE:89:58:00:8C:5A:7D:A3:A9:D3:60:4A:30:53:7A:3D:4E:05:55
Valid until: Saturday 18 June 2044
----------
Variant: releaseUnitTest
Config: none
----------
```

Section 47.17: Seeing dependency tree

Use the task dependencies. Depending on how your modules are set up, it may be either `./gradlew dependencies` or to see the dependencies of module app use `./gradlew :app:dependencies`

The example following build.gradle file

```
dependencies {
    compile 'com.android.support:design:23.2.1'
    compile 'com.android.support:cardview-v7:23.1.1'

    compile 'com.google.android.gms:play-services:6.5.87'
}
```

will produce the following graph:

```
Parallel execution is an incubating feature.
:app:dependencies

------------------------------------------------------------
Project :app
------------------------------------------------------------
. . .
_releaseApk - ## Internal use, do not manually configure ##
+--- com.android.support:design:23.2.1
|    +--- com.android.support:support-v4:23.2.1
|    |    \--- com.android.support:support-annotations:23.2.1
|    +--- com.android.support:appcompat-v7:23.2.1
|    |    +--- com.android.support:support-v4:23.2.1 (*)
|    |    +--- com.android.support:animated-vector-drawable:23.2.1
|    |    |    \--- com.android.support:support-vector-drawable:23.2.1
|    |    |         \--- com.android.support:support-v4:23.2.1 (*)
|    |    \--- com.android.support:support-vector-drawable:23.2.1 (*)
|    \--- com.android.support:recyclerview-v7:23.2.1
|         +--- com.android.support:support-v4:23.2.1 (*)
|         \--- com.android.support:support-annotations:23.2.1
+--- com.android.support:cardview-v7:23.1.1
\--- com.google.android.gms:play-services:6.5.87
\--- com.android.support:support-v4:21.0.0 -> 23.2.1 (*)

. . .
```

Here you can see the project is directly including `com.android.support:design` version 23.2.1, which itself is bringing `com.android.support:support-v4` with version 23.2.1. However, `com.google.android.gms:play-services` itself has a dependency on the same `support-v4` but with an older version 21.0.0, which is a conflict detected by gradle.

`(*)` are used when gradle skips the subtree because those dependencies were already listed previously.

Section 47.18: Disable image compression for a smaller APK file size

If you are optimizing all images manually, disable APT Cruncher for a smaller APK file size.

```
android {

    aaptOptions {
        cruncherEnabled = false
    }
}
```

Section 47.19: Delete "unaligned" apk automatically

If you don't need automatically generated apk files with `unaligned` suffix (which you probably don't), you may add the following code to `build.gradle` file:

```
// delete unaligned files
android.applicationVariants.all { variant ->
  variant.assemble.doLast {
    variant.outputs.each { output ->
        println "aligned " + output.outputFile
        println "unaligned " + output.packageApplication.outputFile

        File unaligned = output.packageApplication.outputFile;
        File aligned = output.outputFile
        if (!unaligned.getName().equalsIgnoreCase(aligned.getName())) {
            println "deleting " + unaligned.getName()
            unaligned.delete()
        }
    }
  }
}
```

From here

Section 47.20: Executing a shell script from gradle

A shell script is a very versatile way to extend your build to basically anything you can think of.

As an example, here is a simple script to compile protobuf files and add the result java files to the source directory for further compilation:

```
def compilePb() {
    exec {
        // NOTICE: gradle will fail if there's an error in the protoc file...
        executable "../pbScript.sh"
    }
}

project.afterEvaluate {
    compilePb()
}
```

The 'pbScript.sh' shell script for this example, located in the project's root folder:

```
#!/usr/bin/env bash
```

```
pp=/home/myself/my/proto

/usr/local/bin/protoc -I=$pp \
 --java_out=./src/main/java \
  --proto_path=$pp \
 $pp/my.proto \
 --proto_path=$pp \
 $pp/my_other.proto
```

Section 47.21: Show all gradle project tasks

```
gradlew tasks -- show all tasks



Android tasks
-------------
androidDependencies - Displays the Android dependencies of the project.
signingReport - Displays the signing info for each variant.
sourceSets - Prints out all the source sets defined in this project.

Build tasks
-----------
assemble - Assembles all variants of all applications and secondary packages.
assembleAndroidTest - Assembles all the Test applications.
assembleDebug - Assembles all Debug builds.
assembleRelease - Assembles all Release builds.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles main classes.
clean - Deletes the build directory.
compileDebugAndroidTestSources
compileDebugSources
compileDebugUnitTestSources
compileReleaseSources
compileReleaseUnitTestSources
extractDebugAnnotations - Extracts Android annotations for the debug variant into the archive file
extractReleaseAnnotations - Extracts Android annotations for the release variant into the archive
file
jar - Assembles a jar archive containing the main classes.
mockableAndroidJar - Creates a version of android.jar that is suitable for unit tests.
testClasses - Assembles test classes.

Build Setup tasks
-----------------
init - Initializes a new Gradle build. [incubating]
wrapper - Generates Gradle wrapper files. [incubating]

Documentation tasks
-------------------
javadoc - Generates Javadoc API documentation for the main source code.

Help tasks
----------
buildEnvironment - Displays all buildscript dependencies declared in root project 'LeitnerBoxPro'.
components - Displays the components produced by root project 'LeitnerBoxPro'. [incubating]
dependencies - Displays all dependencies declared in root project 'LeitnerBoxPro'.
dependencyInsight - Displays the insight into a specific dependency in root project
'LeitnerBoxPro'.
help - Displays a help message.
```

```
model - Displays the configuration model of root project 'LeitnerBoxPro'. [incubating]
projects - Displays the sub-projects of root project 'LeitnerBoxPro'.
properties - Displays the properties of root project 'LeitnerBoxPro'.
tasks - Displays the tasks runnable from root project 'LeitnerBoxPro' (some of the displayed tasks
may belong to subprojects)
.

Install tasks
-------------
installDebug - Installs the Debug build.
installDebugAndroidTest - Installs the android (on device) tests for the Debug build.
uninstallAll - Uninstall all applications.
uninstallDebug - Uninstalls the Debug build.
uninstallDebugAndroidTest - Uninstalls the android (on device) tests for the Debug build.
uninstallRelease - Uninstalls the Release build.

Verification tasks
------------------
check - Runs all checks.
connectedAndroidTest - Installs and runs instrumentation tests for all flavors on connected
devices.
connectedCheck - Runs all device checks on currently connected devices.
connectedDebugAndroidTest - Installs and runs the tests for debug on connected devices.
deviceAndroidTest - Installs and runs instrumentation tests using all Device Providers.
deviceCheck - Runs all device checks using Device Providers and Test Servers.
lint - Runs lint on all variants.
lintDebug - Runs lint on the Debug build.
lintRelease - Runs lint on the Release build.
test - Run unit tests for all variants.
testDebugUnitTest - Run unit tests for the debug build.
testReleaseUnitTest - Run unit tests for the release build.

Other tasks
-----------
assembleDefault
clean
jarDebugClasses
jarReleaseClasses
transformResourcesWithMergeJavaResForDebugUnitTest
transformResourcesWithMergeJavaResForReleaseUnitTest
```

Section 47.22: Debugging your Gradle errors

The following is an excerpt from Gradle - What is a non-zero exit value and how do I fix it?, see it for the full discussion.

Let's say you are developing an application and you get some Gradle error that appears that generally will look like so.

```
:module:someTask FAILED
FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':module:someTask'.
> some message here...  finished with non-zero exit value X
* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more
log output.
BUILD FAILED
Total time: Y.ZZ secs
```

You search here on StackOverflow for your problem, and people say to clean and rebuild your project, or enable MultiDex, and when you try that, it just isn't fixing the problem.

There are ways to get more information, but the Gradle output itself should point at the actual error in the few lines above that message between `:module:someTask FAILED` and the last `:module:someOtherTask` that passed. Therefore, if you ask a question about your error, please edit your questions to include more context to the error.

So, you get a "non-zero exit value." Well, that number is a good indicator of what you should try to fix. Here are a few occur most frequently.

- `1` is a just a general error code and the error is likely in the Gradle output
- `2` seems to be related to overlapping dependencies or project misconfiguration.
- `3` seems to be from including too many dependencies, or a memory issue.

The general solutions for the above (after attempting a Clean and Rebuild of the project) are:

- `1` - Address the error that is mentioned. Generally, this is a compile-time error, meaning some piece of code in your project is not valid. This includes both XML and Java for an Android project.
- `2` & `3` - Many answers here tell you to enable multidex. While it may fix the problem, it is most likely a workaround. If you don't understand why you are using it (see the link), you probably don't need it. General solutions involve cutting back your overuse of library dependencies (such as all of Google Play Services, when you only need to use one library, like Maps or Sign-In, for example).

Section 47.23: Use gradle.properties for central versionnumber/buildconfigurations

You can define central config info's in

- a separate gradle include file Centralizing dependencies via "dependencies.gradle" file
- a stand alone properties file Versioning your builds via "version.properties" file

or do it with root `gradle.properties` file

the project structure

```
root
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
  +- gradle.properties
```

global setting for all submodules in gradle.properties

```
# used for manifest
# todo increment for every release
appVersionCode=19
appVersionName=0.5.2.160726

# android tools settings
appCompileSdkVersion=23
appBuildToolsVersion=23.0.2
```

usage in a submodule

```
apply plugin: 'com.android.application'
android {
    // appXXX are defined in gradle.properties
    compileSdkVersion = Integer.valueOf(appCompileSdkVersion)
    buildToolsVersion = appBuildToolsVersion

    defaultConfig {
        // appXXX are defined in gradle.properties
        versionCode = Long.valueOf(appVersionCode)
        versionName = appVersionName
    }
}

dependencies {
    ...
}
```

Note: If you want to publish your app in the F-Droid app store you have to use magic numbers in the gradle file because else f-droid robot cannot read current versionnumner to detect/verify version changes.

Section 47.24: Defining build types

You can create and configure build types in the module-level `build.gradle` file inside the `android {}` block.

```
    android {
        ...
        defaultConfig {...}

        buildTypes {
            release {
                minifyEnabled true
                proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
            }

            debug {
                applicationIdSuffix ".debug"
            }
        }
    }
```

Chapter 48: FileIO with Android

Reading and writing files in Android are not different from reading and writing files in standard Java. Same `java.io` package can be used. However, there is some specific related to the folders where you are allowed to write, permissions in general and MTP work arounds.

Section 48.1: Obtaining the working folder

You can get your working folder by calling the method `getFilesDir()` on your Activity (Activity is the central class in your application that inherits from Context. See here). Reading is not different. Only your application will have access to this folder.

Your activity could contain the following code, for instance:

```
File myFolder = getFilesDir();
File myFile = new File(myFolder, "myData.bin");
```

Section 48.2: Writing raw array of bytes

```
File myFile = new File(getFilesDir(), "myData.bin");
FileOutputStream out = new FileOutputStream(myFile);

// Write four bytes one two three four:
out.write(new byte [] { 1, 2, 3, 4}
out.close()
```

There is nothing Android specific with this code. If you write lots of small values often, use BufferedOutputStream to reduce the wear of the device internal SSD.

Section 48.3: Serializing the object

The old good Java object serialization is available for you in Android. you can define Serializable classes like:

```
class Cirle implements Serializable {
  final int radius;
  final String name;

  Circle(int radius, int name) {
    this.radius = radius;
    this.name = name;
  }
}
```

and then write then to the ObjectOutputStream:

```
File myFile = new File(getFilesDir(), "myObjects.bin");
FileOutputStream out = new FileOutputStream(myFile);
ObjectOutputStream oout = new ObjectOutputStream(new BufferedOutputStream(out));

oout.writeObject(new Circle(10, "One"));
oout.writeObject(new Circle(12, "Two"));

oout.close()
```

Java object serialization may be either perfect or really bad choice, depending on what do you want to do with it -

outside the scope of this tutorial and sometimes opinion based. Read about the versioning first if you decide to use it.

Section 48.4: Writing to external storage (SD card)

You can also read and write from/to memory card (SD card) that is present in many Android devices. Files in this location can be accessed by other programs, also directly by the user after connecting device to PC via USB cable and enabling MTP protocol.

Finding the SD card location is somewhat more problematic. The Environment class contains static methods to get "external directories" that should normally be inside the SD card, also information if the SD card exists at all and is writable. This question contains valuable answers how to make sure the right location will be found.

Accessing external storage requires permissions in you Android manifest:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

For older versions of Android putting permissions it is enough to put these permissions into manifest (the user must approve during installation). However starting from Android 6.0 Android asks the user for approval at the time of the first access, and you must support this new approach. Otherwise access is denied regardless of your manifest.

In Android 6.0, first you need to check for permission, then, if not granted, request it. The code examples can be found inside this SO question.

Section 48.5: Solving "Invisible MTP files" problem

If you create files for exporting via USB cable to desktop using MTP protocol, may be a problem that newly created files are not immediately visible in the file explorer running on the connected desktop PC. To to make new files visible, you need to call MediaScannerConnection:

```
File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_DOCUMENTS), "theDocument.txt");
FileOutputStream out = new FileOutputStream(file)

... (write the document)

out.close()
MediaScannerConnection.scanFile(this, new String[] {file.getPath()}, null, null);
    context.sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
            Uri.fromFile(file)));
```

This MediaScannerConnection call code works for files only, not for directories. The problem is described in this Android bug report. This may be fixed for some version in the future, or on some devices.

Section 48.6: Working with big files

Small files are processed in a fraction of second and you can read / write them in place of the code where you need this. However if the file is bigger or otherwise slower to process, you may need to use AsyncTask in Android to work with the file in the background:

```
    class FileOperation extends AsyncTask<String, Void, File> {

        @Override
```

```
        protected File doInBackground(String... params) {
          try {
            File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_DOCUMENTS), "bigAndComplexDocument.odf");
            FileOutputStream out = new FileOutputStream(file)

           ... (write the document)

            out.close()
            return file;
            } catch (IOException ex) {
               Log.e("Unable to write", ex);
               return null;
              }
        }

        @Override
        protected void onPostExecute(File result) {
          // This is called when we finish
        }

        @Override
        protected void onPreExecute() {
           // This is called before we begin
        }

        @Override
        protected void onProgressUpdate(Void... values) {
            // Unlikely required for this example
        }
    }
}
```

and then

```
new FileOperation().execute("Some parameters");
```

This SO question contains the complete example on how to create and call the AsyncTask. Also see the question on error handling on how to handle IOExceptions and other errors.

Chapter 49: FileProvider

Section 49.1: Sharing a file

In this example you'll learn how to share a file with other apps. We'll use a pdf file in this example although the code works with every other format as well.

The roadmap:

Specify the directories in which the files you want to share are placed

To share files we'll use a FileProvider, a class allowing secure file sharing between apps. A FileProvider can only share files in predefined directories, so let's define these.

1. Create a new XML file that will contain the paths, e.g. *res/xml/filepaths.xml*
2. Add the paths

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/" />
</paths>
```

Define a FileProvider and link it with the file paths

This is done in the manifest:

```
<manifest>
    ...
    <application>
        ...
        <provider
            android:name="android.support.v4.context.FileProvider"
            android:authorities="com.mydomain.fileprovider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
    </application>
    ...
</manifest>
```

Generate the URI for the file

To share the file we must provide an identifier for the file. This is done by using a URI (Uniform Resource Identifier).

```
// We assume the file we want to load is in the documents/ subdirectory
// of the internal storage
File documentsPath = new File(Context.getFilesDir(), "documents");
File file = new File(documentsPath, "sample.pdf");
// This can also in one line of course:
// File file = new File(Context.getFilesDir(), "documents/sample.pdf");

Uri uri = FileProvider.getUriForFile(getContext(), "com.mydomain.fileprovider", file);
```

As you can see in the code we first make a new File class representing the file. To get a URI we ask FileProvider to get us one. The second argument is important: it passes the authority of a FileProvider. It must be equal to the authority of the FileProvider defined in the manifest.

Share the file with other apps

We use ShareCompat to share the file with other apps:

```
Intent intent = ShareCompat.IntentBuilder.from(getContext())
    .setType("application/pdf")
    .setStream(uri)
    .setChooserTitle("Choose bar")
    .createChooserIntent()
    .addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

Context.startActivity(intent);
```

A chooser is a menu from which the user can choose with which app he/she wants to share the file. The flag Intent.FLAG_GRANT_READ_URI_PERMISSION is needed to grant temporary read access permission to the URI.

Chapter 50: Storing Files in Internal & External Storage

Parameter	Details
name	The name of the file to open. NOTE: Cannot contain path separators
mode	Operating mode. Use `MODE_PRIVATE` for default operation, and `MODE_APPEND` to append an existing file. Other modes include `MODE_WORLD_READABLE` and `MODE_WORLD_WRITEABLE`, which were both deprecated in API 17.
dir	Directory of the file to create a new file in
path	Path to specify the location of the new file
type	Type of files directory to retrieve. Can be `null`, or any of the following: `DIRECTORY_MUSIC`, `DIRECTORY_PODCASTS`, `DIRECTORY_RINGTONES`, `DIRECTORY_ALARMS`, `DIRECTORY_NOTIFICATIONS`, `DIRECTORY_PICTURES`, or `DIRECTORY_MOVIES`

Section 50.1: Android: Internal and External Storage - Terminology Clarification

Android developers(mainly beginners) have been confused regarding Internal & External storage terminology. There are lot of questions on Stackoverflow regarding the same. This is mainly because of the fact that *terminology* according to Google/official Android documentation is quite different to that of normal Android OS user. Hence I thought documenting this would help.

What we think - User's Terminology (UT)

Internal storage(UT)	External storage(UT)
phone's inbuilt internal memory	removable Secure Digital(SD) card or micro SD storage
Example: Nexus 6P's 32 GB internal memory.	**Example:** storage space in removable SD cards provided by vendors like samsung, sandisk, strontium, transcend and others

But, **According to Android Documentation/Guide - Google's Terminology (GT)**

Internal storage(GT):

> By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user).

External storage(GT):

> This can be a removable storage media (such as an SD card) or an internal (non-removable) storage.

External Storage(GT) can be categorized into two types:

Primary External Storage	Secondary External Storage or Removable storage(GT)
This is same as phone's inbuilt internal memory (or) Internal storage(UT)	This is same as removable micro SD card storage (or) External storage(UT)
Example: Nexus 6P's 32 GB internal memory.	**Example:** storage space in removable SD cards provided by vendors like samsung, sandisk, strontium, transcend and others

This type of storage can be accessed on windows PC by connecting your phone to PC via USB cable and selecting *Camera(PTP)* in the USB options notification.	This type of storage can be accessed on windows PC by connecting your phone to PC via USB cable and selecting *File transfer* in the USB options notification.

In a nutshell,

External Storage(GT) = Internal Storage(UT) and External Storage(UT)

Removable Storage(GT) = External Storage(UT)

Internal Storage(GT) doesn't have a term in UT.

Let me explain clearly,

Internal Storage(GT): By default, files saved to the internal storage are private to your application and other applications cannot access them. Your app user also can't access them using file manager; even after enabling "show hidden files" option in file manager. To access files in Internal Storage(GT), you have to root your Android phone. Moreover, when the user uninstalls your application, these files are removed/deleted.

So Internal Storage(GT) is **NOT** what we think as Nexus 6P's 32/64 GB internal memory

Generally, **Internal Storage(GT) location** would be:
`/data/data/your.application.package.appname/someDirectory/`

External Storage(GT):

> Every Android-compatible device supports a shared "external storage" that you can use to save files. Files saved to the external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.

External Storage(GT) location: It could be *anywhere* in your internal storage(UT) or in your removable storage(GT) i.e. micro SD card. It depends on your phone's OEM and also on Android OS version.

In order to read or write files on the External Storage(GT), your app must acquire the `READ_EXTERNAL_STORAGE` or `WRITE_EXTERNAL_STORAGE` system permissions.

For example:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

> If you need to both read and write files, then you need to request only the `WRITE_EXTERNAL_STORAGE` permission, because it implicitly requires read access as well.

In **External Storage(GT)**, you may also save files that are **app-private**

But,

> When the user uninstalls your application, this directory and all its contents are deleted.

When do you need to save files that are **app-private** in **External Storage(GT)**?

> If you are handling files that are not intended for other apps to use (such as graphic textures or sound effects used by only your app), you should use a private storage directory on the external storage

> Beginning with Android 4.4, reading or writing files in your app's private directories does not require the `READ_EXTERNAL_STORAGE` or `WRITE_EXTERNAL_STORAGE` permissions. So you can declare the permission should be requested only on the lower versions of Android by adding the `maxSdkVersion` attribute:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
                     android:maxSdkVersion="18" />
    ...
</manifest
```

Methods to store in Internal Storage(GT):

Both these methods are present in Context class

```
File getDir (String name, int mode)

File getFilesDir ()
```

Methods to store in Primary External Storage i.e. Internal Storage(UT):

```
File getExternalStorageDirectory ()

File getExternalFilesDir (String type)

File getExternalStoragePublicDirectory (String type)
```

In the beginning, everyone used Environment.getExternalStorageDirectory() , which pointed to the **root** of **Primary External Storage**. As a result, Primary External Storage was filled with random content.

Later, these two methods were added:

1. In `Context` class, they added getExternalFilesDir(), pointing to an **app-specific directory** on Primary External Storage. This directory and its contents **will be deleted** when the app is uninstalled.
2. Environment.getExternalStoragePublicDirectory() for centralized places to store well-known file types, like photos and movies. This directory and its contents **will NOT be deleted** when the app is uninstalled.

Methods to store in Removable Storage(GT) i.e. micro SD card

Before **API level 19**, there was **no official way** to store in SD card. But, many could do it using unofficial libraries or APIs.

Officially, one method was introduced in `Context` class in API level 19 (Android version 4.4 - Kitkat).

```
File[] getExternalFilesDirs (String type)
```

> It returns absolute paths to application-specific directories on all shared/external storage devices where the application can place persistent files it owns. These files are internal to the application, and not typically visible to the user as media.

That means, it will return paths to **both** types of External Storage(GT) - Internal memory and Micro SD card. Generally **second path** would be storage path of micro SD card(but not always). So you need to check it out by executing the code with this method.

Example with code snippet:

I created a new android project with empty activity, wrote the following code inside

`protected void onCreate(Bundle savedInstanceState)` method of `MainActivity.java`

```
    File internal_m1 = getDir("custom", 0);
    File internal_m2 = getFilesDir();

    File external_m1 =  Environment.getExternalStorageDirectory();

    File external_m2 =  getExternalFilesDir(null);
    File external_m2_Args = getExternalFilesDir(Environment.DIRECTORY_PICTURES);

    File external_m3 =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);

    File[] external_AND_removable_storage_m1 = getExternalFilesDirs(null);
    File[] external_AND_removable_storage_m1_Args =
getExternalFilesDirs(Environment.DIRECTORY_PICTURES);
```

After executing above code,

Output:

```
internal_m1: /data/data/your.application.package.appname/app_custom

internal_m2: /data/data/your.application.package.appname/files

external_m1: /storage/emulated/0

external_m2: /storage/emulated/0/Android/data/your.application.package.appname/files

external_m2_Args: /storage/emulated/0/Android/data/your.application.package.appname/files/Pictures

external_m3: /storage/emulated/0/Pictures

external_AND_removable_storage_m1 (first path):
/storage/emulated/0/Android/data/your.application.package.appname/files

external_AND_removable_storage_m1 (second path):
/storage/sdcard1/Android/data/your.application.package.appname/files

 external_AND_removable_storage_m1_Args (first path):
/storage/emulated/0/Android/data/your.application.package.appname/files/Pictures

external_AND_removable_storage_m1_Args (second path):
/storage/sdcard1/Android/data/your.application.package.appname/files/Pictures
```

Note: I have connected my phone to Windows PC; enabled both developer options, USB debugging and then ran

this code. If you **do not connect your phone**; but instead run this on **Android emulator**, your output may vary. My phone model is Coolpad Note 3 - running on Android 5.1

Storage locations on my phone:

Micro SD storage location: `/storage/sdcard1`

Internal Storage(UT) location: `/storage/sdcard0`.

Note that `/sdcard` & `/storage/emulated/0` also point to Internal Storage(UT). But these are symlinks to `/storage/sdcard0`.

To clearly understand different storage paths in Android, Please go through this answer

Disclaimer: All the storage paths mentioned above are paths on **my** phone. Your files may **not** be stored on same storage paths. Because, the storage locations/paths may vary on other mobile phones depending on your vendor, manufacturer and different versions of Android OS.

Section 50.2: Using External Storage

"External" Storage is another type of storage that we can use to save files to the user's device. It has some key differences from "Internal" Storage, namely:

- It is not always available. In the case of a removable medium (SD card), the user can simply remove the storage.
- It is not private. The user (and other applications) have access to these files.
- If the user uninstalls the app, the files you save in the directory retrieved with `getExternalFilesDir()` will be removed.

To use External Storage, we need to first obtain the proper permissions. You will need to use:

- `android.permission.WRITE_EXTERNAL_STORAGE` for reading and writing
- `android.permission.READ_EXTERNAL_STORAGE` for just reading

To grant these permissions, you will need to identify them in your `AndroidManifest.xml` as such

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

> NOTE: Since they are Dangerous permissions if you are using **API Level 23** or above, you will need to request the **permissions at runtime**.

Before attempting to write or read from External Storage, you should always check that the storage medium is available.

```
String state = Environment.getExternalStorageState();
if (state.equals(Environment.MEDIA_MOUNTED)) {
    // Available to read and write
}
if (state.equals(Environment.MEDIA_MOUNTED) ||
        state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
    // Available to at least read
}
```

When writing files to the External Storage, you should decide if the file should be recognized as Public or Private.

While both of these types of files are still accessible to the user and other applications on the device, there is a key distinction between them.

Public files should remain on the device when the user uninstalls the app. An example of a file that should be saved as Public would be photos that are taken through your application.

Private files should all be removed when the user uninstalls the app. These types of files would be app specific, and not be of use to the user or other applications. Ex. temporary files downloaded/used by your application.

Here's how to get access to the `Documents` directory for both Public and Private files.

Public

```
// Access your app's directory in the device's Public documents directory
File docs = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_DOCUMENTS), "YourAppDirectory");
// Make the directory if it does not yet exist
myDocs.mkdirs();
```

Private

```
// Access your app's Private documents directory
File file = new File(context.getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS),
        "YourAppDirectory");
// Make the directory if it does not yet exist
myDocs.mkdirs();
```

Section 50.3: Using Internal Storage

By default, any files that you save to Internal Storage are private to your application. They cannot be accessed by other applications, nor the user under normal circumstances. **These files are deleted when the user uninstalls the application**.

To Write Text to a File

```
String fileName= "helloworld";
String textToWrite = "Hello, World!";
FileOutputStream fileOutputStream;

try {
  fileOutputStream = openFileOutput(fileName, Context.MODE_PRIVATE);
  fileOutputStream.write(textToWrite.getBytes());
  fileOutputStream.close();
} catch (Exception e) {
  e.printStackTrace();
}
```

To Append Text to an Existing File

Use `Context.MODE_APPEND` for the mode parameter of `openFileOutput`

```
fileOutputStream = openFileOutput(fileName, Context.MODE_APPEND);
```

Section 50.4: Fetch Device Directory :

First Add Storage permission to read/fetch device directory.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Create model class

```
//create one directory model class
//to store directory title and type in list

public class DirectoryModel {
    String dirName;
    int dirType; // set 1 or 0, where 0 for directory and 1 for file.

    public int getDirType() {
        return dirType;
    }

    public void setDirType(int dirType) {
        this.dirType = dirType;
    }

    public String getDirName() {
        return dirName;
    }

    public void setDirName(String dirName) {
        this.dirName = dirName;
    }
}
```

Create list using directory model to add directory data.

```
//define list to show directory

List<DirectoryModel> rootDir = new ArrayList<>();
```

Fetch directory using following method.

```
//to fetch device directory

private void getDirectory(String currDir) { // pass device root directory
        File f = new File(currDir);
        File[] files = f.listFiles();
        if (files != null) {
            if (files.length > 0) {
                rootDir.clear();
                for (File inFile : files) {
                    if (inFile.isDirectory()) { //return true if it's directory
                        // is directory
                        DirectoryModel dir = new DirectoryModel();
                        dir.setDirName(inFile.toString().replace("/storage/emulated/0", ""));
                        dir.setDirType(0); // set 0 for directory
                        rootDir.add(dir);
                    } else if (inFile.isFile()) { // return true if it's file
                        //is file
                        DirectoryModel dir = new DirectoryModel();
                        dir.setDirName(inFile.toString().replace("/storage/emulated/0", ""));
                        dir.setDirType(1); // set 1 for file
                        rootDir.add(dir);
                    }
                }
```

```
            }
            printDirectoryList();
        }
    }
```

Print directory list in log.

```
//print directory list in logs

private void printDirectoryList() {
    for (int i = 0; i < rootDir.size(); i++) {
        Log.e(TAG, "printDirectoryLogs: " + rootDir.get(i).toString());
    }
}
```

Usage

```
//to Fetch Directory Call function with root directory.

String rootPath = Environment.getExternalStorageDirectory().toString(); // return ==>
 /storage/emulated/0/
getDirectory(rootPath );
```

To fetch inner files/folder of specific directory use same method just change argument, pass the current selected path in argument and handle response for same.

To get File Extension :

```
private String getExtension(String filename) {

    String filenameArray[] = filename.split("\\.");
    String extension = filenameArray[filenameArray.length - 1];
    Log.d(TAG, "getExtension: " + extension);

    return extension;
}
```

Section 50.5: Save Database on SD Card (Backup DB on SD)

```
    public static Boolean ExportDB(String DATABASE_NAME , String packageName , String folderName){
    //DATABASE_NAME including ".db" at the end like "mayApp.db"
    String DBName = DATABASE_NAME.substring(0, DATABASE_NAME.length() - 3);
    File data = Environment.getDataDirectory();
    FileChannel source=null;
    FileChannel destination=null;
    String currentDBPath = "/data/"+ packageName +"/databases/"+DATABASE_NAME; // getting app db
path

    File sd = Environment.getExternalStorageDirectory(); // getting phone SD card path
    String backupPath = sd.getAbsolutePath() + folderName; // if you want to set backup in specific
folder name
        /* be careful , foldername must initial like this : "/myFolder" . don't forget "/" at begin
of folder name
            you could define foldername like this : "/myOutterFolder/MyInnerFolder" and so on ...
        */
    File dir = new File(backupPath);
    if(!dir.exists()) // if there was no folder at this path , it create it .
    {
        dir.mkdirs();
```

```
    }

    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
    Date date = new Date();
        /* use date including file name for arrange them and preventing to make file with the same*/
    File currentDB = new File(data, currentDBPath);
    File backupDB = new File(backupPath, DBName +"("+ dateFormat.format(date)+").db");
    try {
        if (currentDB.exists() && !backupDB.exists()) {
            source = new FileInputStream(currentDB).getChannel();
            destination = new FileOutputStream(backupDB).getChannel();
            destination.transferFrom(source, 0, source.size());
            source.close();
            destination.close();
            return true;
        }
        return false;
    } catch(IOException e) {
        e.printStackTrace();
        return false;
    }
}
```

call this method this way :

> ExportDB("myDB.db","com.example.exam","/myFolder");

Chapter 51: Zip file in android

Section 51.1: Zip file on android

```
import android.util.Log;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;


  public class Compress {
      private static final int BUFFER = 2048;

      private String[] _files;
      private String _zipFile;

      public Compress(String[] files, String zipFile) {
        _files = files;
        _zipFile = zipFile;
      }

      public void zip() {
        try  {
          BufferedInputStream origin = null;
          FileOutputStream dest = new FileOutputStream(_zipFile);

          ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(dest));

          byte data[] = new byte[BUFFER];

          for(int i=0; i < _files.length; i++) {
            Log.v("Compress", "Adding: " + _files[i]);
            FileInputStream fi = new FileInputStream(_files[i]);
            origin = new BufferedInputStream(fi, BUFFER);
            ZipEntry entry = new ZipEntry(_files[i].substring(_files[i].lastIndexOf("/") + 1));
            out.putNextEntry(entry);
            int count;
            while ((count = origin.read(data, 0, BUFFER)) != -1) {
              out.write(data, 0, count);
            }
            origin.close();
          }

          out.close();
        } catch(Exception e) {
          e.printStackTrace();
        }

      }

    }
```

Chapter 52: Unzip File in Android

Section 52.1: Unzip file

```
private boolean unpackZip(String path, String zipname){
 InputStream is;
 ZipInputStream zis;
 try
 {
     String filename;
     is = new FileInputStream(path + zipname);
     zis = new ZipInputStream(new BufferedInputStream(is));
     ZipEntry ze;
     byte[] buffer = new byte[1024];
     int count;

     while ((ze = zis.getNextEntry()) != null){
         // zapis do souboru
         filename = ze.getName();

         // Need to create directories if not exists, or
         // it will generate an Exception...
         if (ze.isDirectory()) {
            File fmd = new File(path + filename);
            fmd.mkdirs();
            continue;
         }

         FileOutputStream fout = new FileOutputStream(path + filename);

         // cteni zipu a zapis
         while ((count = zis.read(buffer)) != -1){
             fout.write(buffer, 0, count);
         }

         fout.close();
         zis.closeEntry();
     }

     zis.close();
 }
 catch(IOException e){
     e.printStackTrace();
     return false;
 }

return true;}
```

Chapter 53: Camera and Gallery

Section 53.1: Take photo

Add a permission to access the camera to the AndroidManifest file:

```
<uses-permission android:name="android.permission.CAMERA"></uses-permission>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Xml file :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<SurfaceView android:id="@+id/surfaceView" android:layout_height="0dip"
android:layout_width="0dip"></SurfaceView>
<ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
android:id="@+id/imageView"></ImageView>
</LinearLayout>
```

Activity

```
import java.io.IOException;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera;
import android.hardware.Camera.Parameters;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.ImageView;

public class TakePicture extends Activity implements SurfaceHolder.Callback
{
    //a variable to store a reference to the Image View at the main.xml file
    private ImageView iv_image;
    //a variable to store a reference to the Surface View at the main.xml file
    private SurfaceView sv;

    //a bitmap to display the captured image
    private Bitmap bmp;

    //Camera variables
    //a surface holder
    private SurfaceHolder sHolder;
    //a variable to control the camera
    private Camera mCamera;
    //the camera parameters
    private Parameters parameters;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //get the Image View at the main.xml file
        iv_image = (ImageView) findViewById(R.id.imageView);

        //get the Surface View at the main.xml file
        sv = (SurfaceView) findViewById(R.id.surfaceView);

        //Get a surface
        sHolder = sv.getHolder();

        //add the callback interface methods defined below as the Surface View callbacks
        sHolder.addCallback(this);

        //tells Android that this surface will have its data constantly replaced
        sHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    @Override
    public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3)
    {
         //get camera parameters
         parameters = mCamera.getParameters();

         //set camera parameters
         mCamera.setParameters(parameters);
         mCamera.startPreview();

         //sets what code should be executed after the picture is taken
         Camera.PictureCallback mCall = new Camera.PictureCallback()
         {
             @Override
             public void onPictureTaken(byte[] data, Camera camera)
             {
                 //decode the data obtained by the camera into a Bitmap
                 bmp = BitmapFactory.decodeByteArray(data, 0, data.length);
                String filename=Environment.getExternalStorageDirectory()
                                + File.separator + "testimage.jpg";
                FileOutputStream out = null;
                try {
                    out = new FileOutputStream(filename);
                    bmp.compress(Bitmap.CompressFormat.PNG, 100, out); // bmp is your Bitmap
instance
                    // PNG is a lossless format, the compression factor (100) is ignored
                } catch (Exception e) {
                    e.printStackTrace();
                } finally {
                    try {
                        if (out != null) {
                            out.close();
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                 //set the iv_image
                 iv_image.setImageBitmap(bmp);
             }
         };

         mCamera.takePicture(null, null, mCall);
```

```
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder)
    {
        // The Surface has been created, acquire the camera and tell it where
        // to draw the preview.
        mCamera = Camera.open();
        try {
           mCamera.setPreviewDisplay(holder);

        } catch (IOException exception) {
            mCamera.release();
            mCamera = null;
        }
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder)
    {
        //stop the preview
        mCamera.stopPreview();
        //release the camera
        mCamera.release();
        //unbind the camera from this object
        mCamera = null;
    }
}
```

Section 53.2: Taking full-sized photo from camera

To take a photo, first we need to declare required permissions in `AndroidManifest.xml`. We need two permissions:

- `Camera` - to open camera app. If attribute `required` is set to **`true`** you will not be able to install this app if you don't have hardware camera.
- `WRITE_EXTERNAL_STORAGE` - This permission is required to create new file, in which captured photo will be saved.

AndroidManifest.xml

```
<uses-feature android:name="android.hardware.camera"
         android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

The main idea in taking full-sized photo from camera is that we need to create new file for photo, before we open camera app and capture photo.

```
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException ex) {
            Log.e("DEBUG_TAG", "createFile", ex);
        }
        // Continue only if the File was successfully created
        if (photoFile != null) {
```

```
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(photoFile));
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
        }
    }
}

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(new
Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getAlbumDir();
    File image = File.createTempFile(
            imageFileName,  /* prefix */
            ".jpg",         /* suffix */
            storageDir      /* directory */
    );

    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}

private File getAlbumDir() {
    File storageDir = null;

    if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {

        storageDir = new File(Environment.getExternalStorageDirectory()
                + "/dcim/"
                + "MyRecipes");

        if (!storageDir.mkdirs()) {
            if (!storageDir.exists()) {
                Log.d("CameraSample", "failed to create directory");
                return null;
            }
        }

    } else {
        Log.v(getString(R.string.app_name), "External storage is not mounted READ/WRITE.");
    }

    return storageDir;
}

private void setPic() {

    /* There isn't enough memory to open up more than a couple camera photos */
    /* So pre-scale the target bitmap into which the file is decoded */

    /* Get the size of the ImageView */
    int targetW = recipeImage.getWidth();
    int targetH = recipeImage.getHeight();

    /* Get the size of the image */
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;
```

```
    /* Figure out which way needs to be reduced less */
    int scaleFactor = 2;
    if ((targetW > 0) && (targetH > 0)) {
        scaleFactor = Math.max(photoW / targetW, photoH / targetH);
    }

    /* Set bitmap options to scale the image decode target */
    bmOptions.inJustDecodeBounds = false;
    bmOptions.inSampleSize = scaleFactor;
    bmOptions.inPurgeable = true;

    Matrix matrix = new Matrix();
    matrix.postRotate(getRotation());

    /* Decode the JPEG file into a Bitmap */
    Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    bitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix,
false);

    /* Associate the Bitmap to the ImageView */
    recipeImage.setImageBitmap(bitmap);
}

private float getRotation() {
    try {
        ExifInterface ei = new ExifInterface(mCurrentPhotoPath);
        int orientation = ei.getAttributeInt(ExifInterface.TAG_ORIENTATION,
ExifInterface.ORIENTATION_NORMAL);

        switch (orientation) {
            case ExifInterface.ORIENTATION_ROTATE_90:
                return 90f;
            case ExifInterface.ORIENTATION_ROTATE_180:
                return 180f;
            case ExifInterface.ORIENTATION_ROTATE_270:
                return 270f;
            default:
                return 0f;
        }
    } catch (Exception e) {
        Log.e("Add Recipe", "getRotation", e);
        return 0f;
    }
}

private void galleryAddPic() {
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    File f = new File(mCurrentPhotoPath);
    Uri contentUri = Uri.fromFile(f);
    mediaScanIntent.setData(contentUri);
    sendBroadcast(mediaScanIntent);
}

private void handleBigCameraPhoto() {

    if (mCurrentPhotoPath != null) {
        setPic();
        galleryAddPic();
    }
}
```

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == Activity.RESULT_OK) {
        handleBigCameraPhoto();
    }
}
```

Section 53.3: Decode bitmap correctly rotated from the uri fetched with the intent

```
private static final String TAG = "IntentBitmapFetch";
private static final String COLON_SEPARATOR = ":";
private static final String IMAGE = "image";

@Nullable
public Bitmap getBitmap(@NonNull Uri bitmapUri, int maxDimen) {
    InputStream is = context.getContentResolver().openInputStream(bitmapUri);
    Bitmap bitmap = BitmapFactory.decodeStream(is, null, getBitmapOptions(bitmapUri, maxDimen));

    int imgRotation = getImageRotationDegrees(bitmapUri);

    int endRotation = (imgRotation < 0) ? -imgRotation : imgRotation;
    endRotation %= 360;
    endRotation = 90 * (endRotation / 90);
    if (endRotation > 0 && bitmap != null) {
        Matrix m = new Matrix();
        m.setRotate(endRotation);
        Bitmap tmp = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), m,
true);
        if (tmp != null) {
            bitmap.recycle();
            bitmap = tmp;
        }
    }

    return bitmap;
}

private BitmapFactory.Options getBitmapOptions(Uri uri, int imageMaxDimen){
    BitmapFactory.Options options = new BitmapFactory.Options();
    if (imageMaxDimen > 0) {
        options.inJustDecodeBounds = true;
        decodeImage(null, uri, options);
        options.inSampleSize = calculateScaleFactor(options, imageMaxDimen);
        options.inJustDecodeBounds = false;
        options.inPreferredConfig = Bitmap.Config.RGB_565;
        addInBitmapOptions(options);
    }
}

private int calculateScaleFactor(@NonNull BitmapFactory.Options bitmapOptionsMeasureOnly, int
imageMaxDimen) {
    int inSampleSize = 1;
    if (bitmapOptionsMeasureOnly.outHeight > imageMaxDimen || bitmapOptionsMeasureOnly.outWidth >
imageMaxDimen) {
        final int halfHeight = bitmapOptionsMeasureOnly.outHeight / 2;
        final int halfWidth = bitmapOptionsMeasureOnly.outWidth / 2;
        while ((halfHeight / inSampleSize) > imageMaxDimen && (halfWidth / inSampleSize) >
imageMaxDimen) {
            inSampleSize *= 2;
```

```
        }
    }
    return inSampleSize;
}

    public int getImageRotationDegrees(@NonNull Uri imgUri) {
    int photoRotation = ExifInterface.ORIENTATION_UNDEFINED;

    try {
        boolean hasRotation = false;
        //If image comes from the gallery and is not in the folder DCIM (Scheme: content://)
        String[] projection = {MediaStore.Images.ImageColumns.ORIENTATION};
        Cursor cursor = context.getContentResolver().query(imgUri, projection, null, null, null);
        if (cursor != null) {
            if (cursor.getColumnCount() > 0 && cursor.moveToFirst()) {
                photoRotation = cursor.getInt(cursor.getColumnIndex(projection[0]));
                hasRotation = photoRotation != 0;
                Log.d("Cursor orientation: "+ photoRotation);
            }
            cursor.close();
        }

        //If image comes from the camera (Scheme: file://) or is from the folder DCIM (Scheme:
content://)
        if (!hasRotation) {
            ExifInterface exif = new ExifInterface(getAbsolutePath(imgUri));
            int exifRotation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION,
                    ExifInterface.ORIENTATION_NORMAL);
            switch (exifRotation) {
                case ExifInterface.ORIENTATION_ROTATE_90: {
                    photoRotation = 90;
                    break;
                }
                case ExifInterface.ORIENTATION_ROTATE_180: {
                    photoRotation = 180;
                    break;
                }
                case ExifInterface.ORIENTATION_ROTATE_270: {
                    photoRotation = 270;
                    break;
                }
            }
            Log.d(TAG, "Exif orientation: "+ photoRotation);
        }
    } catch (IOException e) {
        Log.e(TAG, "Error determining rotation for image"+ imgUri, e);
    }
    return photoRotation;
}

@TargetApi(Build.VERSION_CODES.KITKAT)
private String getAbsolutePath(Uri uri) {
    //Code snippet edited from: http://stackoverflow.com/a/20559418/2235133
    String filePath = uri.getPath();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(context, uri)) {
        // Will return "image:x*"
        String[] wholeID = TextUtils.split(DocumentsContract.getDocumentId(uri), COLON_SEPARATOR);
        // Split at colon, use second item in the array
        String type = wholeID[0];
        if (IMAGE.equalsIgnoreCase(type)) {//If it not type image, it means it comes from a remote
location, like Google Photos
```

```
            String id = wholeID[1];
            String[] column = {MediaStore.Images.Media.DATA};
            // where id is equal to
            String sel = MediaStore.Images.Media._ID + "=?";
            Cursor cursor = context.getContentResolver().
                    query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
                            column, sel, new String[]{id}, null);
            if (cursor != null) {
                int columnIndex = cursor.getColumnIndex(column[0]);
                if (cursor.moveToFirst()) {
                    filePath = cursor.getString(columnIndex);
                }
                cursor.close();
            }
            Log.d(TAG, "Fetched absolute path for uri" + uri);
        }
    }
    return filePath;
}
```

Section 53.4: Set camera resolution

Set High resolution programmatically.

```
Camera mCamera = Camera.open();
Camera.Parameters params = mCamera.getParameters();

// Check what resolutions are supported by your camera
List<Size> sizes = params.getSupportedPictureSizes();

// Iterate through all available resolutions and choose one.
// The chosen resolution will be stored in mSize.
Size mSize;
for (Size size : sizes) {
    Log.i(TAG, "Available resolution: "+size.width+" "+size.height);
        mSize = size;
    }
}

Log.i(TAG, "Chosen resolution: "+mSize.width+" "+mSize.height);
params.setPictureSize(mSize.width, mSize.height);
mCamera.setParameters(params);
```

Section 53.5: How to start camera or gallery and save camera result to storage

First of all you need `Uri` and temp Folders and request codes :

```
public final int REQUEST_SELECT_PICTURE = 0x01;
public final int REQUEST_CODE_TAKE_PICTURE = 0x2;
public static String TEMP_PHOTO_FILE_NAME ="photo_";
Uri mImageCaptureUri;
File mFileTemp;
```

Then init mFileTemp :

```
public void initTempFile(){
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
```

```
        mFileTemp = new File(Environment.getExternalStorageDirectory() + File.separator
                + getResources().getString(R.string.app_foldername) + File.separator
                + getResources().getString(R.string.pictures_folder)
                , TEMP_PHOTO_FILE_NAME
                + System.currentTimeMillis() + ".jpg");
        mFileTemp.getParentFile().mkdirs();
    } else {
        mFileTemp = new File(getFilesDir() + File.separator
                + getResources().getString(R.string.app_foldername)
                + File.separator + getResources().getString(R.string.pictures_folder)
                , TEMP_PHOTO_FILE_NAME + System.currentTimeMillis() + ".jpg");
        mFileTemp.getParentFile().mkdirs();
    }
}
```

Opening `Camera` and `Gallery` intents :

```
public void openCamera(){
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    try {
        mImageCaptureUri = null;
        String state = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(state)) {
            mImageCaptureUri = Uri.fromFile(mFileTemp);

        } else {

            mImageCaptureUri = InternalStorageContentProvider.CONTENT_URI;

        }
        intent.putExtra(MediaStore.EXTRA_OUTPUT, mImageCaptureUri);
        intent.putExtra("return-data", true);
        startActivityForResult(intent, REQUEST_CODE_TAKE_PICTURE);
    } catch (Exception e) {

        Log.d("error", "cannot take picture", e);
    }
}

public void openGallery(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN
            && ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {
        requestPermission(Manifest.permission.READ_EXTERNAL_STORAGE,
                getString(R.string.permission_read_storage_rationale),
                REQUEST_STORAGE_READ_ACCESS_PERMISSION);
    } else {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        startActivityForResult(Intent.createChooser(intent, getString(R.string.select_image)),
REQUEST_SELECT_PICTURE);
    }

}
```

Then in `onActivityResult` method :

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
    if (resultCode != RESULT_OK) {
        return;
    }
    Bitmap bitmap;

    switch (requestCode) {

        case REQUEST_SELECT_PICTURE:
            try {
                Uri uri = data.getData();
                try {
                    bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), uri);
                    Bitmap bitmapScaled = Bitmap.createScaledBitmap(bitmap, 800, 800, true);
                    Drawable drawable=new BitmapDrawable(bitmapScaled);
                    mImage.setImageDrawable(drawable);
                    mImage.setVisibility(View.VISIBLE);
                } catch (IOException e) {
                    Log.v("act result", "there is an error : "+e.getContent());
                }
            } catch (Exception e) {
                Log.v("act result", "there is an error : "+e.getContent());
            }
            break;
        case REQUEST_CODE_TAKE_PICTURE:
            try{
               Bitmap bitmappicture = MediaStore.Images.Media.getBitmap(getContentResolver() ,
mImageCaptureUri);
               mImage.setImageBitmap(bitmappicture);
               mImage.setVisibility(View.VISIBLE);
           }catch (IOException e){
              Log.v("error camera",e.getMessage());
           }
           break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

You need theese permissions in `AndroidManifest.xml` :

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
```

And you need to handle runtime permissions such as Read/Write external storage etc ...

I am checking `READ_EXTERNAL_STORAGE` permission in my `openGallery` method :

My `requestPermission` method :

```
protected void requestPermission(final String permission, String rationale, final int requestCode)
{
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, permission)) {
        showAlertDialog(getString(R.string.permission_title_rationale), rationale,
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        ActivityCompat.requestPermissions(BasePermissionActivity.this,
                                new String[]{permission}, requestCode);
                    }
                }, getString(android.R.string.ok), null, getString(android.R.string.cancel));
    } else {
```

```
        ActivityCompat.requestPermissions(this, new String[]{permission}, requestCode);
    }
}
```

Then Override `onRequestPermissionsResult` method :

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
    switch (requestCode) {
        case REQUEST_STORAGE_READ_ACCESS_PERMISSION:
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                handleGallery();
            }
            break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

`showAlertDialog` method :

```
protected void showAlertDialog(@Nullable String title, @Nullable String message,
                               @Nullable DialogInterface.OnClickListener
onPositiveButtonClickListener,
                               @NonNull String positiveText,
                               @Nullable DialogInterface.OnClickListener
onNegativeButtonClickListener,
                               @NonNull String negativeText) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.setPositiveButton(positiveText, onPositiveButtonClickListener);
    builder.setNegativeButton(negativeText, onNegativeButtonClickListener);
    mAlertDialog = builder.show();
}
```

Chapter 54: Camera 2 API

Parameter	Details
`CameraCaptureSession`	A configured capture session for a `CameraDevice`, used for capturing images from the camera or reprocessing images captured from the camera in the same session previously
`CameraDevice`	A representation of a single camera connected to an Android device
`CameraCharacteristics`	The properties describing a CameraDevice. These properties are fixed for a given CameraDevice, and can be queried through the CameraManager interface with `getCameraCharacteristics(String)`
`CameraManager`	A system service manager for detecting, characterizing, and connecting to `CameraDevices`. You can get an instance of this class by calling `Context.getSystemService()`
`CaptureRequest`	An immutable package of settings and outputs needed to capture a single image from the camera device. Contains the configuration for the capture hardware (sensor, lens, flash), the processing pipeline, the control algorithms, and the output buffers. Also contains the list of target Surfaces to send image data to for this capture. Can be created by using a `CaptureRequest.Builder` instance, obtained by calling `createCaptureRequest(int)`
`CaptureResult`	The subset of the results of a single image capture from the image sensor. Contains a subset of the final configuration for the capture hardware (sensor, lens, flash), the processing pipeline, the control algorithms, and the output buffers. It is produced by a `CameraDevice` after processing a `CaptureRequest`

Section 54.1: Preview the main camera in a TextureView

In this case, building against API 23, so permissions are handled too.

You must add in the Manifest the following permission (wherever the API level you're using):

```
<uses-permission android:name="android.permission.CAMERA"/>
```

We're about to create an activity (Camera2Activity.java) that fills a `TextureView` with the preview of the device's camera.

The Activity we're going to use is a typical AppCompatActivity:

```
public class Camera2Activity extends AppCompatActivity {
```

Attributes (You may need to read the entire example to understand some of it)

The `MAX_PREVIEW_SIZE` guaranteed by Camera2 API is 1920x1080

```
private static final int MAX_PREVIEW_WIDTH = 1920;
private static final int MAX_PREVIEW_HEIGHT = 1080;
```

`TextureView.SurfaceTextureListener` handles several lifecycle events on a `TextureView`. In this case, we're listening to those events. When the SurfaceTexture is ready, we initialize the camera. When it size changes, we setup the preview coming from the camera accordingly

```
private final TextureView.SurfaceTextureListener mSurfaceTextureListener
        = new TextureView.SurfaceTextureListener() {

    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture texture, int width, int height) {
        openCamera(width, height);
    }
```

```
    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture texture, int width, int height) {
        configureTransform(width, height);
    }

    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture texture) {
        return true;
    }

    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture texture) {
    }

};
```

A `CameraDevice` represent one physical device's camera. In this attribute, we save the ID of the current `CameraDevice`

```
private String mCameraId;
```

This is the view (`TextureView`) that we'll be using to "draw" the preview of the Camera

```
private TextureView mTextureView;
```

The `CameraCaptureSession` for camera preview

```
private CameraCaptureSession mCaptureSession;
```

A reference to the opened `CameraDevice`

```
private CameraDevice mCameraDevice;
```

The `Size` of camera preview.

```
private Size mPreviewSize;
```

`CameraDevice.StateCallback` is called when `CameraDevice` changes its state

```
private final CameraDevice.StateCallback mStateCallback = new CameraDevice.StateCallback() {

    @Override
    public void onOpened(@NonNull CameraDevice cameraDevice) {
        // This method is called when the camera is opened.  We start camera preview here.
        mCameraOpenCloseLock.release();
        mCameraDevice = cameraDevice;
        createCameraPreviewSession();
    }

    @Override
    public void onDisconnected(@NonNull CameraDevice cameraDevice) {
        mCameraOpenCloseLock.release();
        cameraDevice.close();
        mCameraDevice = null;
    }

    @Override
    public void onError(@NonNull CameraDevice cameraDevice, int error) {
```

```
        mCameraOpenCloseLock.release();
        cameraDevice.close();
        mCameraDevice = null;
        finish();
    }

};
```

An additional thread for running tasks that shouldn't block the UI

```
private HandlerThread mBackgroundThread;
```

A `Handler` for running tasks in the background

```
private Handler mBackgroundHandler;
```

An `ImageReader` that handles still image capture

```
private ImageReader mImageReader;
```

`CaptureRequest.Builder` for the camera preview

```
private CaptureRequest.Builder mPreviewRequestBuilder;
```

`CaptureRequest` generated by `mPreviewRequestBuilder`

```
private CaptureRequest mPreviewRequest;
```

A `Semaphore` to prevent the app from exiting before closing the camera.

```
private Semaphore mCameraOpenCloseLock = new Semaphore(1);
```

Constant ID of the permission request

```
private static final int REQUEST_CAMERA_PERMISSION = 1;
```

Android Lifecycle methods

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera2);

    mTextureView = (TextureView) findViewById(R.id.texture);
}

@Override
public void onResume() {
    super.onResume();
    startBackgroundThread();

    // When the screen is turned off and turned back on, the SurfaceTexture is already
    // available, and "onSurfaceTextureAvailable" will not be called. In that case, we can open
    // a camera and start preview from here (otherwise, we wait until the surface is ready in
    // the SurfaceTextureListener).
    if (mTextureView.isAvailable()) {
        openCamera(mTextureView.getWidth(), mTextureView.getHeight());
```

```
    } else {
        mTextureView.setSurfaceTextureListener(mSurfaceTextureListener);
    }
}

@Override
public void onPause() {
    closeCamera();
    stopBackgroundThread();
    super.onPause();
}
```

Camera2 related methods

Those are methods that uses the Camera2 APIs

```
private void openCamera(int width, int height) {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
            != PackageManager.PERMISSION_GRANTED) {
        requestCameraPermission();
        return;
    }
    setUpCameraOutputs(width, height);
    configureTransform(width, height);
    CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
    try {
        if (!mCameraOpenCloseLock.tryAcquire(2500, TimeUnit.MILLISECONDS)) {
            throw new RuntimeException("Time out waiting to lock camera opening.");
        }
        manager.openCamera(mCameraId, mStateCallback, mBackgroundHandler);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        throw new RuntimeException("Interrupted while trying to lock camera opening.", e);
    }
}
```

Closes the current camera

```
private void closeCamera() {
    try {
        mCameraOpenCloseLock.acquire();
        if (null != mCaptureSession) {
            mCaptureSession.close();
            mCaptureSession = null;
        }
        if (null != mCameraDevice) {
            mCameraDevice.close();
            mCameraDevice = null;
        }
        if (null != mImageReader) {
            mImageReader.close();
            mImageReader = null;
        }
    } catch (InterruptedException e) {
        throw new RuntimeException("Interrupted while trying to lock camera closing.", e);
    } finally {
        mCameraOpenCloseLock.release();
    }
}
```

Sets up member variables related to camera

```
private void setUpCameraOutputs(int width, int height) {
    CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
    try {
        for (String cameraId : manager.getCameraIdList()) {
            CameraCharacteristics characteristics
                    = manager.getCameraCharacteristics(cameraId);

            // We don't use a front facing camera in this sample.
            Integer facing = characteristics.get(CameraCharacteristics.LENS_FACING);
            if (facing != null && facing == CameraCharacteristics.LENS_FACING_FRONT) {
                continue;
            }

            StreamConfigurationMap map = characteristics.get(
                    CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
            if (map == null) {
                continue;
            }

            // For still image captures, we use the largest available size.
            Size largest = Collections.max(
                    Arrays.asList(map.getOutputSizes(ImageFormat.JPEG)),
                    new CompareSizesByArea());
            mImageReader = ImageReader.newInstance(largest.getWidth(), largest.getHeight(),
                    ImageFormat.JPEG, /*maxImages*/2);
            mImageReader.setOnImageAvailableListener(
                    null, mBackgroundHandler);

            Point displaySize = new Point();
            getWindowManager().getDefaultDisplay().getSize(displaySize);
            int rotatedPreviewWidth = width;
            int rotatedPreviewHeight = height;
            int maxPreviewWidth = displaySize.x;
            int maxPreviewHeight = displaySize.y;

            if (maxPreviewWidth > MAX_PREVIEW_WIDTH) {
                maxPreviewWidth = MAX_PREVIEW_WIDTH;
            }

            if (maxPreviewHeight > MAX_PREVIEW_HEIGHT) {
                maxPreviewHeight = MAX_PREVIEW_HEIGHT;
            }

            // Danger! Attempting to use too large a preview size could  exceed the camera
            // bus' bandwidth limitation, resulting in gorgeous previews but the storage of
            // garbage capture data.
            mPreviewSize = chooseOptimalSize(map.getOutputSizes(SurfaceTexture.class),
                    rotatedPreviewWidth, rotatedPreviewHeight, maxPreviewWidth,
                    maxPreviewHeight, largest);

            mCameraId = cameraId;
            return;
        }
    } catch (CameraAccessException e) {
        e.printStackTrace();
    } catch (NullPointerException e) {
        // Currently an NPE is thrown when the Camera2API is used but not supported on the
        // device this code runs.
        Toast.makeText(Camera2Activity.this, "Camera2 API not supported on this device",
Toast.LENGTH_LONG).show();
```

```
    }
}
```

Creates a new `CameraCaptureSession` for camera preview

```
private void createCameraPreviewSession() {
    try {
        SurfaceTexture texture = mTextureView.getSurfaceTexture();
        assert texture != null;

        // We configure the size of default buffer to be the size of camera preview we want.
        texture.setDefaultBufferSize(mPreviewSize.getWidth(), mPreviewSize.getHeight());

        // This is the output Surface we need to start preview.
        Surface surface = new Surface(texture);

        // We set up a CaptureRequest.Builder with the output Surface.
        mPreviewRequestBuilder
                = mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
        mPreviewRequestBuilder.addTarget(surface);

        // Here, we create a CameraCaptureSession for camera preview.
        mCameraDevice.createCaptureSession(Arrays.asList(surface, mImageReader.getSurface()),
                new CameraCaptureSession.StateCallback() {

                    @Override
                    public void onConfigured(@NonNull CameraCaptureSession cameraCaptureSession) {
                        // The camera is already closed
                        if (null == mCameraDevice) {
                            return;
                        }

                        // When the session is ready, we start displaying the preview.
                        mCaptureSession = cameraCaptureSession;
                        try {
                            // Auto focus should be continuous for camera preview.
                            mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AF_MODE,
                                    CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);

                            // Finally, we start displaying the camera preview.
                            mPreviewRequest = mPreviewRequestBuilder.build();
                            mCaptureSession.setRepeatingRequest(mPreviewRequest,
                                    null, mBackgroundHandler);
                        } catch (CameraAccessException e) {
                            e.printStackTrace();
                        }
                    }

                    @Override
                    public void onConfigureFailed(
                            @NonNull CameraCaptureSession cameraCaptureSession) {
                        showToast("Failed");
                    }
                }, null
        );
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
}
```

Permissions related methods For Android API 23+

```
private void requestCameraPermission() {
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CAMERA)) {
        new AlertDialog.Builder(Camera2Activity.this)
                .setMessage("R string request permission")
                .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        ActivityCompat.requestPermissions(Camera2Activity.this,
                                new String[]{Manifest.permission.CAMERA},
                                REQUEST_CAMERA_PERMISSION);
                    }
                })
                .setNegativeButton(android.R.string.cancel,
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                finish();

                            }
                        })
                .create();

    } else {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
                REQUEST_CAMERA_PERMISSION);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length != 1 || grantResults[0] != PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(Camera2Activity.this, "ERROR: Camera permissions not granted",
Toast.LENGTH_LONG).show();
        }
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

Background thread / handler methods

```
private void startBackgroundThread() {
    mBackgroundThread = new HandlerThread("CameraBackground");
    mBackgroundThread.start();
    mBackgroundHandler = new Handler(mBackgroundThread.getLooper());
}

private void stopBackgroundThread() {
    mBackgroundThread.quitSafely();
    try {
        mBackgroundThread.join();
        mBackgroundThread = null;
        mBackgroundHandler = null;
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Utility methods

Given choices of Sizes supported by a camera, choose the smallest one that is at least at large as the respective texture view size, and that is as most as large as the respective max size, and whose aspect ratio matches with the specified value. If doesn't exist, choose the largest one that is at most as large as the respective max size, and whose aspect ratio matches with the specified value

```
private static Size chooseOptimalSize(Size[] choices, int textureViewWidth,
                                      int textureViewHeight, int maxWidth, int maxHeight, Size
aspectRatio) {

    // Collect the supported resolutions that are at least as big as the preview Surface
    List<Size> bigEnough = new ArrayList<>();
    // Collect the supported resolutions that are smaller than the preview Surface
    List<Size> notBigEnough = new ArrayList<>();
    int w = aspectRatio.getWidth();
    int h = aspectRatio.getHeight();
    for (Size option : choices) {
        if (option.getWidth() <= maxWidth && option.getHeight() <= maxHeight &&
                option.getHeight() == option.getWidth() * h / w) {
            if (option.getWidth() >= textureViewWidth &&
                    option.getHeight() >= textureViewHeight) {
                bigEnough.add(option);
            } else {
                notBigEnough.add(option);
            }
        }
    }

    // Pick the smallest of those big enough. If there is no one big enough, pick the
    // largest of those not big enough.
    if (bigEnough.size() > 0) {
        return Collections.min(bigEnough, new CompareSizesByArea());
    } else if (notBigEnough.size() > 0) {
        return Collections.max(notBigEnough, new CompareSizesByArea());
    } else {
        Log.e("Camera2", "Couldn't find any suitable preview size");
        return choices[0];
    }
}
```

This method congfigures the neccesary `Matrix` transformation to `mTextureView`

```
private void configureTransform(int viewWidth, int viewHeight) {
    if (null == mTextureView || null == mPreviewSize) {
        return;
    }
    int rotation = getWindowManager().getDefaultDisplay().getRotation();
    Matrix matrix = new Matrix();
    RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);
    RectF bufferRect = new RectF(0, 0, mPreviewSize.getHeight(), mPreviewSize.getWidth());
    float centerX = viewRect.centerX();
    float centerY = viewRect.centerY();
    if (Surface.ROTATION_90 == rotation || Surface.ROTATION_270 == rotation) {
        bufferRect.offset(centerX - bufferRect.centerX(), centerY - bufferRect.centerY());
        matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.FILL);
        float scale = Math.max(
                (float) viewHeight / mPreviewSize.getHeight(),
                (float) viewWidth / mPreviewSize.getWidth());
        matrix.postScale(scale, scale, centerX, centerY);
        matrix.postRotate(90 * (rotation - 2), centerX, centerY);
    } else if (Surface.ROTATION_180 == rotation) {
```

```
        matrix.postRotate(180, centerX, centerY);
    }
    mTextureView.setTransform(matrix);
}
```

This method compares two `Sizes` based on their areas.

```
static class CompareSizesByArea implements Comparator<Size> {

    @Override
    public int compare(Size lhs, Size rhs) {
        // We cast here to ensure the multiplications won't overflow
        return Long.signum((long) lhs.getWidth() * lhs.getHeight() -
                (long) rhs.getWidth() * rhs.getHeight());
    }
}
```

Not much to see here

```
/**
 * Shows a {@link Toast} on the UI thread.
 *
 * @param text The message to show
 */
private void showToast(final String text) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            Toast.makeText(Camera2Activity.this, text, Toast.LENGTH_SHORT).show();
        }
    });
}
```

Chapter 55: Fingerprint API in android

Section 55.1: How to use Android Fingerprint API to save user passwords

This example helper class interacts with the finger print manager and performs encryption and decryption of password. Please note that the method used for encryption in this example is AES. This is not the only way to encrypt and other examples exist. In this example the data is encrypted and decrypted in the following manner:

Encryption:

1. User gives helper the desired non-encrypted password.
2. User is required to provide fingerprint.
3. Once authenticated, the helper obtains a key from the `KeyStore` and encrypts the password using a `Cipher`.
4. Password and IV salt (IV is recreated for every encryption and is not reused) are saved to shared preferences to be used later in the decryption process.

Decryption:

1. User requests to decrypt the password.
2. User is required to provide fingerprint.
3. The helper builds a `Cipher` using the IV and once user is authenticated, the KeyStore obtains a key from the `KeyStore` and deciphers the password.

```
public class FingerPrintAuthHelper {

    private static final String FINGER_PRINT_HELPER = "FingerPrintAuthHelper";
    private static final String ENCRYPTED_PASS_SHARED_PREF_KEY = "ENCRYPTED_PASS_SHARED_PREF_KEY";
    private static final String LAST_USED_IV_SHARED_PREF_KEY = "LAST_USED_IV_SHARED_PREF_KEY";
    private static final String MY_APP_ALIAS = "MY_APP_ALIAS";

    private KeyguardManager keyguardManager;
    private FingerprintManager fingerprintManager;

    private final Context context;
    private KeyStore keyStore;
    private KeyGenerator keyGenerator;

    private String lastError;


    public interface Callback {
        void onSuccess(String savedPass);

        void onFailure(String message);

        void onHelp(int helpCode, String helpString);
    }

    public FingerPrintAuthHelper(Context context) {
        this.context = context;
    }

    public String getLastError() {
        return lastError;
    }
```

```
    @TargetApi(Build.VERSION_CODES.M)
    public boolean init() {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
            setError("This Android version does not support fingerprint authentication");
            return false;
        }

        keyguardManager = (KeyguardManager) context.getSystemService(KEYGUARD_SERVICE);
        fingerprintManager = (FingerprintManager) context.getSystemService(FINGERPRINT_SERVICE);

        if (!keyguardManager.isKeyguardSecure()) {
            setError("User hasn't enabled Lock Screen");
            return false;
        }

        if (!hasPermission()) {
            setError("User hasn't granted permission to use Fingerprint");
            return false;
        }

        if (!fingerprintManager.hasEnrolledFingerprints()) {
            setError("User hasn't registered any fingerprints");
            return false;
        }

        if (!initKeyStore()) {
            return false;
        }
        return false;
    }

    @Nullable
    @RequiresApi(api = Build.VERSION_CODES.M)
    private Cipher createCipher(int mode) throws NoSuchPaddingException, NoSuchAlgorithmException,
UnrecoverableKeyException, KeyStoreException, InvalidKeyException,
InvalidAlgorithmParameterException {
        Cipher cipher = Cipher.getInstance(KeyProperties.KEY_ALGORITHM_AES + "/" +
                KeyProperties.BLOCK_MODE_CBC + "/" +
                KeyProperties.ENCRYPTION_PADDING_PKCS7);

        Key key = keyStore.getKey(MY_APP_ALIAS, null);
        if (key == null) {
            return null;
        }
        if(mode == Cipher.ENCRYPT_MODE) {
            cipher.init(mode, key);
            byte[] iv = cipher.getIV();
            saveIv(iv);
        } else {
            byte[] lastIv = getLastIv();
            cipher.init(mode, key, new IvParameterSpec(lastIv));
        }
        return cipher;
    }

    @NonNull
    @RequiresApi(api = Build.VERSION_CODES.M)
    private KeyGenParameterSpec createKeyGenParameterSpec() {
        return new KeyGenParameterSpec.Builder(MY_APP_ALIAS, KeyProperties.PURPOSE_ENCRYPT |
KeyProperties.PURPOSE_DECRYPT)
                        .setBlockModes(KeyProperties.BLOCK_MODE_CBC)
                        .setUserAuthenticationRequired(true)
```

```
                        .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_PKCS7)
                        .build();
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    private boolean initKeyStore() {
        try {
            keyStore = KeyStore.getInstance("AndroidKeyStore");
            keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES,
"AndroidKeyStore");
            keyStore.load(null);
            if (getLastIv() == null) {
                KeyGenParameterSpec keyGeneratorSpec = createKeyGenParameterSpec();
                keyGenerator.init(keyGeneratorSpec);
                keyGenerator.generateKey();
            }
        } catch (Throwable t) {
            setError("Failed init of keyStore & keyGenerator: " + t.getMessage());
            return false;
        }
        return true;
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    private void authenticate(CancellationSignal cancellationSignal,
FingerPrintAuthenticationListener authListener, int mode) {
        try {
            if (hasPermission()) {
                Cipher cipher = createCipher(mode);
                FingerprintManager.CryptoObject crypto = new
FingerprintManager.CryptoObject(cipher);
                fingerprintManager.authenticate(crypto, cancellationSignal, 0, authListener, null);
            } else {
                authListener.getCallback().onFailure("User hasn't granted permission to use
Fingerprint");
            }
        } catch (Throwable t) {
            authListener.getCallback().onFailure("An error occurred: " + t.getMessage());
        }
    }

    private String getSavedEncryptedPassword() {
        SharedPreferences sharedPreferences = getSharedPreferences();
        if (sharedPreferences != null) {
            return sharedPreferences.getString(ENCRYPTED_PASS_SHARED_PREF_KEY, null);
        }
        return null;
    }

    private void saveEncryptedPassword(String encryptedPassword) {
        SharedPreferences.Editor edit = getSharedPreferences().edit();
        edit.putString(ENCRYPTED_PASS_SHARED_PREF_KEY, encryptedPassword);
        edit.commit();
    }

    private byte[] getLastIv() {
        SharedPreferences sharedPreferences = getSharedPreferences();
        if (sharedPreferences != null) {
            String ivString = sharedPreferences.getString(LAST_USED_IV_SHARED_PREF_KEY, null);

            if (ivString != null) {
                return decodeBytes(ivString);
```

```
            }
        }
        return null;
    }

    private void saveIv(byte[] iv) {
        SharedPreferences.Editor edit = getSharedPreferences().edit();
        String string = encodeBytes(iv);
        edit.putString(LAST_USED_IV_SHARED_PREF_KEY, string);
        edit.commit();
    }

    private SharedPreferences getSharedPreferences() {
        return context.getSharedPreferences(FINGER_PRINT_HELPER, 0);
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    private boolean hasPermission() {
        return ActivityCompat.checkSelfPermission(context, Manifest.permission.USE_FINGERPRINT) ==
PackageManager.PERMISSION_GRANTED;
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    public void savePassword(@NonNull String password, CancellationSignal cancellationSignal,
Callback callback) {
        authenticate(cancellationSignal, new FingerPrintEncryptPasswordListener(callback,
password), Cipher.ENCRYPT_MODE);
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    public void getPassword(CancellationSignal cancellationSignal, Callback callback) {
        authenticate(cancellationSignal, new FingerPrintDecryptPasswordListener(callback),
Cipher.DECRYPT_MODE);
    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    public boolean encryptPassword(Cipher cipher, String password) {
        try {
            // Encrypt the text
            if(password.isEmpty()) {
                setError("Password is empty");
                return false;
            }

            if (cipher == null) {
                setError("Could not create cipher");
                return false;
            }

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            CipherOutputStream cipherOutputStream = new CipherOutputStream(outputStream, cipher);
            byte[] bytes = password.getBytes(Charset.defaultCharset());
            cipherOutputStream.write(bytes);
            cipherOutputStream.flush();
            cipherOutputStream.close();
            saveEncryptedPassword(encodeBytes(outputStream.toByteArray()));
        } catch (Throwable t) {
            setError("Encryption failed " + t.getMessage());
            return false;
        }

        return true;
```

```
    }

    private byte[] decodeBytes(String s) {
        final int len = s.length();

        // "111" is not a valid hex encoding.
        if( len%2 != 0 )
            throw new IllegalArgumentException("hexBinary needs to be even-length: "+s);

        byte[] out = new byte[len/2];

        for( int i=0; i<len; i+=2 ) {
            int h = hexToBin(s.charAt(i  ));
            int l = hexToBin(s.charAt(i+1));
            if( h==-1 || l==-1 )
                throw new IllegalArgumentException("contains illegal character for hexBinary: "+s);

            out[i/2] = (byte)(h*16+l);
        }

        return out;
    }

    private static int hexToBin( char ch ) {
        if( '0'<=ch && ch<='9' )    return ch-'0';
        if( 'A'<=ch && ch<='F' )    return ch-'A'+10;
        if( 'a'<=ch && ch<='f' )    return ch-'a'+10;
        return -1;
    }

    private static final char[] hexCode = "0123456789ABCDEF".toCharArray();

    public String encodeBytes(byte[] data) {
        StringBuilder r = new StringBuilder(data.length*2);
        for ( byte b : data) {
            r.append(hexCode[(b >> 4) & 0xF]);
            r.append(hexCode[(b & 0xF)]);
        }
        return r.toString();
    }

    @NonNull
    private String decipher(Cipher cipher) throws IOException, IllegalBlockSizeException,
BadPaddingException {
        String retVal = null;
        String savedEncryptedPassword = getSavedEncryptedPassword();
        if (savedEncryptedPassword != null) {
            byte[] decodedPassword = decodeBytes(savedEncryptedPassword);
            CipherInputStream cipherInputStream = new CipherInputStream(new
ByteArrayInputStream(decodedPassword), cipher);

            ArrayList<Byte> values = new ArrayList<>();
            int nextByte;
            while ((nextByte = cipherInputStream.read()) != -1) {
                values.add((byte) nextByte);
            }
            cipherInputStream.close();

            byte[] bytes = new byte[values.size()];
            for (int i = 0; i < values.size(); i++) {
                bytes[i] = values.get(i).byteValue();
            }
```

```
            retVal = new String(bytes, Charset.defaultCharset());
        }
        return retVal;
    }

    private void setError(String error) {
        lastError = error;
        Log.w(FINGER_PRINT_HELPER, lastError);
    }

    @RequiresApi(Build.VERSION_CODES.M)
    protected class FingerPrintAuthenticationListener extends
FingerprintManager.AuthenticationCallback {

        protected final Callback callback;

        public FingerPrintAuthenticationListener(@NonNull Callback callback) {
            this.callback = callback;
        }

        public void onAuthenticationError(int errorCode, CharSequence errString) {
            callback.onFailure("Authentication error [" + errorCode + "] " + errString);
        }

        /**
         * Called when a recoverable error has been encountered during authentication. The help
         * string is provided to give the user guidance for what went wrong, such as
         * "Sensor dirty, please clean it."
         * @param helpCode An integer identifying the error message
         * @param helpString A human-readable string that can be shown in UI
         */
        public void onAuthenticationHelp(int helpCode, CharSequence helpString) {
            callback.onHelp(helpCode, helpString.toString());
        }

        /**
         * Called when a fingerprint is recognized.
         * @param result An object containing authentication-related data
         */
        public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
        }

        /**
         * Called when a fingerprint is valid but not recognized.
         */
        public void onAuthenticationFailed() {
            callback.onFailure("Authentication failed");
        }

        public @NonNull
        Callback getCallback() {
            return callback;
        }

    }

    @RequiresApi(api = Build.VERSION_CODES.M)
    private class FingerPrintEncryptPasswordListener extends FingerPrintAuthenticationListener {

        private final String password;

        public FingerPrintEncryptPasswordListener(Callback callback, String password) {
```

```
            super(callback);
            this.password = password;
        }

        public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
            Cipher cipher = result.getCryptoObject().getCipher();
            try {
                if (encryptPassword(cipher, password)) {
                    callback.onSuccess("Encrypted");
                } else {
                    callback.onFailure("Encryption failed");
                }

            } catch (Exception e) {
                callback.onFailure("Encryption failed " + e.getMessage());
            }
        }
    }

    @RequiresApi(Build.VERSION_CODES.M)
    protected class FingerPrintDecryptPasswordListener extends FingerPrintAuthenticationListener {

        public FingerPrintDecryptPasswordListener(@NonNull Callback callback) {
            super(callback);
        }

        public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {
            Cipher cipher = result.getCryptoObject().getCipher();
            try {
                String savedPass = decipher(cipher);
                if (savedPass != null) {
                    callback.onSuccess(savedPass);
                } else {
                    callback.onFailure("Failed deciphering");
                }

            } catch (Exception e) {
                callback.onFailure("Deciphering failed " + e.getMessage());
            }
        }
    }
}
```

This activity below is a very basic example of how to get a user saved password and interact with the helper.

```
public class MainActivity extends AppCompatActivity {

    private TextView passwordTextView;
    private FingerPrintAuthHelper fingerPrintAuthHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        passwordTextView = (TextView) findViewById(R.id.password);
        errorTextView = (TextView) findViewById(R.id.error);

        View savePasswordButton = findViewById(R.id.set_password_button);
        savePasswordButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
```

```
                    fingerPrintAuthHelper.savePassword(passwordTextView.getText().toString(), new
CancellationSignal(), getAuthListener(false));
                }
            }
        });

        View getPasswordButton = findViewById(R.id.get_password_button);
        getPasswordButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    fingerPrintAuthHelper.getPassword(new CancellationSignal(),
getAuthListener(true));
                }
            }
        });
    }

    // Start the finger print helper. In case this fails show error to user
    private void startFingerPrintAuthHelper() {
        fingerPrintAuthHelper = new FingerPrintAuthHelper(this);
        if (!fingerPrintAuthHelper.init()) {
            errorTextView.setText(fingerPrintAuthHelper.getLastError());
        }
    }

    @NonNull
    private FingerPrintAuthHelper.Callback getAuthListener(final boolean isGetPass) {
        return new FingerPrintAuthHelper.Callback() {
            @Override
            public void onSuccess(String result) {
                if (isGetPass) {
                    errorTextView.setText("Success!!! Pass = " + result);
                } else {
                    errorTextView.setText("Encrypted pass = " + result);
                }
            }

            @Override
            public void onFailure(String message) {
                errorTextView.setText("Failed - " + message);
            }

            @Override
            public void onHelp(int helpCode, String helpString) {
                errorTextView.setText("Help needed - " + helpString);
            }
        };
    }
}
```

Section 55.2: Adding the Fingerprint Scanner in Android application

Android supports fingerprint api from Android 6.0 (Marshmallow) SDK 23

> To use this feature in your app, first add the USE_FINGERPRINT permission in your manifest.

```
<uses-permission
```

```
        android:name="android.permission.USE_FINGERPRINT" />
```

Here the procedure to follow

> First you need to create a symmetric key in the Android Key Store using KeyGenerator which can be only be used after the user has authenticated with fingerprint and pass a KeyGenParameterSpec.

```
KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");
keyPairGenerator.initialize(
        new KeyGenParameterSpec.Builder(KEY_NAME,
                KeyProperties.PURPOSE_SIGN)
                .setDigests(KeyProperties.DIGEST_SHA256)
                .setAlgorithmParameterSpec(new ECGenParameterSpec("secp256r1"))
                .setUserAuthenticationRequired(true)
                .build());
keyPairGenerator.generateKeyPair();
```

> By setting KeyGenParameterSpec.Builder.setUserAuthenticationRequired to true, you can permit the use of the key only after the user authenticate it including when authenticated with the user's fingerprint.

```
KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PublicKey publicKey =
        keyStore.getCertificate(MainActivity.KEY_NAME).getPublicKey();

KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
keyStore.load(null);
PrivateKey key = (PrivateKey) keyStore.getKey(KEY_NAME, null);
```

> Then start listening to a fingerprint on the fingerprint sensor by calling FingerprintManager.authenticate with a Cipher initialized with the symmetric key created. Or alternatively you can fall back to server-side verified password as an authenticator.

Create and initialise the `FingerprintManger` from `fingerprintManger.class`

`getContext().getSystemService(FingerprintManager.class)`

To authenticate use `FingerprintManger` api and create subclass using

`FingerprintManager.AuthenticationCallback` and override the methods

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

To Start

To startListening the fingerPrint event call authenticate method with crypto

```
fingerprintManager
              .authenticate(cryptoObject, mCancellationSignal, 0 , this, null);
```

Cancel

to stop listenting the scanner call

```
android.os.CancellationSignal;
```

> Once the fingerprint (or password) is verified, the FingerprintManager.AuthenticationCallback#onAuthenticationSucceeded() callback is called.

```
@Override

public void onAuthenticationSucceeded(AuthenticationResult result) {

            }
```

Chapter 56: Bluetooth and Bluetooth LE API

Section 56.1: Permissions

Add this permission to the manifest file to use Bluetooth features in your application:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

If you need to initiate device discovery or manipulate Bluetooth settings, you also need to add this permission:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Targetting Android API level 23 and above, will require location access:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- OR -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

* Also see the Permissions topic for more details on how to use permissions appropriately.

Section 56.2: Check if bluetooth is enabled

```
private static final int REQUEST_ENABLE_BT = 1; // Unique request code
BluetoothAdapter mBluetoothAdapter;

// ...

if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}

// ...

@Override
protected void onActivityResult(final int requestCode, final int resultCode, final Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_ENABLE_BT) {
        if (resultCode == RESULT_OK) {
            // Bluetooth was enabled
        } else if (resultCode == RESULT_CANCELED) {
            // Bluetooth was not enabled
        }
    }
}
```

Section 56.3: Find nearby Bluetooth Low Energy devices

The BluetoothLE API was introduced in API 18. However, the way of scanning devices has changed in API 21. The searching of devices must start with defining the service UUID that is to be scanned (either officailly adopted 16-bit UUID's or proprietary ones). This example illustrates, how to make an API independent way of searching for BLE devices:

1. Create bluetooth device model:

```
public class BTDevice {
    String address;
    String name;

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

2. Define Bluetooth Scanning interface:

```
public interface ScanningAdapter {

    void startScanning(String name, String[] uuids);
    void stopScanning();
    List<BTDevice> getFoundDeviceList();
}
```

3. Create scanning factory class:

```
public class BluetoothScanningFactory implements ScanningAdapter {

    private ScanningAdapter mScanningAdapter;

    public BluetoothScanningFactory() {
        if (isNewerAPI()) {
            mScanningAdapter = new LollipopBluetoothLEScanAdapter();
        } else {
            mScanningAdapter = new JellyBeanBluetoothLEScanAdapter();
        }
    }

    private boolean isNewerAPI() {
        return Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP;
    }

    @Override
    public void startScanning(String[] uuids) {
        mScanningAdapter.startScanning(uuids);
    }

    @Override
    public void stopScanning() {
        mScanningAdapter.stopScanning();
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {
```

```
        return mScanningAdapter.getFoundDeviceList();
    }
}
```

4. Create factory implementation for each API:

API 18:

```
import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.Build;
import android.os.Parcelable;
import android.util.Log;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@TargetApi(Build.VERSION_CODES.JELLY_BEAN_MR2)
public class JellyBeanBluetoothLEScanAdapter implements ScanningAdapter{
    BluetoothAdapter bluetoothAdapter;
    ScanCallback mCallback;

    List<BTDevice> mBluetoothDeviceList;

    public JellyBeanBluetoothLEScanAdapter() {
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        mCallback = new ScanCallback();
        mBluetoothDeviceList = new ArrayList<>();
    }

    @Override
    public void startScanning(String[] uuids) {
        if (uuids == null || uuids.length == 0) {
            return;
        }
        UUID[] uuidList = createUUIDList(uuids);
        bluetoothAdapter.startLeScan(uuidList, mCallback);
    }

    private UUID[] createUUIDList(String[] uuids) {
        UUID[] uuidList = new UUID[uuids.length];
        for (int i = 0 ; i < uuids.length ; ++i) {
            String uuid = uuids[i];
            if (uuid == null) {
                continue;
            }
            uuidList[i] = UUID.fromString(uuid);
        }
        return uuidList;
    }

    @Override
    public void stopScanning() {
        bluetoothAdapter.stopLeScan(mCallback);
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {
```

```
        return mBluetoothDeviceList;
    }

    private class ScanCallback implements BluetoothAdapter.LeScanCallback {

        @Override
        public void onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord) {
            if (isAlreadyAdded(device)) {
                return;
            }
            BTDevice btDevice = new BTDevice();
            btDevice.setName(new String(device.getName()));
            btDevice.setAddress(device.getAddress());
            mBluetoothDeviceList.add(btDevice);
            Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress());
            Parcelable[] uuids = device.getUuids();
            String uuid = "";
            if (uuids != null) {
                for (Parcelable ep : uuids) {
                    uuid += ep + " ";
                }
                Log.d("Bluetooth discovery", device.getName() + " " + device.getAddress() + " " +
uuid);
            }
        }

        private boolean isAlreadyAdded(BluetoothDevice bluetoothDevice) {
            for (BTDevice device : mBluetoothDeviceList) {
                String alreadyAddedDeviceMACAddress = device.getAddress();
                String newDeviceMACAddress = bluetoothDevice.getAddress();
                if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
                    return true;
                }
            }
            return false;
        }
    }
}
```

API 21:

```
import android.annotation.TargetApi;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanFilter;
import android.bluetooth.le.ScanResult;
import android.bluetooth.le.ScanSettings;
import android.os.Build;
import android.os.ParcelUuid;

import bluetooth.model.BTDevice;

import java.util.ArrayList;
import java.util.List;

@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class LollipopBluetoothLEScanAdapter implements ScanningAdapter {
    BluetoothLeScanner bluetoothLeScanner;
    ScanCallback mCallback;

    List<BTDevice> mBluetoothDeviceList;
```

```
    public LollipopBluetoothLEScanAdapter() {
        bluetoothLeScanner = BluetoothAdapter.getDefaultAdapter().getBluetoothLeScanner();
        mCallback = new ScanCallback();
        mBluetoothDeviceList = new ArrayList<>();
    }

    @Override
    public void startScanning(String[] uuids) {
        if (uuids == null || uuids.length == 0) {
            return;
        }
        List<ScanFilter> filterList = createScanFilterList(uuids);
        ScanSettings scanSettings = createScanSettings();
        bluetoothLeScanner.startScan(filterList, scanSettings, mCallback);
    }

    private List<ScanFilter> createScanFilterList(String[] uuids) {
        List<ScanFilter> filterList = new ArrayList<>();
        for (String uuid : uuids) {
            ScanFilter filter = new ScanFilter.Builder()
                    .setServiceUuid(ParcelUuid.fromString(uuid))
                    .build();
            filterList.add(filter);
        };
        return filterList;
    }

    private ScanSettings createScanSettings() {
        ScanSettings settings = new ScanSettings.Builder()
                .setScanMode(ScanSettings.SCAN_MODE_BALANCED)
                .build();
        return settings;
    }

    @Override
    public void stopScanning() {
        bluetoothLeScanner.stopScan(mCallback);
    }

    @Override
    public List<BTDevice> getFoundDeviceList() {
        return mBluetoothDeviceList;
    }

    public class ScanCallback extends android.bluetooth.le.ScanCallback {

        @Override
        public void onScanResult(int callbackType, ScanResult result) {
            super.onScanResult(callbackType, result);
            if (result == null) {
                return;
            }
            BTDevice device = new BTDevice();
            device.setAddress(result.getDevice().getAddress());
            device.setName(new StringBuffer(result.getScanRecord().getDeviceName()).toString());
            if (device == null || device.getAddress() == null) {
                return;
            }
            if (isAlreadyAdded(device)) {
                return;
            }
            mBluetoothDeviceList.add(device);
```

```
        }

        private boolean isAlreadyAdded(BTDevice bluetoothDevice) {
            for (BTDevice device : mBluetoothDeviceList) {
                String alreadyAddedDeviceMACAddress = device.getAddress();
                String newDeviceMACAddress = bluetoothDevice.getAddress();
                if (alreadyAddedDeviceMACAddress.equals(newDeviceMACAddress)) {
                    return true;
                }
            }
            return false;
        }
    }
}
```

5. Get found device list by calling:

```
scanningFactory.startScanning({uuidlist});

wait few seconds...

List<BTDevice> bluetoothDeviceList = scanningFactory.getFoundDeviceList();
```

Section 56.4: Make device discoverable

```
private static final int REQUEST_DISCOVERABLE_BT = 2; // Unique request code
private static final int DISCOVERABLE_DURATION = 120; // Discoverable duration time in seconds
                                                      // 0 means always discoverable
                                                      // maximum value is 3600

// ...

Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, DISCOVERABLE_DURATION);
startActivityForResult(discoverableIntent, REQUEST_DISCOVERABLE_BT);

// ...

@Override
protected void onActivityResult(final int requestCode, final int resultCode, final Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_DISCOVERABLE_BT) {
        if (resultCode == RESULT_OK) {
            // Device is discoverable
        } else if (resultCode == RESULT_CANCELED) {
            // Device is not discoverable
        }
    }
}
```

Section 56.5: Connect to Bluetooth device

After you obtained BluetoothDevice, you can communicate with it. This kind of communication performed by using socket input\output streams:

Those are the basic steps for Bluetooth communication establishment:

1) Initialize socket:

```
private BluetoothSocket _socket;
//...
public InitializeSocket(BluetoothDevice device){
   try {
       _socket = device.createRfcommSocketToServiceRecord(<Your app UDID>);
   } catch (IOException e) {
       //Error
   }
 }
```

2) Connect to socket:

```
try {
    _socket.connect();
} catch (IOException connEx) {
    try {
        _socket.close();
    } catch (IOException closeException) {
        //Error
    }
}

if (_socket != null && _socket.isConnected()) {
    //Socket is connected, now we can obtain our IO streams
}
```

3) Obtaining socket Input\Output streams

```
private InputStream _inStream;
private OutputStream _outStream;
//....
try {
    _inStream = _socket.getInputStream();
    _outStream =  _socket.getOutputStream();
} catch (IOException e) {
   //Error
}
```

Input stream - Used as incoming data channel (receive data from connected device)

Output stream - Used as outgoing data channel (send data to connected device)

After finishing 3rd step, we can receive and send data between both devices using previously initialized streams:

1) Receiving data (reading from socket input stream)

```
byte[] buffer = new byte[1024];  // buffer (our data)
int bytesCount; // amount of read bytes

while (true) {
    try {
        //reading data from input stream
        bytesCount = _inStream.read(buffer);
        if(buffer != null && bytesCount > 0)
        {
            //Parse received bytes
```

```
        }
    } catch (IOException e) {
        //Error
    }
}
```

2) Sending data (Writing to output stream)

```
public void write(byte[] bytes) {
    try {
        _outStream.write(bytes);
    } catch (IOException e) {
        //Error
    }
}
```

- Of course, connection, reading and writing functionality should be done in a dedicated thread.
- Sockets and Stream objects need to be

Section 56.6: Find nearby bluetooth devices

Declare a `BluetoothAdapter` first.

```
BluetoothAdapter mBluetoothAdapter;
```

Now create a `BroadcastReceiver` for `ACTION_FOUND`

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();

    //Device found
    if (BluetoothDevice.ACTION_FOUND.equals(action))
    {
        // Get the BluetoothDevice object from the Intent
        BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
        // Add the name and address to an array adapter to show in a list
        mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
  }
};
```

Register the `BroadcastReceiver`

```
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

Then start discovering the nearby bluetooth devices by calling `startDiscovery`

```
mBluetoothAdapter.startDiscovery();
```

Don't forget to unregister the `BroadcastReceiver` inside `onDestroy`

```
unregisterReceiver(mReceiver);
```

Chapter 57: Runtime Permissions in API-23 +

Android Marshmallow introduced Runtime Permission model. Permissions are categorized into two categories i.e. Normal and Dangerous Permissions. where dangerous permissions are now granted by the user at run time.

Section 57.1: Android 6.0 multiple permissions

This example shows how to check permissions at runtime in Android 6 and later.

```
public static final int MULTIPLE_PERMISSIONS = 10; // code you want.

String[] permissions = new String[] {
    Manifest.permission.WRITE_EXTERNAL_STORAGE,
    Manifest.permission.CAMERA,
    Manifest.permission.ACCESS_COARSE_LOCATION,
    Manifest.permission.ACCESS_FINE_LOCATION
};

@Override
void onStart() {
    if (checkPermissions()){
        // permissions granted.
    } else {
        // show dialog informing them that we lack certain permissions
    }
}

private boolean checkPermissions() {
    int result;
    List<String> listPermissionsNeeded = new ArrayList<>();
    for (String p:permissions) {
        result = ContextCompat.checkSelfPermission(getActivity(),p);
        if (result != PackageManager.PERMISSION_GRANTED) {
            listPermissionsNeeded.add(p);
        }
    }
    if (!listPermissionsNeeded.isEmpty()) {
        ActivityCompat.requestPermissions(this, listPermissionsNeeded.toArray(new
String[listPermissionsNeeded.size()]), MULTIPLE_PERMISSIONS);
        return false;
    }
    return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MULTIPLE_PERMISSIONS:{
            if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                // permissions granted.
            } else {
                // no permissions granted.
            }
            return;
        }
    }
}
```

Section 57.2: Multiple Runtime Permissions From Same Permission Groups

In the manifest we have fours dangerous runtime permissions from two groups.

```
<!-- Required to read and write to shredPref file. -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>

<!-- Required to get location of device. -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

In the activity where the permissions are required. Note it is important to check for permissions in any activity that requires permissions, as the permissions can be revoked while the app is in the background and the app will then crash.

```
final private int REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS = 124;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.act_layout);

    // A simple check of whether runtime permissions need to be managed
    if (Build.VERSION.SDK_INT >= 23) {
        checkMultiplePermissions();
    }
```

We only need to ask for permission for one of these from each group and all other permissions from this group are granted unless the permission is revoked by the user.

```
private void checkMultiplePermissions() {

    if (Build.VERSION.SDK_INT >= 23) {
        List<String> permissionsNeeded = new ArrayList<String>();
        List<String> permissionsList = new ArrayList<String>();

        if (!addPermission(permissionsList, android.Manifest.permission.ACCESS_FINE_LOCATION)) {
            permissionsNeeded.add("GPS");
        }

        if (!addPermission(permissionsList, android.Manifest.permission.READ_EXTERNAL_STORAGE)) {
            permissionsNeeded.add("Read Storage");
        }

        if (permissionsList.size() > 0) {
            requestPermissions(permissionsList.toArray(new String[permissionsList.size()]),
                    REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS);
            return;
        }
    }
}



private boolean addPermission(List<String> permissionsList, String permission) {
    if (Build.VERSION.SDK_INT >= 23)
```

```
        if (checkSelfPermission(permission) != PackageManager.PERMISSION_GRANTED) {
            permissionsList.add(permission);

            // Check for Rationale Option
            if (!shouldShowRequestPermissionRationale(permission))
                return false;
        }
    return true;
}
```

This deals with the result of the user allowing or not allowing permissions. In this example, if the permissions are not allowed, the app is killed.

```
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    switch (requestCode) {
        case REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS: {

            Map<String, Integer> perms = new HashMap<String, Integer>();
            // Initial
            perms.put(android.Manifest.permission.ACCESS_FINE_LOCATION,
PackageManager.PERMISSION_GRANTED);
            perms.put(android.Manifest.permission.READ_EXTERNAL_STORAGE,
PackageManager.PERMISSION_GRANTED);

            // Fill with results
            for (int i = 0; i < permissions.length; i++)
                perms.put(permissions[i], grantResults[i]);
            if (perms.get(android.Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED
                    && perms.get(android.Manifest.permission.READ_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED) {
                // All Permissions Granted
                return;
            } else {
                // Permission Denied
                if (Build.VERSION.SDK_INT >= 23) {
                    Toast.makeText(
                            getApplicationContext(),
                            "My App cannot run without Location and Storage " +
                                    "Permissions.\nRelaunch My App or allow permissions" +
                                    " in Applications Settings",
                            Toast.LENGTH_LONG).show();
                    finish();
                }
            }
        }
        break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

More Information
https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en

Section 57.3: Using PermissionUtil

PermissionUtil is a simple and convenient way of asking for permissions in context. You can easily provide what should happen in case of all requested permissions granted (`onAllGranted()`), any request was denied

(`onAnyDenied()`) or in case that a rational is needed (`onRational()`).

Anywhere in your AppCompatActivity or Fragment that you want to ask for user's permisssion

```
mRequestObject =
PermissionUtil.with(this).request(Manifest.permission.WRITE_EXTERNAL_STORAGE).onAllGranted(
                new Func() {
                    @Override protected void call() {
                        //Happy Path
                    }
                }).onAnyDenied(
                new Func() {
                    @Override protected void call() {
                        //Sad Path
                    }
                }).ask(REQUEST_CODE_STORAGE);
```

And add this to `onRequestPermissionsResult`

```
if(mRequestObject!=null){
    mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

Add the requested permission to your AndroidManifest.xml as well

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Section 57.4: Include all permission-related code to an abstract base class and extend the activity of this base class to achieve cleaner/reusable code

```
public abstract class BaseActivity extends AppCompatActivity {
    private Map<Integer, PermissionCallback> permissionCallbackMap = new HashMap<>();

    @Override
    protected void onStart() {
        super.onStart();
        ...
    }

    @Override
    public void setContentView(int layoutResId) {
        super.setContentView(layoutResId);
        bindViews();
    }

    ...

    @Override
    public void onRequestPermissionsResult(
            int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        PermissionCallback callback = permissionCallbackMap.get(requestCode);

        if (callback == null) return;

        // Check whether the permission request was rejected.
        if (grantResults.length < 0 && permissions.length > 0) {
            callback.onPermissionDenied(permissions);
```

```
            return;
        }

        List<String> grantedPermissions = new ArrayList<>();
        List<String> blockedPermissions = new ArrayList<>();
        List<String> deniedPermissions = new ArrayList<>();
        int index = 0;

        for (String permission : permissions) {
            List<String> permissionList = grantResults[index] == PackageManager.PERMISSION_GRANTED
                    ? grantedPermissions
                    : ! ActivityCompat.shouldShowRequestPermissionRationale(this, permission)
                        ? blockedPermissions
                        : deniedPermissions;
            permissionList.add(permission);
            index ++;
        }

        if (grantedPermissions.size() > 0) {
            callback.onPermissionGranted(
                    grantedPermissions.toArray(new String[grantedPermissions.size()]));
        }

        if (deniedPermissions.size() > 0) {
            callback.onPermissionDenied(
                    deniedPermissions.toArray(new String[deniedPermissions.size()]));
        }

        if (blockedPermissions.size() > 0) {
            callback.onPermissionBlocked(
                    blockedPermissions.toArray(new String[blockedPermissions.size()]));
        }

        permissionCallbackMap.remove(requestCode);
    }

    /**
     * Check whether a permission is granted or not.
     *
     * @param permission
     * @return
     */
    public boolean hasPermission(String permission) {
        return ContextCompat.checkSelfPermission(this, permission) ==
PackageManager.PERMISSION_GRANTED;
    }

    /**
     * Request permissions and get the result on callback.
     *
     * @param permissions
     * @param callback
     */
    public void requestPermission(String [] permissions, @NonNull PermissionCallback callback) {
        int requestCode = permissionCallbackMap.size() + 1;
        permissionCallbackMap.put(requestCode, callback);
        ActivityCompat.requestPermissions(this, permissions, requestCode);
    }

    /**
     * Request permission and get the result on callback.
     *
```

```
     * @param permission
     * @param callback
     */
    public void requestPermission(String permission, @NonNull PermissionCallback callback) {
        int requestCode = permissionCallbackMap.size() + 1;
        permissionCallbackMap.put(requestCode, callback);
        ActivityCompat.requestPermissions(this, new String[] { permission }, requestCode);
    }
}
```

Example usage in the activity

The activity should extend the abstract base class defined above as follows:

```
private void requestLocationAfterPermissionCheck() {
    if (hasPermission(Manifest.permission.ACCESS_FINE_LOCATION)) {
        requestLocation();
        return;
    }

    // Call the base class method.
    requestPermission(Manifest.permission.ACCESS_FINE_LOCATION, new PermissionCallback() {
        @Override
        public void onPermissionGranted(String[] grantedPermissions) {
            requestLocation();
        }

        @Override
        public void onPermissionDenied(String[] deniedPermissions) {
            // Do something.
        }

        @Override
        public void onPermissionBlocked(String[] blockedPermissions) {
            // Do something.
        }
    });
}
```

Section 57.5: Enforcing Permissions in Broadcasts, URI

You can do a permissions check when sending an Intent to a registered broadcast receiver. The permissions you send are cross-checked with the ones registered under the tag. They restrict who can send broadcasts to the associated receiver.

To send a broadcast request with permissions, specify the permission as a string in the `Context.sendBroadcast(Intent intent, String permission)` call, but keep in mind that the Receiver's app **MUST** have that permission in order to receive your broadcast. The receiver should be installed first before the sender.

The method signature is:

```
void sendBroadcast (Intent intent, String receiverPermission)
//for example to send a broadcast to Bcastreceiver receiver
Intent broadcast = new Intent(this, Bcastreceiver.class);
sendBroadcast(broadcast, "org.quadcore.mypermission");
```

and you can specify in your manifest that the broadcast sender is required to include the requested permission sent through the sendBroadcast:

```
<!--  Your special permission -->
<permission android:name="org.quadcore.mypermission"
  android:label="my_permission"
  android:protectionLevel="dangerous"></permission>
```

Also declare the permission in the manifest of the application that is supposed to receive this broadcast:

```
<!--  I use the permission ! -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- along with the receiver -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

Note: Both a receiver and a broadcaster can require a permission, and when this happens, both permission checks must pass for the Intent to be delivered to the associated target. The App that defines the permission should be installed first.

Find the full documentation here on Permissions.

Chapter 58: Android Places API

Section 58.1: Getting Current Places by Using Places API

You can get the current location and local places of user by using the Google Places API.

Ar first, you should call the `PlaceDetectionApi.getCurrentPlace()` method in order to retrieve local business or other places. This method returns a `PlaceLikelihoodBuffer` object which contains a list of `PlaceLikelihood` objects. Then, you can get a `Place` object by calling the `PlaceLikelihood.getPlace()` method.

Important: You must request and obtain the `ACCESS_FINE_LOCATION` permission in order to allow your app to access precise location information.

```
private static final int PERMISSION_REQUEST_TO_ACCESS_LOCATION = 1;

private TextView txtLocation;
private GoogleApiClient googleApiClient;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_location);

    txtLocation = (TextView) this.findViewById(R.id.txtLocation);
    googleApiClient = new GoogleApiClient.Builder(this)
            .addApi(Places.GEO_DATA_API)
            .addApi(Places.PLACE_DETECTION_API)
            .enableAutoManage(this, this)
            .build();

    getCurrentLocation();
}

private void getCurrentLocation() {
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        Log.e(LOG_TAG, "Permission is not granted");

        ActivityCompat.requestPermissions(this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},PERMISSION_REQUEST_TO_ACCESS_LOCATION);
        return;
    }

    Log.i(LOG_TAG, "Permission is granted");

    PendingResult<PlaceLikelihoodBuffer> result =
Places.PlaceDetectionApi.getCurrentPlace(googleApiClient, null);
    result.setResultCallback(new ResultCallback<PlaceLikelihoodBuffer>() {
        @Override
        public void onResult(PlaceLikelihoodBuffer likelyPlaces) {
            Log.i(LOG_TAG, String.format("Result received : %d " , likelyPlaces.getCount() ));
            StringBuilder stringBuilder = new StringBuilder();

            for (PlaceLikelihood placeLikelihood : likelyPlaces) {
                stringBuilder.append(String.format("Place : '%s' %n",
                        placeLikelihood.getPlace().getName()));
            }
            likelyPlaces.release();
            txtLocation.setText(stringBuilder.toString());
```

```
        }
    });
}

@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case PERMISSION_REQUEST_TO_ACCESS_LOCATION: {
            // If the request is cancelled, the result arrays are empty.
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                getCurrentLocation();
            } else {
                // Permission denied, boo!
                // Disable the functionality that depends on this permission.
            }
            return;
        }

        // Add further 'case' lines to check for other permissions this app might request.
    }
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    Log.e(LOG_TAG, "GoogleApiClient connection failed: " + connectionResult.getErrorMessage());
}
```

Section 58.2: Place Autocomplete Integration

The autocomplete feature in the Google Places API for Android provides place predictions to user. While user types in the search box, autocomplete shows places according to user's queries.

AutoCompleteActivity.java

```
private TextView txtSelectedPlaceName;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_autocomplete);

    txtSelectedPlaceName = (TextView) this.findViewById(R.id.txtSelectedPlaceName);

    PlaceAutocompleteFragment autocompleteFragment = (PlaceAutocompleteFragment)
            getFragmentManager().findFragmentById(R.id.fragment_autocomplete);

    autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
        @Override
        public void onPlaceSelected(Place place) {
            Log.i(LOG_TAG, "Place: " + place.getName());
            txtSelectedPlaceName.setText(String.format("Selected places : %s  - %s" ,
place.getName(), place.getAddress()));
        }

        @Override
        public void onError(Status status) {
            Log.i(LOG_TAG, "An error occurred: " + status);
            Toast.makeText(AutoCompleteActivity.this, "Place cannot be selected!!",
Toast.LENGTH_SHORT).show();
        }
```

```
    });

}
```

}

activity_autocomplete.xml

```
<fragment
   android:id="@+id/fragment_autocomplete"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:name="com.google.android.gms.location.places.ui.PlaceAutocompleteFragment"
   />



<TextView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:id="@+id/txtSelectedPlaceName"
   android:layout_margin="20dp"
   android:padding="15dp"
   android:hint="@string/txt_select_place_hint"
   android:textSize="@dimen/place_autocomplete_prediction_primary_text"/>
```

Section 58.3: Place Picker Usage Example

Place Picker is a really simple UI widget provided by Places API. It provides a built-in map, current location, nearby places, search abilities and autocomplete.

This is a sample usage of Place Picker UI widget.

```
private static int PLACE_PICKER_REQUEST = 1;

private TextView txtPlaceName;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_place_picker_sample);

    txtPlaceName = (TextView) this.findViewById(R.id.txtPlaceName);
    Button btnSelectPlace = (Button) this.findViewById(R.id.btnSelectPlace);
    btnSelectPlace.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            openPlacePickerView();
        }
    });

}

private void openPlacePickerView(){
    PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();
    try {
        startActivityForResult(builder.build(this), PLACE_PICKER_REQUEST);
    } catch (GooglePlayServicesRepairableException e) {
        e.printStackTrace();
```

```
    } catch (GooglePlayServicesNotAvailableException e) {
        e.printStackTrace();
    }
}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PLACE_PICKER_REQUEST) {
        if (resultCode == RESULT_OK) {
            Place place = PlacePicker.getPlace(this, data);
            Log.i(LOG_TAG, String.format("Place Name : %s", place.getName()));
            Log.i(LOG_TAG, String.format("Place Address : %s", place.getAddress()));
            Log.i(LOG_TAG, String.format("Place Id : %s", place.getId()));

            txtPlaceName.setText(String.format("Place : %s - %s" , place.getName() ,
place.getAddress()));
        }
    }
}
```

Section 58.4: Setting place type filters for PlaceAutocomplete

In some scenarios, we might want to narrow down the results being shown by **PlaceAutocomplete** to a specific country or maybe to show only Regions. This can be achieved by setting an **AutocompleteFilter** on the intent. For example, if I want to look only for places of type REGION and only belonging to India, I would do the following:

MainActivity.java

```
public class MainActivity extends AppComatActivity {

    private static final int PLACE_AUTOCOMPLETE_REQUEST_CODE = 1;
    private TextView selectedPlace;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        selectedPlace = (TextView) findViewById(R.id.selected_place);
        try {
            AutocompleteFilter typeFilter = new AutocompleteFilter.Builder()
                    .setTypeFilter(AutocompleteFilter.TYPE_FILTER_REGIONS)
                    .setCountry("IN")
                    .build();

            Intent intent =
                    new PlaceAutocomplete.IntentBuilder(PlaceAutocomplete.MODE_FULLSCREEN)
                            .setFilter(typeFilter)
                            .build(this);
            startActivityForResult(intent, PLACE_AUTOCOMPLETE_REQUEST_CODE);

        } catch (GooglePlayServicesRepairableException
                | GooglePlayServicesNotAvailableException e) {
            e.printStackTrace();
        }
}

protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == PLACE_AUTOCOMPLETE_REQUEST_CODE && resultCode == Activity.RESULT_OK) {
```

```
        final Place place = PlacePicker.getPlace(this, data);
        selectedPlace.setText(place.getName().toString().toUpperCase());
    } else {
        Toast.makeText(MainActivity.this, "Could not get location.", Toast.LENGTH_SHORT).show();
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
         android:orientation="vertical"
         android:layout_width="match_parent"
         android:layout_height="match_parent">

<TextView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:id="@+id/selected_place"/>

</LinearLayout>
```

The **PlaceAutocomplete** will launch automatically and you can then select a place from the results which will only be of the type **REGION** and will only belong to the specified country. The intent can also be launched at the click of a button.

Section 58.5: Adding more than one google auto complete activity

```
 public static final int PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE=1;
   public static final int PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE=2;

fromPlaceEdit.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    try {
                      //Do your stuff from place
                        startActivityForResult(intent, PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE);

                    } catch (GooglePlayServicesRepairableException e) {
                        // TODO: Handle the error.
                    } catch (GooglePlayServicesNotAvailableException e) {
                        // TODO: Handle the error.
                    }
                }
            });
toPlaceEdit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                try {
                   //Do your stuff to place
                    startActivityForResult(intent, PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE);

                } catch (GooglePlayServicesRepairableException e) {
                    // TODO: Handle the error.
                } catch (GooglePlayServicesNotAvailableException e) {
                    // TODO: Handle the error.
```

```
                }
            }
        });
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
       if (requestCode == PLACE_AUTOCOMPLETE_FROM_PLACE_REQUEST_CODE) {
           if (resultCode == RESULT_OK) {
              //Do your ok >from place< stuff here
           } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
              //Handle your error >from place<
           } else if (resultCode == RESULT_CANCELED) {
               // The user canceled the operation.
           }
       } else if (requestCode == PLACE_AUTOCOMPLETE_TO_PLACE_REQUEST_CODE) {
           if (resultCode == RESULT_OK) {
              //Do your ok >to place< stuff here
           } else if (resultCode == PlaceAutocomplete.RESULT_ERROR) {
              //Handle your error >to place<
           } else if (resultCode == RESULT_CANCELED) {
               // The user canceled the operation.
           }
        }
    }
```

Chapter 59: Android NDK

Section 59.1: How to log in ndk

First make sure you link against the logging library in your `Android.mk` file:

```
LOCAL_LDLIBS := -llog
```

Then use one of the following `__android_log_print()` calls:

```
#include <android/log.h>
#define TAG "MY LOG"

__android_log_print(ANDROID_LOG_VERBOSE,    TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_WARN,       TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_DEBUG,      TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_INFO,       TAG, "The value of 1 + 1 is %d", 1 + 1)
__android_log_print(ANDROID_LOG_ERROR,      TAG, "The value of 1 + 1 is %d", 1 + 1)
```

Or use those in a more convenient way by defining corresponding macros:

```
#define  LOGV(...)  __android_log_print(ANDROID_LOG_VERBOSE,    TAG, __VA_ARGS__)
#define  LOGW(...)  __android_log_print(ANDROID_LOG_WARN,       TAG, __VA_ARGS__)
#define  LOGD(...)  __android_log_print(ANDROID_LOG_DEBUG,      TAG, __VA_ARGS__)
#define  LOGI(...)  __android_log_print(ANDROID_LOG_INFO,       TAG, __VA_ARGS__)
#define  LOGE(...)  __android_log_print(ANDROID_LOG_ERROR,      TAG, __VA_ARGS__)
```

Example:

```
int x = 42;
LOGD("The value of x is %d", x);
```

Section 59.2: Building native executables for Android

project/jni/main.c

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
  printf("Hello world!\n");
  return 0;
}
```

project/jni/Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE    := hello_world
LOCAL_SRC_FILES := main.c
include $(BUILD_EXECUTABLE)
```

project/jni/Application.mk

```
APP_ABI := all
APP_PLATFORM := android-21
```

If you want to support devices running Android versions lower than 5.0 (API 21), you need to compile your binary with `APP_PLATFORM` set to an older API, e.g. `android-8`. This is a consequence of Android 5.0 enforcing *Position Independent Binaries* (PIE), whereas older devices do not necessarily support PIEs. Therefore, you need to use either the PIE or the non-PIE, depending on the device version. If you want to use the binary from within your Android application, you need to check the API level and extract the correct binary.

`APP_ABI` can be changed to specific platforms such as `armeabi` to build the binary for those architectures only.

In the worst case, you will have both a PIE and a non-PIE binary for each architecture (about 14 different binaries using ndk-r10e).

To build the executable:

```
cd project
ndk-build
```

You will find the binaries at `project/libs/<architecture>/hello_world`. You can use them via `ADB` (`push` and `chmod` it with executable permission) or from your application (extract and `chmod` it with executable permission).

To determine the architecture of the CPU, retrieve the build property `ro.product.cpu.abi` for the primary architecture or `ro.product.cpu.abilist` (on newer devices) for a complete list of supported architectures. You can do this using the `android.os.Build` class from within your application or using `getprop <name>` via ADB.

Section 59.3: How to clean the build

If you need to clean the build:

```
ndk-build clean
```

Section 59.4: How to use a makefile other than Android.mk

ndk-build NDK_PROJECT_PATH=PROJECT_PATH APP_BUILD_SCRIPT=MyAndroid.mk

Chapter 60: DayNight Theme (AppCompat v23.2 / API 14+)

Section 60.1: Adding the DayNight theme to an app

The DayNight theme gives an app the cool capability of switching color schemes based on the time of day and the device's last known location.

Add the following to your `styles.xml`:

```
<style name="AppTheme" parent="Theme.AppCompat.DayNight">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
</style>
```

The themes you can extend from to add day night theme switching capability are the following:

- `"Theme.AppCompat.DayNight"`
- `"Theme.AppCompat.DayNight.NoActionBar"`
- `"Theme.AppCompat.DayNight.DarkActionBar"`

Apart from `colorPrimary`, `colorPrimaryDark` and `colorAccent`, you can also add any other colors that you would like to be switched, e.g. `textColorPrimary` or `textColorSecondary`. You can add your app's custom colors to this `style` as well.

For theme switching to work, you need to define a default `colors.xml` in the `res/values` directory and another `colors.xml` in the `res/values-night` directory and define day/night colors appropriately.

To switch the theme, call the `AppCompatDelegate.setDefaultNightMode(int)` method from your Java code. (This will change the color scheme for the whole app, not just any one activity or fragment.) For example:

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

You can pass any of the following three according to your choice:

- `AppCompatDelegate.MODE_NIGHT_NO`: this sets the default theme for your app and takes the colors defined in the `res/values` directory. It is recommended to use light colors for this theme.
- `AppCompatDelegate.MODE_NIGHT_YES`: this sets a night theme for your app and takes the colors defined in the `res/values-night` directory. It is recommended to use dark colors for this theme.
- `AppCompatDelegate.MODE_NIGHT_AUTO`: this auto switches the colors of the app based on the time of the day and the colors you have defined in `values` and `values-night` directories.

It is also possible to get the current night mode status using the `getDefaultNightMode()` method. For example:

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

Please note, however, that the theme switch will not persist if you kill the app and reopen it. If you do that, the theme will switch back to `AppCompatDelegate.MODE_NIGHT_AUTO`, which is the default value. If you want the theme switch to persist, make sure you store the value in shared preferences and load the stored value each time the app is opened after it has been destroyed.

Chapter 61: Glide

**** WARNING This documentation is unmaintained and frequently inaccurate ****

Glide's official documentation is a much better source:

For Glide v4, see http://bumptech.github.io/glide/. For Glide v3, see https://github.com/bumptech/glide/wiki.

Section 61.1: Loading an image

ImageView

To load an image from a specified URL, Uri, resource id, or any other model into an `ImageView`:

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
    .into(imageView);
```

For Uris, replace `yourUrl` with your Uri (`content://media/external/images/1`). For Drawables replace `yourUrl` with your resource id (`R.drawable.image`).

RecyclerView and ListView

In ListView or RecyclerView, you can use exactly the same lines:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    Glide.with(context)
        .load(currentUrl)
        .into(myViewHolder.imageView);
}
```

If you don't want to start a load in `onBindViewHolder`, make sure you `clear()` any `ImageView` Glide may be managing before modifying the `ImageView`:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    if (TextUtils.isEmpty(currentUrl)) {
        Glide.clear(viewHolder.imageView);
        // Now that the view has been cleared, you can safely set your own resource
        viewHolder.imageView.setImageResource(R.drawable.missing_image);
    } else {
        Glide.with(context)
            .load(currentUrl)
            .into(myViewHolder.imageView);
    }
}
```

Section 61.2: Add Glide to your project

From the official documentation:

With Gradle:

```
repositories {
  mavenCentral() // jcenter() works as well because it pulls from Maven Central
}

dependencies {
  compile 'com.github.bumptech.glide:glide:4.0.0'
  compile 'com.android.support:support-v4:25.3.1'
  annotationProcessor 'com.github.bumptech.glide:compiler:4.0.0'
}
```

With Maven:

```
<dependency>
  <groupId>com.github.bumptech.glide</groupId>
  <artifactId>glide</artifactId>
  <version>4.0.0</version>
</dependency>
<dependency>
  <groupId>com.google.android</groupId>
  <artifactId>support-v4</artifactId>
  <version>r7</version>
</dependency>
<dependency>
  <groupId>com.github.bumptech.glide</groupId>
  <artifactId>compiler</artifactId>
  <version>4.0.0</version>
  <optional>true</optional>
</dependency>
```

Depending on your ProGuard (DexGuard) config and usage, you may also need to include the following lines in your proguard.cfg (See Glide's wiki for more info):

```
-keep public class * implements com.bumptech.glide.module.GlideModule
-keep public class * extends com.bumptech.glide.AppGlideModule
-keep public enum com.bumptech.glide.load.resource.bitmap.ImageHeaderParser$**     {
  **[] $VALUES;
  public *;
}

# for DexGuard only
-keepresourcexmlelements manifest/application/meta-data@value=GlideModule
```

Section 61.3: Glide circle transformation (Load image in a circular ImageView)

Create a circle image with glide.

```
public class CircleTransform extends BitmapTransformation {

    public CircleTransform(Context context) {
        super(context);
    }
```

```
    @Override protected Bitmap transform(BitmapPool pool, Bitmap toTransform, int outWidth, int
outHeight) {
        return circleCrop(pool, toTransform);
    }

    private static Bitmap circleCrop(BitmapPool pool, Bitmap source) {
        if (source == null) return null;

        int size = Math.min(source.getWidth(), source.getHeight());
        int x = (source.getWidth() - size) / 2;
        int y = (source.getHeight() - size) / 2;

        Bitmap squared = Bitmap.createBitmap(source, x, y, size, size);

        Bitmap result = pool.get(size, size, Bitmap.Config.ARGB_8888);
        if (result == null) {
            result = Bitmap.createBitmap(size, size, Bitmap.Config.ARGB_8888);
        }

        Canvas canvas = new Canvas(result);
        Paint paint = new Paint();
        paint.setShader(new BitmapShader(squared, BitmapShader.TileMode.CLAMP,
BitmapShader.TileMode.CLAMP));
        paint.setAntiAlias(true);
        float r = size / 2f;
        canvas.drawCircle(r, r, r, paint);
        return result;
    }

    @Override public String getId() {
        return getClass().getName();
    }
}
```

Usage:

```
Glide.with(context)
    .load(yourimageurl)
    .transform(new CircleTransform(context))
    .into(userImageView);
```

Section 61.4: Default transformations

Glide includes two default transformations, fit center and center crop.

Fit center:

```
Glide.with(context)
    .load(yourUrl)
    .fitCenter()
    .into(yourView);
```

Fit center performs the same transformation as Android's ScaleType.FIT_CENTER.

Center crop:

```
Glide.with(context)
    .load(yourUrl)
    .centerCrop()
```

```
    .into(yourView);
```

Center crop performs the same transformation as Android's ScaleType.CENTER_CROP.

For more information, see Glide's wiki.

Section 61.5: Glide rounded corners image with custom Glide target

First make utility class or use this method in class needed

```
public class UIUtils {
public static BitmapImageViewTarget getRoundedImageTarget(@NonNull final Context context, @NonNull
final ImageView imageView,
                                                          final float radius) {
    return new BitmapImageViewTarget(imageView) {
        @Override
        protected void setResource(final Bitmap resource) {
            RoundedBitmapDrawable circularBitmapDrawable =
                    RoundedBitmapDrawableFactory.create(context.getResources(), resource);
            circularBitmapDrawable.setCornerRadius(radius);
            imageView.setImageDrawable(circularBitmapDrawable);
        }
    };
}
```

Loading image:

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Because you use asBitmap() the animations will be removed though. You can use your own animation in this place using the animate() method.

Example with similar fade in to default Glide animation.

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .animate(R.anim.abc_fade_in)
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Please note this animation is support library private resource - it is unrecommended to use as it can change or even be removed.

Note you also need to have support library to use RoundedBitmapDrawableFactory

Section 61.6: Placeholder and Error handling

If you want to add a Drawable be shown during the load, you can add a placeholder:

```
Glide.with(context)
    .load(yourUrl)
    .placeholder(R.drawable.placeholder)
```

```
    .into(imageView);
```

If you want a Drawable to be shown if the load fails for any reason:

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

If you want a Drawable to be shown if you provide a null model (URL, Uri, file path etc):

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

Section 61.7: Preloading images

To preload remote images and ensure that the image is only downloaded once:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

Then:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // ALL works here too
    .into(imageView);
```

To preload local images and make sure a transformed copy is in the disk cache (and maybe the memory cache):

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Or whatever transformation you want
    .preload(200, 200); // Or whatever width and height you want
```

Then:

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // You must use the same transformation as above
    .override(200, 200) // You must use the same width and height as above
    .into(imageView);
```

Section 61.8: Handling Glide image load failed

```
Glide
        .with(context)
        .load(currentUrl)
        .into(new BitmapImageViewTarget(profilePicture) {
    @Override
    protected void setResource(Bitmap resource) {
        RoundedBitmapDrawable circularBitmapDrawable =
                RoundedBitmapDrawableFactory.create(context.getResources(), resource);
```

```
        circularBitmapDrawable.setCornerRadius(radius);
        imageView.setImageDrawable(circularBitmapDrawable);
    }

    @Override
    public void onLoadFailed(@NonNull Exception e, Drawable errorDrawable) {
        super.onLoadFailed(e, SET_YOUR_DEFAULT_IMAGE);
        Log.e(TAG, e.getMessage(), e);
    }
});
```

Here at `SET_YOUR_DEFAULT_IMAGE` place you can set any default `Drawable`. This image will be shown if Image loading is failed.

Section 61.9: Load image in a circular ImageView without custom transformations

Create a custom BitmapImageViewTarget to load the image into:

```
public class CircularBitmapImageViewTarget extends BitmapImageViewTarget
{
    private Context context;
    private ImageView imageView;

    public CircularBitmapImageViewTarget(Context context, ImageView imageView)
    {
        super(imageView);
        this.context = context;
        this.imageView = imageView;
    }

    @Override
    protected void setResource(Bitmap resource)
    {
        RoundedBitmapDrawable bitmapDrawable =
RoundedBitmapDrawableFactory.create(context.getResources(), resource);
        bitmapDrawable.setCircular(true);
        imageView.setImageDrawable(bitmapDrawable);
    }
}
```

Usage:

```
Glide
    .with(context)
    .load(yourimageidentifier)
    .asBitmap()
    .into(new CircularBitmapImageViewTarget(context, imageView));
```

Chapter 62: Dialog

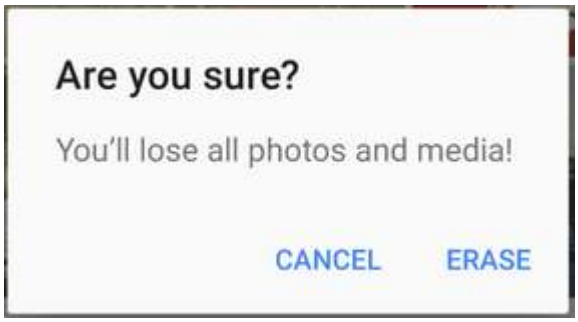
Line	Description
show();	Shows the dialog
setContentView(R.layout.yourlayout);	sets the ContentView of the dialog to your custom layout.
dismiss()	Closes the dialog

Section 62.1: Adding Material Design AlertDialog to your app using Appcompat

`AlertDialog` is a subclass of `Dialog` that can display one, two or three buttons. If you only want to display a String in this dialog box, use the `setMessage()` method.

The `AlertDialog` from `android.app` package displays differently on different Android OS Versions.

The Android V7 Appcompat library provides an `AlertDialog` implementation which will display with Material Design on all supported Android OS versions, as shown below:



First you need to add the V7 Appcompat library to your project. you can do this in the app level build.gradle file:

```
dependencies {
    compile 'com.android.support:appcompat-v7:24.2.1'
    //........
}
```

Be sure to import the correct class:

```
import android.support.v7.app.AlertDialog;
```

Then Create AlertDialog like this:

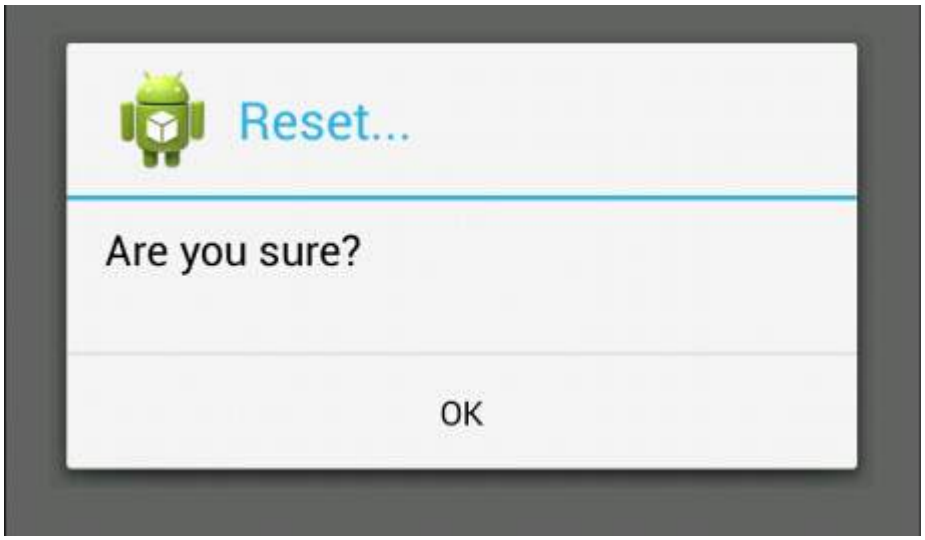
```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Are you sure?");
builder.setMessage("You'll lose all photos and media!");
builder.setPositiveButton("ERASE", null);
builder.setNegativeButton("CANCEL", null);
builder.show();
```

Section 62.2: A Basic Alert Dialog

```
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    //Set Title
    builder.setTitle("Reset...")
            //Set Message
            .setMessage("Are you sure?")
```

```
            //Set the icon of the dialog
            .setIcon(drawable)
            //Set the positive button, in this case, OK, which will dismiss the dialog and do
everything in the onClick method
            .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    // Reset
                }
            });
    AlertDialog dialog = builder.create();
    //Now, any time you can call on:
    dialog.show();
    //So you can show the dialog.
```

Now this code will achieve this:



(*Image source: WikiHow*)

Section 62.3: ListView in AlertDialog

We can always use `ListView` or `RecyclerView` for selection from list of items, but if we have small amount of choices and among those choices we want user to select one, we can use `AlertDialog.Builder setAdapter`.

```
    private void showDialog()
    {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Choose any item");

        final List<String> lables = new ArrayList<>();
        lables.add("Item 1");
        lables.add("Item 2");
        lables.add("Item 3");
        lables.add("Item 4");

        ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, lables);
        builder.setAdapter(dataAdapter, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this,"You have selected " +
lables.get(which),Toast.LENGTH_LONG).show();
            }
        });
        AlertDialog dialog = builder.create();
```

```
        dialog.show();
    }
```

Perhaps, if we don't need any particular `ListView`, we can use a basic way:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Select an item")
       .setItems(R.array.your_array, new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int which) {
               // The 'which' argument contains the index position of the selected item
               Log.v(TAG, "Selected item on position " + which);
           }
        });
builder.create().show();
```

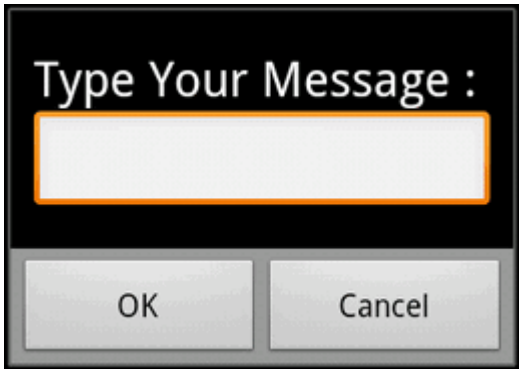
Section 62.4: Custom Alert Dialog with EditText

```
void alertDialogDemo() {
        // get alert_dialog.xml view
        LayoutInflater li = LayoutInflater.from(getApplicationContext());
        View promptsView = li.inflate(R.layout.alert_dialog, null);

        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                getApplicationContext());

        // set alert_dialog.xml to alertdialog builder
        alertDialogBuilder.setView(promptsView);

        final EditText userInput = (EditText) promptsView.findViewById(R.id.etUserInput);

        // set dialog message
        alertDialogBuilder
                .setCancelable(false)
                .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        // get user input and set it to result
                        // edit text
                        Toast.makeText(getApplicationContext(), "Entered:
"+userInput.getText().toString(), Toast.LENGTH_LONG).show();
                    }
                })
                .setNegativeButton("Cancel",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });

        // create alert dialog
        AlertDialog alertDialog = alertDialogBuilder.create();

        // show it
        alertDialog.show();
    }
```

Xml file: res/layout/alert_dialog.xml

```
<TextView
   android:id="@+id/textView1"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Type Your Message : "
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:id="@+id/etUserInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

     <requestFocus />

</EditText>
```



Section 62.5: DatePickerDialog

`DatePickerDialog` is the simplest way to use `DatePicker`, because you can show dialog anywhere in your app. You don't have to implement your own layout with `DatePicker` widget.

How to show dialog:

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

You can get `DataPicker` widget from dialog above, to get access to more functions, and for example set minimum date in milliseconds:

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

Section 62.6: DatePicker

`DatePicker` allows user to pick date. When we create new instance of `DatePicker`, we can set initial date. If we don't set initial date, current date will be set by default.

We can show `DatePicker` to user by using `DatePickerDialog` or by creating our own layout with `DatePicker` widget.

Also we can limit range of date, which user can pick.

> By setting minimum date in milliseconds

```
//In this case user can pick date only from future
```

```
datePicker.setMinDate(System.currentTimeMillis());
```

> By setting maximum date in milliseconds

```
//In this case user can pick date only, before following week.
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

To receive information, about which date was picked by user, we have to use `Listener`.

If we are using `DatePickerDialog`, we can set `OnDateSetListener` in constructor when we are creating new instance of `DatePickerDialog`:

Sample use of `DatePickerDialog`

```
public class SampleActivity extends AppCompatActivity implements DatePickerDialog.OnDateSetListener
{

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ...
    }

    private void showDatePicker() {
        //We need calendar to set current date as initial date in DatePickerDialog.
        Calendar calendar = new GregorianCalendar(Locale.getDefault());
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new DatePickerDialog(this, this, year, month, day);
        datePickerDialog.show();
    }

    @Override
    public void onDateSet(DatePicker datePicker, int year, int month, int day) {

    }
}
```

Otherwise, if we are creating our own layout with `DatePicker` widget, we also have to create our own listener as it was shown in other example

Section 62.7: Alert Dialog

```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                MainActivity.this);

        alertDialogBuilder.setTitle("Title Dialog");
        alertDialogBuilder
                .setMessage("Message Dialog")
                .setCancelable(true)
                .setPositiveButton("Yes",
                        new DialogInterface.OnClickListener() {

                            public void onClick(DialogInterface dialog, int arg1) {
                                // Handle Positive Button
```

```
                        }
                    })
                .setNegativeButton("No",
                        new DialogInterface.OnClickListener() {

                            public void onClick(DialogInterface dialog, int arg1) {
                                // Handle Negative Button
                                dialog.cancel();
                            }
                        });

        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
```

Section 62.8: Alert Dialog with Multi-line Title

The setCustomTitle() method of AlertDialog.Builder lets you specify an arbitrary view to be used for the dialog title. One common use for this method is to build an alert dialog that has a long title.

```
AlertDialog.Builder builder = new AlertDialog.Builder(context, Theme_Material_Light_Dialog);
builder.setCustomTitle(inflate(context, R.layout.my_dialog_title, null))
       .setView(inflate(context, R.layout.my_dialog, null))
       .setPositiveButton("OK", null);

Dialog dialog = builder.create();
dialog.show();
```

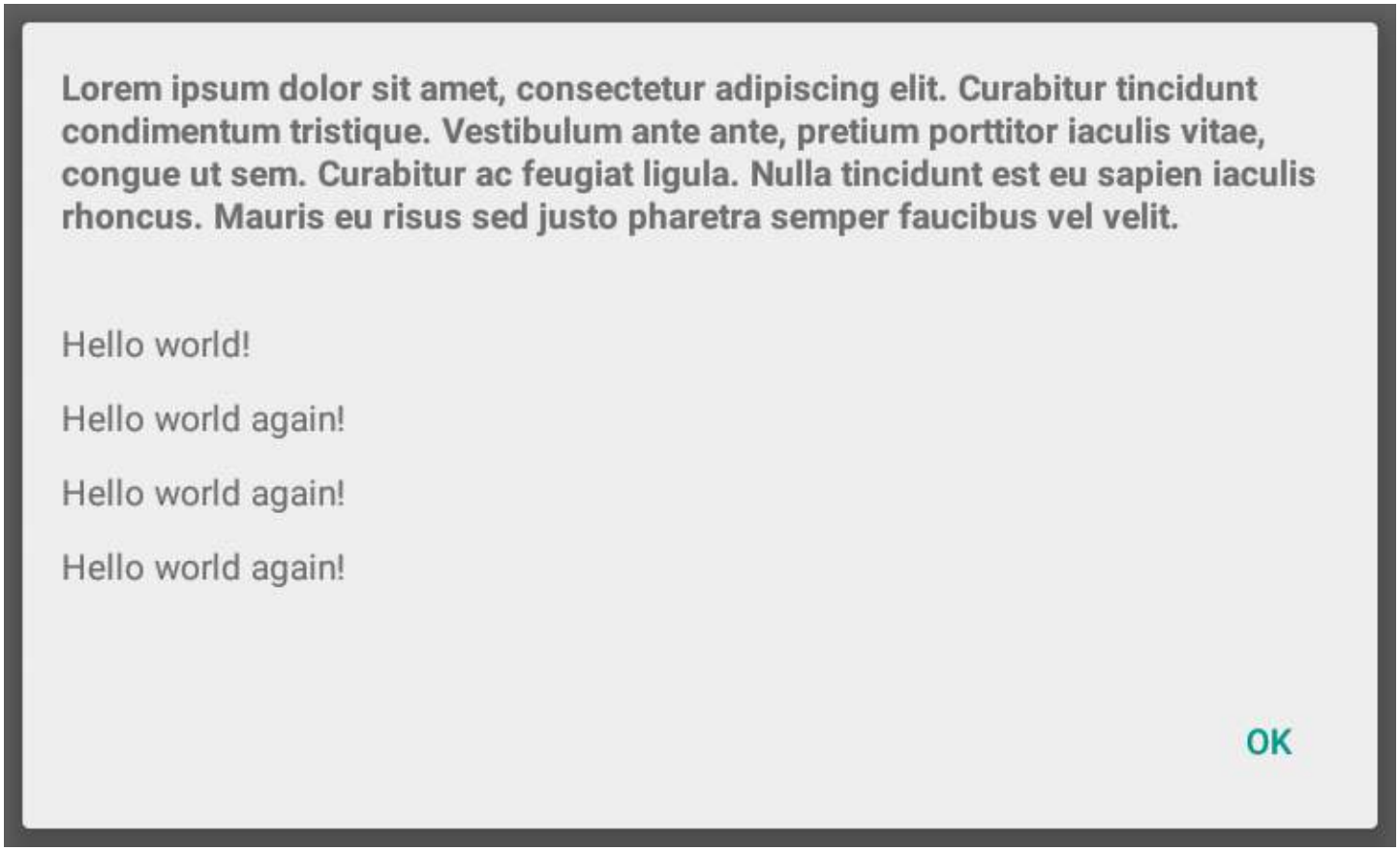
my_dialog_title.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:padding="16dp">

    <TextView
       style="@android:style/TextAppearance.Small"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:text="Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur
tincidunt condimentum tristique. Vestibulum ante ante, pretium porttitor
iaculis vitae, congue ut sem. Curabitur ac feugiat ligula. Nulla
tincidunt est eu sapien iaculis rhoncus. Mauris eu risus sed justo
pharetra semper faucibus vel velit."
       android:textStyle="bold"/>

</LinearLayout>
```

my_dialog.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:orientation="vertical"
        android:padding="16dp"
        android:scrollbars="vertical">

        <TextView
            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world!" />

        <TextView
            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world again!" />

        <TextView
            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world again!" />

        <TextView

            style="@android:style/TextAppearance.Small"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:text="Hello world again!" />

    </LinearLayout>
</ScrollView>
```



Section 62.9: Date Picker within DialogFragment

xml of the Dialog:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <DatePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/datePicker"
        android:layout_gravity="center_horizontal"
        android:calendarViewShown="false"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ACCEPT"
        android:id="@+id/buttonAccept" />

</LinearLayout>
```

Dialog Class:

```
public class ChooseDate extends DialogFragment implements View.OnClickListener {

    private DatePicker datePicker;
    private Button acceptButton;

    private boolean isDateSetted = false;
    private int year;
```

```
    private int month;
    private int day;

    private DateListener listener;

    public interface DateListener {
        onDateSelected(int year, int month, int day);
    }

    public ChooseDate(){}

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.dialog_year_picker, container);

        getDialog().setTitle(getResources().getString("TITLE"));

        datePicker = (DatePicker) rootView.findViewById(R.id.datePicker);
        acceptButton = (Button) rootView.findViewById(R.id.buttonAccept);
        acceptButton.setOnClickListener(this);

        if (isDateSetted) {
            datePicker.updateDate(year, month, day);
        }

        return rootView;
    }

    @Override
    public void onClick(View v) {
        switch(v.getId()){
            case R.id.buttonAccept:
                int year = datePicker.getYear();
                int month = datePicker.getMonth() + 1; // months start in 0
                int day = datePicker.getDayOfMonth();

                listener.onDateSelected(year, month, day);
                break;
        }
        this.dismiss();
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        listener = (DateListener) context;
    }

    public void setDate(int year, int month, int day) {

        this.year = year;
        this.month = month;
        this.day = day;
        this.isDateSetted = true;
    }

}
```

Activity calling the dialog:

```
public class MainActivity extends AppCompatActivity implements ChooseDate.DateListener{

    private int year;
    private int month;
    private int day;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        private void showDateDialog();
    }

    private void showDateDialog(){
        ChooseDate pickDialog = new ChooseDate();
        // We could set a date
        //  pickDialog.setDate(23, 10, 2016);
        pickDialog.show(getFragmentManager(), "");
    }

    @Override
    onDateSelected(int year, int month, int day){
        this.day = day;
        this.month = month;
        this.year = year;
    }
}
```

Section 62.10: Fullscreen Custom Dialog with no background and no title

in `styles.xml` add your custom style:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
    </style>
</resources>
```

Create your custom layout for the dialog: `fullscreen.xml`:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >

</RelativeLayout>
```

Then in java file you can use it for an Activity or Dialog etc:

```
import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;

public class FullscreenActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        //You can set no content for the activity.
        Dialog mDialog = new Dialog(this, R.style.AppBaseTheme);
        mDialog.setContentView(R.layout.fullscreen);
        mDialog.show();
    }
}
```

Chapter 63: Enhancing Alert Dialogs

This topic is about enhancing an `AlertDialog` with additional features.

Section 63.1: Alert dialog containing a clickable link

In order to show an alert dialog containing a link which can be opened by clicking it, you can use the following code:

```
AlertDialog.Builder builder1 = new AlertDialog.Builder(youractivity.this);

builder1.setMessage(Html.fromHtml("your message,<a href=\"http://www.google.com\">link</a>"));

builder1.setCancelable(false);
builder1.setPositiveButton("ok", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
    }
});

AlertDialog Alert1 = builder1.create();
Alert1 .show();
((TextView)Alert1.findViewById(android.R.id.message)).setMovementMethod(LinkMovementMethod.getInsta
nce());
```

Chapter 64: Animated AlertDialog Box

Animated Alert Dialog Which display with some animation effects.. You Can Get Some Animation for dialog box like Fadein, Slideleft, Slidetop, SlideBottom, Sliderright, Fall, Newspager, Fliph, Flipv, RotateBottom, RotateLeft, Slit, Shake, Sidefill to make Your application attractive..

Section 64.1: Put Below code for Animated dialog..

animated_android_dialog_box.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="16dp">

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#1184be"
    android:onClick="animatedDialog1"
    android:text="Animated Fall Dialog"
    android:textColor="#fff" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginTop="16dp"
    android:background="#1184be"
    android:onClick="animatedDialog2"
    android:text="Animated Material Flip Dialog"
    android:textColor="#fff" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#1184be"
    android:onClick="animatedDialog3"
    android:text="Animated Material Shake Dialog"
    android:textColor="#fff" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginTop="16dp"
    android:background="#1184be"
    android:onClick="animatedDialog4"
    android:text="Animated Slide Top Dialog"
    android:textColor="#fff" />
```

AnimatedAndroidDialogExample.java

```
   public class AnimatedAndroidDialogExample extends AppCompatActivity {

    NiftyDialogBuilder materialDesignAnimatedDialog;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.animated_android_dialog_box);

        materialDesignAnimatedDialog = NiftyDialogBuilder.getInstance(this);
    }

    public void animatedDialog1(View view) {
        materialDesignAnimatedDialog
                .withTitle("Animated Fall Dialog Title")
                .withMessage("Add your dialog message here. Animated dialog description place.")
                .withDialogColor("#FFFFFF")
                .withButton1Text("OK")
                .withButton2Text("Cancel")
                .withDuration(700)
                .withEffect(Effectstype.Fall)
                .show();
    }

    public void animatedDialog2(View view) {
        materialDesignAnimatedDialog
                .withTitle("Animated Flip Dialog Title")
                .withMessage("Add your dialog message here. Animated dialog description place.")
                .withDialogColor("#1c90ec")
                .withButton1Text("OK")
                .withButton2Text("Cancel")
                .withDuration(700)
                .withEffect(Effectstype.Fliph)
                .show();
    }

    public void animatedDialog3(View view) {
        materialDesignAnimatedDialog
                .withTitle("Animated Shake Dialog Title")
                .withMessage("Add your dialog message here. Animated dialog description place.")
                .withDialogColor("#1c90ec")
                .withButton1Text("OK")
                .withButton2Text("Cancel")
                .withDuration(700)
                .withEffect(Effectstype.Shake)
                .show();
    }

    public void animatedDialog4(View view) {
        materialDesignAnimatedDialog
                .withTitle("Animated Slide Top Dialog Title")
                .withMessage("Add your dialog message here. Animated dialog description place.")
                .withDialogColor("#1c90ec")
                .withButton1Text("OK")
                .withButton2Text("Cancel")
                .withDuration(700)
                .withEffect(Effectstype.Slidetop)
                .show();
    }
}
```

Add the below lines in your build.gradle to include the NifyBuilder(CustomView)

build.gradle

```
dependencies {
```

```
    compile 'com.nineoldandroids:library:2.4.0'

    compile 'com.github.sd6352051.niftydialogeffects:niftydialogeffects:1.0.0@aar'

}
```

Reference Link : https://github.com/sd6352051/NiftyDialogEffects

Chapter 65: GreenDAO

GreenDAO is an Object-Relational Mapping library to help developers use SQLite databases for persistent local storage.

Section 65.1: Helper methods for SELECT, INSERT, DELETE, UPDATE queries

This example shows a helper class that contains methods useful, when executing the queries for data. Every method here uses Java Generic's in order to be very flexible.

```
public <T> List<T> selectElements(AbstractDao<T, ?> dao) {
    if (dao == null) {
        return null;
    }
    QueryBuilder<T> qb = dao.queryBuilder();
    return qb.list();
}

public <T> void insertElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.insertOrReplaceInTx(items);
}

public <T> T insertElement(AbstractDao<T, ?> absDao, T item) {
    if (item == null || absDao == null) {
        return null;
    }
    absDao.insertOrReplaceInTx(item);
    return item;
}

public <T> void updateElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.updateInTx(items);
}

public <T> T selectElementByCondition(AbstractDao<T, ?> absDao,
                                      WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
    return items != null && items.size() > 0 ? items.get(0) : null;
}

public <T> List<T> selectElementsByCondition(AbstractDao<T, ?> absDao,
                                             WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
```

```
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSort(AbstractDao<T, ?> absDao,
                                                    Property sortProperty,
                                                    String sortStrategy,
                                                    WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    qb.orderCustom(sortProperty, sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSortWithNullHandling(AbstractDao<T, ?> absDao,
                                                                    Property sortProperty,
                                                                    boolean handleNulls,
                                                                    String sortStrategy,
                                                                    WhereCondition... conditions) {
    if (!handleNulls) {
        return selectElementsByConditionAndSort(absDao, sortProperty, sortStrategy, conditions);
    }
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    qb.orderRaw("(CASE WHEN " + "T." + sortProperty.columnName + " IS NULL then 1 ELSE 0 END)," +
"T." + sortProperty.columnName + " " + sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T, V extends Class> List<T> selectByJoin(AbstractDao<T, ?> absDao,
                                                 V className,
                                                 Property property, WhereCondition whereCondition)
{
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
    return qb.list();
}

public <T> void deleteElementsByCondition(AbstractDao<T, ?> absDao,
                                          WhereCondition... conditions) {
    if (absDao == null) {
        return;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
```

```
    }
    List<T> list = qb.list();
    absDao.deleteInTx(list);
}

public <T> T deleteElement(DaoSession session, AbstractDao<T, ?> absDao, T object) {
    if (absDao == null) {
        return null;
    }
    absDao.delete(object);
    session.clear();
    return object;
}

public <T, V extends Class> void deleteByJoin(AbstractDao<T, ?> absDao,
                                              V className,
                                              Property property, WhereCondition whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
    qb.buildDelete().executeDeleteWithoutDetachingEntities();
}

public <T> void deleteAllFromTable(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return;
    }
    absDao.deleteAll();
}

public <T> long countElements(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return 0;
    }
    return absDao.count();
}
```

Section 65.2: Creating an Entity with GreenDAO 3.X that has a Composite Primary Key

When creating a model for a table that has a composite primary key, additional work is required on the Object for the model Entity to respect those constraints.

The following example SQL table and Entity demonstrates the structure to store a review left by a customer for an item in an online store. In this example, we want the `customer_id` and `item_id` columns to be a composite primary key, allowing only one review to exist between a specific customer and item.

SQL Table

```
CREATE TABLE review (
    customer_id STRING NOT NULL,
    item_id STRING NOT NULL,
    star_rating INTEGER NOT NULL,
    content STRING,
    PRIMARY KEY (customer_id, item_id)
);
```

Usually we would use the `@Id` and `@Unique` annotations above the respective fields in the entity class, however for a composite primary key we do the following:

1. Add the `@Index` annotation inside the class-level `@Entity` annotation. The value property contains a comma-delimited list of the fields that make up the key. Use the `unique` property as shown to enforce uniqueness on the key.
2. GreenDAO requires every Entity have a **`long`** or `Long` object as a primary key. We still need to add this to the Entity class, however we do not need to use it or worry about it affecting our implementation. In the example below it is called `localID`

Entity

```
@Entity(indexes = { @Index(value = "customer_id,item_id", unique = true)})
public class Review {

    @Id(autoincrement = true)
    private Long localID;

    private String customer_id;
    private String item_id;

    @NotNull
    private Integer star_rating;

    private String content;

    public Review() {}
}
```

Section 65.3: Getting started with GreenDao v3.X

After adding the GreenDao library dependency and Gradle plugin, we need to first create an entity object.

Entity

An entity is a Plain Old Java Object *(POJO)* that models some data in the database. GreenDao will use this class to create a table in the SQLite database and automatically generate helper classes we can use to access and store data without having to write SQL statements.

```
@Entity
public class Users {

    @Id(autoincrement = true)
    private Long id;

    private String firstname;
    private String lastname;

    @Unique
    private String email;

    // Getters and setters for the fields...

}
```

One-time GreenDao setup

Each time an application is launched GreenDao needs to be initialized. GreenDao suggests keeping this code in an Application class or somewhere it will only be run once.

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

GreenDao Helper Classes

After the entity object is created, GreenDao automatically creates the helper classes used to interact with the database. These are named similarly to the name of the entity object that was created, followed by `Dao` and are retrieved from the `daoSession` object.

```
UsersDao usersDao = daoSession.getUsersDao();
```

Many typical database actions can now be performed using this Dao object with the entity object.

Query

```
String email = "jdoe@example.com";
String firstname = "John";

// Single user query WHERE email matches "jdoe@example.com"
Users user = userDao.queryBuilder()
                .where(UsersDao.Properties.Email.eq(email)).build().unique();

// Multiple user query WHERE firstname = "John"
List<Users> user = userDao.queryBuilder()
                .where(UsersDao.Properties.Firstname.eq(firstname)).build().list();
```

Insert

```
Users newUser = new User("John","Doe","jdoe@example.com");
usersDao.insert(newUser);
```

Update

```
// Modify a previously retrieved user object and update
user.setLastname("Dole");
usersDao.update(user);
```

Delete

```
// Delete a previously retrieved user object
usersDao.delete(user);
```

Chapter 66: Tools Attributes

Section 66.1: Designtime Layout Attributes

These attributes are used when the layout is rendered in Android Studio, but have no impact on the runtime.

In general you can use any Android framework attribute, just using the `tools:` namespace rather than the `android:` namespace for layout preview. You can add both the `android:` namespace attribute (which is used at runtime) and the matching `tools:` attribute (which overrides the runtime attribute in the layout preview only).

Just define the tools namespace as described in the remarks section.

For example the `text` attribute:

```
<EditText
     tools:text="My Text"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content" />
```

Or the `visibility` attribute to unset a view for preview:

```
<LinearLayout
       android:id="@+id/ll1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       tools:visibility="gone" />
```

Or the `context` attribute to associate the layout with activity or fragment

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   tools:context=".MainActivity" >
```

Or the `showIn` attribute to see and included layout preview in another layout

```
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/text"
   tools:showIn="@layout/activity_main" />
```

Chapter 67: Formatting Strings

Section 67.1: Format a string resource

You can add wildcards in string resources and populate them at runtime:

1. Edit strings.xml

```
<string name="my_string">This is %1$s</string>
```

2. Format string as needed

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

Section 67.2: Formatting data types to String and vise versa

Data types to string formatting

Data types like int, float, double, long, boolean can be formatted to string using String.valueOf().

```
String.valueOf(1); //Output -> "1"
String.valueOf(1.0); //Output -> "1.0"
String.valueOf(1.2345); //Output -> "1.2345"
String.valueOf(true); //Output -> "true"
```

Vise versa of this, formatting string to other data type

```
Integer.parseInt("1"); //Output -> 1
Float.parseFloat("1.2"); //Output -> 1.2
Boolean.parseBoolean("true"); //Output -> true
```

Section 67.3: Format a timestamp to string

For full description of patterns, see SimpleDateFormat reference

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

Chapter 68: SpannableString

Section 68.1: Add styles to a TextView

In the following example, we create an Activity to display a single TextView.

The TextView will use a `SpannableString` as its content, which will illustrate some of the available styles.

Here' what we're gonna do with the text :

- Make it larger
- Bold
- Underline
- Italicize
- Strike-through
- Colored
- Highlighted
- Show as superscript
- Show as subscript
- Show as a link
- Make it clickable.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

 SpannableString styledString
  = new SpannableString("Large\n\n"     // index 0 - 5
       + "Bold\n\n"           // index 7 - 11
       + "Underlined\n\n"     // index 13 - 23
       + "Italic\n\n"         // index 25 - 31
       + "Strikethrough\n\n" // index 33 - 46
       + "Colored\n\n"        // index 48 - 55
       + "Highlighted\n\n"    // index 57 - 68
       + "K Superscript\n\n" // "Superscript" index 72 - 83
       + "K Subscript\n\n"    // "Subscript" index 87 - 96
       + "Url\n\n"            //  index 98 - 101
       + "Clickable\n\n");    // index 103 - 112

 // make the text twice as large
 styledString.setSpan(new RelativeSizeSpan(2f), 0, 5, 0);

 // make text bold
 styledString.setSpan(new StyleSpan(Typeface.BOLD), 7, 11, 0);

 // underline text
 styledString.setSpan(new UnderlineSpan(), 13, 23, 0);

 // make text italic
 styledString.setSpan(new StyleSpan(Typeface.ITALIC), 25, 31, 0);

 styledString.setSpan(new StrikethroughSpan(), 33, 46, 0);

 // change text color
 styledString.setSpan(new ForegroundColorSpan(Color.GREEN), 48, 55, 0);

 // highlight text
 styledString.setSpan(new BackgroundColorSpan(Color.CYAN), 57, 68, 0);
```

```
 // superscript
 styledString.setSpan(new SuperscriptSpan(), 72, 83, 0);
 // make the superscript text smaller
 styledString.setSpan(new RelativeSizeSpan(0.5f), 72, 83, 0);

 // subscript
 styledString.setSpan(new SubscriptSpan(), 87, 96, 0);
 // make the subscript text smaller
 styledString.setSpan(new RelativeSizeSpan(0.5f), 87, 96, 0);

 // url
 styledString.setSpan(new URLSpan("http://www.google.com"), 98, 101, 0);

 // clickable text
 ClickableSpan clickableSpan = new ClickableSpan() {

 @Override
 public void onClick(View widget) {
// We display a Toast. You could do anything you want here.
Toast.makeText(SpanExample.this, "Clicked", Toast.LENGTH_SHORT).show();

  }
};

 styledString.setSpan(clickableSpan, 103, 112, 0);


 // Give the styled string to a TextView
 TextView textView = new TextView(this);

 // this step is mandated for the url and clickable styles.
 textView.setMovementMethod(LinkMovementMethod.getInstance());

 // make it neat
textView.setGravity(Gravity.CENTER);
textView.setBackgroundColor(Color.WHITE);

textView.setText(styledString);

setContentView(textView);

}
```

And the result will look like this:

Section 68.2: Multi string , with multi color

Method: **setSpanColor**

```
public Spanned setSpanColor(String string, int color){
    SpannableStringBuilder builder = new SpannableStringBuilder();
    SpannableString ss = new SpannableString(string);
    ss.setSpan(new ForegroundColorSpan(color), 0, string.length(), 0);
    builder.append(ss);
    return ss;
}
```

Usage:

```
String a = getString(R.string.string1);
String b = getString(R.string.string2);

Spanned color1 = setSpanColor(a,Color.CYAN);
Spanned color2 = setSpanColor(b,Color.RED);
Spanned mixedColor = TextUtils.concat(color1, " ", color2);
// Now we use `mixedColor`
```

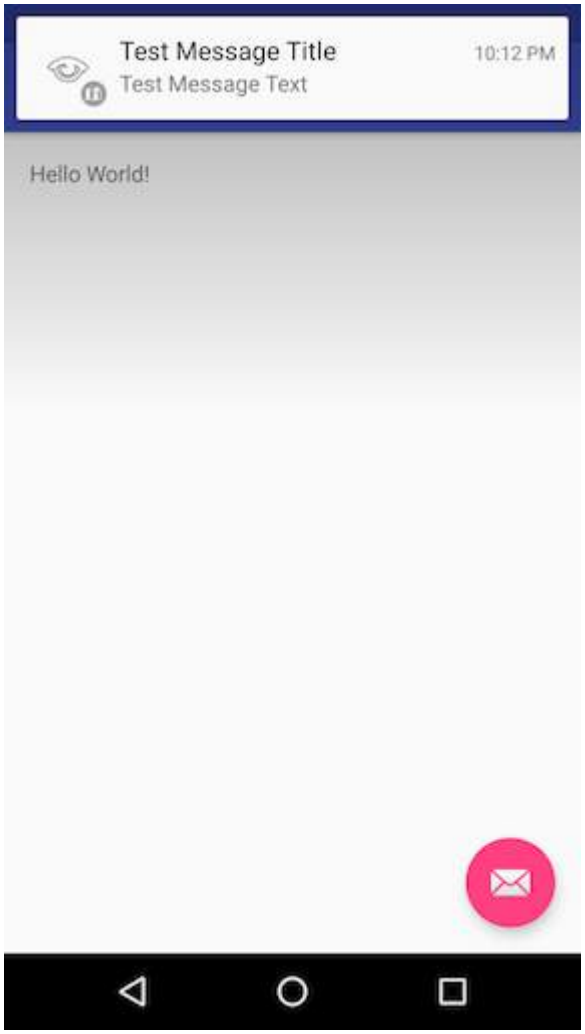
Chapter 69: Notifications

Section 69.1: Heads Up Notification with Ticker for older devices

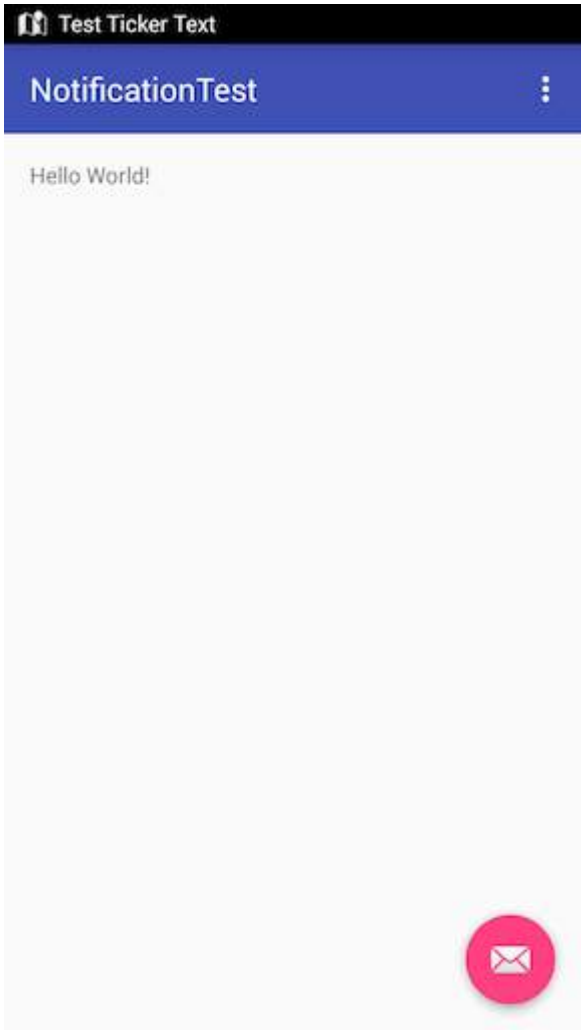
Here is how to make a Heads Up Notification for capable devices, and use a Ticker for older devices.

```
// Tapping the Notification will open up MainActivity
Intent i = new Intent(this, MainActivity.class);

// an action to use later
// defined as an app constant:
// public static final String MESSAGE_CONSTANT = "com.example.myapp.notification";
i.setAction(MainActivity.MESSAGE_CONSTANT);
// you can use extras as well
i.putExtra("some_extra", "testValue");

i.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT | Intent.FLAG_ACTIVITY_SINGLE_TOP);
PendingIntent notificationIntent = PendingIntent.getActivity(this, 999, i,
PendingIntent.FLAG_UPDATE_CURRENT);
NotificationCompat.Builder builder = new NotificationCompat.Builder(this.getApplicationContext());
builder.setContentIntent(notificationIntent);
builder.setAutoCancel(true);
builder.setLargeIcon(BitmapFactory.decodeResource(this.getResources(),
android.R.drawable.ic_menu_view));
builder.setSmallIcon(android.R.drawable.ic_dialog_map);
builder.setContentText("Test Message Text");
builder.setTicker("Test Ticker Text");
builder.setContentTitle("Test Message Title");

// set high priority for Heads Up Notification
builder.setPriority(NotificationCompat.PRIORITY_HIGH);
builder.setVisibility(NotificationCompat.VISIBILITY_PUBLIC);

// It won't show "Heads Up" unless it plays a sound
if (Build.VERSION.SDK_INT >= 21) builder.setVibrate(new long[0]);

NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(999, builder.build());
```

Here is what it looks like on Android Marshmallow with the Heads Up Notification:

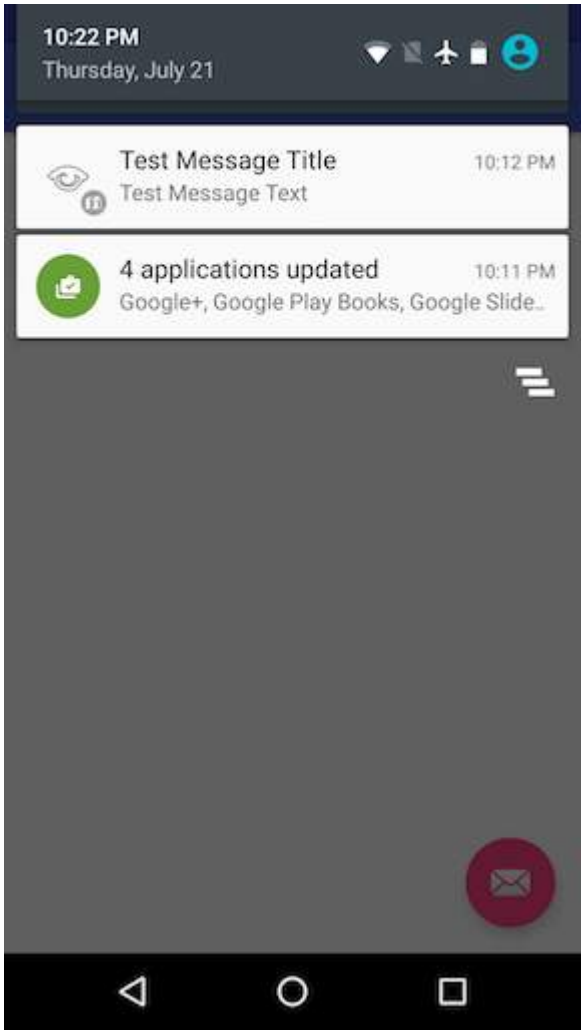


Here is what it looks like on Android KitKat with the Ticker:

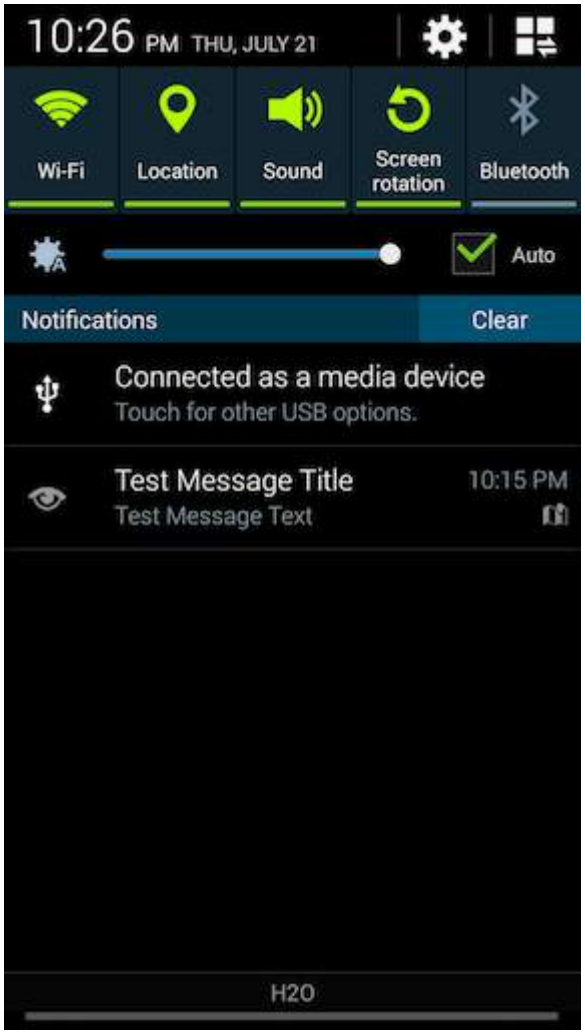


On all Android versions, the `Notification` is shown in the notification drawer.

Android 6.0 Marshmallow:



Android 4.4.x KitKat:



Section 69.2: Creating a simple Notification

This example shows how to create a simple notification that starts an application when the user clicks it.

Specify the notification's content:

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.ic_launcher) // notification icon
        .setContentTitle("Simple notification") // title
        .setContentText("Hello word") // body message
        .setAutoCancel(true); // clear notification when clicked
```

Create the intent to fire on click:

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

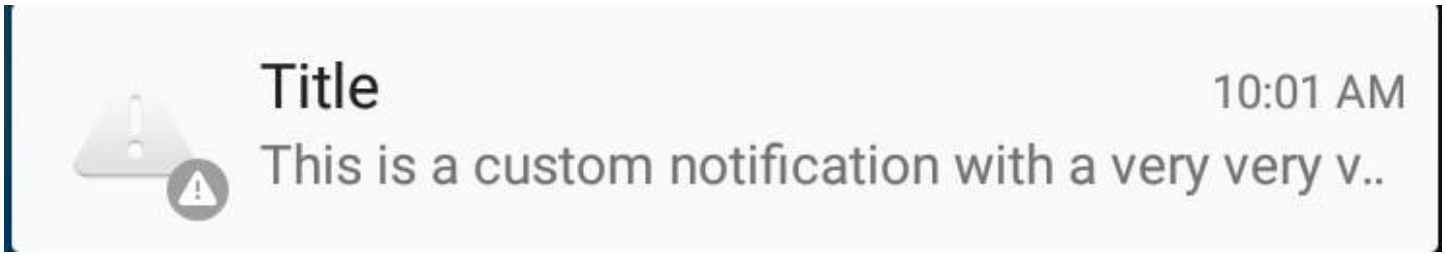
Finally, build the notification and show it

```
NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

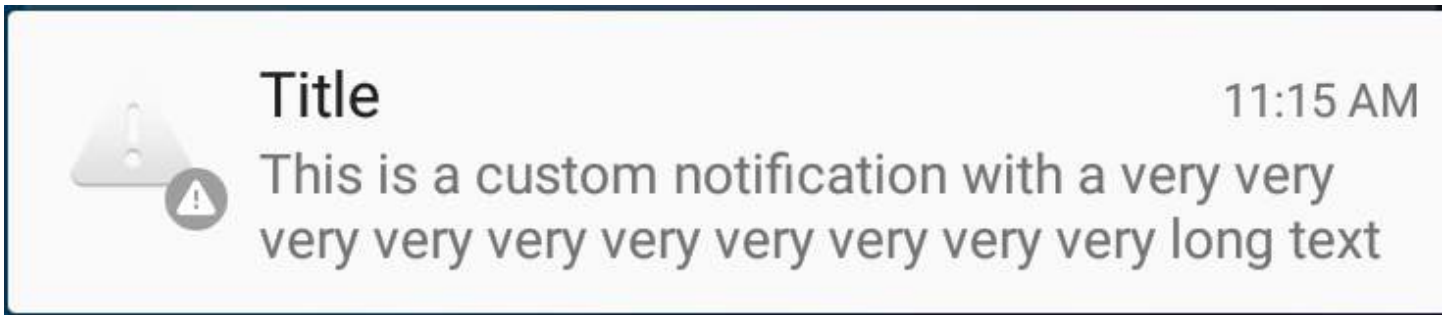
Section 69.3: Set custom notification - show full content text

If you want have a long text to display in the context, you need to set a custom content.

For example, you have this:



But you wish your text will be fully shown:



All you need to do, is to **add a style** to your content like below:

```
  private void generateNotification(Context context) {
        String message = "This is a custom notification with a very very very very very very very
very very very very long text";
        Bitmap largeIcon = BitmapFactory.decodeResource(getResources(),
android.R.drawable.ic_dialog_alert);

        NotificationCompat.Builder builder = new NotificationCompat.Builder(context);

        builder.setContentTitle("Title").setContentText(message)
                .setSmallIcon(android.R.drawable.ic_dialog_alert)
                .setLargeIcon(largeIcon)
                .setAutoCancel(true)
                .setWhen(System.currentTimeMillis())
                .setStyle(new NotificationCompat.BigTextStyle().bigText(message));

        Notification notification = builder.build();
        NotificationManagerCompat notificationManager = NotificationManagerCompat.from(context);
        notificationManager.notify(101, notification);
    }
```

Section 69.4: Dynamically getting the correct pixel size for the large icon

If you're creating an image, decoding an image, or resizing an image to fit the large notification image area, you can get the correct pixel dimensions like so:

```
Resources resources = context.getResources();
int width  = resources.getDimensionPixelSize(android.R.dimen.notification_large_icon_width);
int height = resources.getDimensionPixelSize(android.R.dimen.notification_large_icon_height);
```

Section 69.5: Ongoing notification with Action button

```
// Cancel older notification with same id,
NotificationManager notificationMgr =
```

```
(NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);
notificationMgr.cancel(CALL_NOTIFY_ID);// any constant value

// Create Pending Intent,
Intent notificationIntent = null;
PendingIntent contentIntent = null;
notificationIntent = new Intent (context, YourActivityName);
contentIntent = PendingIntent.getActivity(context, 0, notificationIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

// Notification builder
builder = new NotificationCompat.Builder(context);
builder.setContentText("Ongoing Notification..");
builder.setContentTitle("ongoing notification sample");
builder.setSmallIcon(R.drawable.notification_icon);
builder.setUsesChronometer(true);
builder.setDefaults(Notification.DEFAULT_LIGHTS);
builder.setContentIntent(contentIntent);
builder.setOngoing(true);

// Add action button in the notification
Intent intent = new Intent("action.name");
PendingIntent pIntent = PendingIntent.getBroadcast(context, 1, intent, 0);
builder.addAction(R.drawable.action_button_icon, "Action button name",pIntent);

// Notify using notificationMgr
Notification finalNotification = builder.build();
notificationMgr.notify(CALL_NOTIFY_ID, finalNotification);
```

Register a broadcast receiver for the same action to handle action button click event.

Section 69.6: Setting Different priorities in notification

```
NotificationCompat.Builder mBuilder =

        (NotificationCompat.Builder) new NotificationCompat.Builder(context)

        .setSmallIcon(R.drawable.some_small_icon)
        .setContentTitle("Title")
        .setContentText("This is a test notification with MAX priority")
        .setPriority(Notification.PRIORITY_MAX);
```

> When notification contains image and you want to auto expand image when notification received use "PRIORITY_MAX", you can use other priority levels as per requirments

Different Priority Levels Info:

PRIORITY_MAX -- Use for critical and urgent notifications that alert the user to a condition that is time-critical or needs to be resolved before they can continue with a particular task.

PRIORITY_HIGH -- Use primarily for important communication, such as message or chat events with content that is particularly interesting for the user. High-priority notifications trigger the heads-up notification display.

PRIORITY_DEFAULT -- Use for all notifications that don't fall into any of the other priorities described here.

PRIORITY_LOW -- Use for notifications that you want the user to be informed about, but that are less urgent. Low-priority notifications tend to show up at the bottom of the list, which makes them a good choice for things like

public or undirected social updates: The user has asked to be notified about them, but these notifications should never take precedence over urgent or direct communication.

PRIORITY_MIN -- Use for contextual or background information such as weather information or contextual location information. Minimum-priority notifications do not appear in the status bar. The user discovers them on expanding the notification shade.

References: Material Design Guidelines - notifications

Section 69.7: Set custom notification icon using `Picasso` library

```
    PendingIntent pendingIntent = PendingIntent.getActivity(context,
    uniqueIntentId, intent, PendingIntent.FLAG_CANCEL_CURRENT);

    final RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
R.layout.remote_view_notification);
    remoteViews.setImageViewResource(R.id.remoteview_notification_icon,
R.mipmap.ic_navigation_favorites);

    Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder(context)
                    .setSmallIcon(R.mipmap.ic_navigation_favorites) //just dummy icon
                    .setContent(remoteViews) // here we apply our view
                    .setAutoCancel(true)
                    .setContentIntent(pendingIntent)
                    .setPriority(NotificationCompat.PRIORITY_DEFAULT);

    final Notification notification = notificationBuilder.build();

    if (android.os.Build.VERSION.SDK_INT >= 16) {
        notification.bigContentView = remoteViews;
    }

    NotificationManager notificationManager =
            (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);

    notificationManager.notify(uniqueIntentId, notification);


    //don't forget to include picasso to your build.gradle file.
    Picasso.with(context)
            .load(avatar)
            .into(remoteViews, R.id.remoteview_notification_icon, uniqueIntentId, notification);
```

And then define a layout inside your layouts folder:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="@android:color/white"
   android:orientation="vertical">



    <ImageView
```

```
        android:id="@+id/remoteview_notification_icon"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_marginRight="2dp"
        android:layout_weight="0"
        android:scaleType="centerCrop"/>
</LinearLayout>
```

Section 69.8: Scheduling notifications

Sometimes it is required to display a notification at a specific time, a task that unfortunately is not trivial on the Android system, as there is no method `setTime()` or similiar for notifications. This example outlines the steps needed to schedule notifications using the `AlarmManager`:

1. **Add a `BroadcastReceiver`** that listens to `Intents` broadcasted by the Android `AlarmManager`.

This is the place where you build your notification based on the extras provided with the `Intent`:

```
   public class NotificationReceiver extends BroadcastReceiver {
       @Override
       public void onReceive(Context context, Intent intent) {
           // Build notification based on Intent
           Notification notification = new NotificationCompat.Builder(context)
               .setSmallIcon(R.drawable.ic_notification_small_icon)
               .setContentTitle(intent.getStringExtra("title", ""))
               .setContentText(intent.getStringExtra("text", ""))
               .build();
           // Show notification
           NotificationManager manager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
           manager.notify(42, notification);
       }
   }
```

2. **Register the `BroadcastReceiver`** in your `AndroidManifest.xml` file (otherwise the receiver won't receive any `Intents` from the `AlarmManager`):

   ```
   <receiver
      android:name=".NotificationReceiver"
      android:enabled="true" />
   ```

3. **Schedule a notification** by passing a `PendingIntent` for your `BroadcastReceiver` with the needed `Intent` extras to the system `AlarmManager`. Your `BroadcastReceiver` will receive the `Intent` once the given time has arrived and display the notification. The following method schedules a notification:

   ```
   public static void scheduleNotification(Context context, long time, String title, String
   text) {
       Intent intent = new Intent(context, NotificationReceiver.class);
       intent.putExtra("title", title);
       intent.putExtra("text", text);
       PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
   PendingIntent.FLAG_UPDATE_CURRENT);
       // Schdedule notification
       AlarmManager manager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
       manager.setExactAndAllowWhileIdle(AlarmManager.RTC_WAKEUP, time, pending);
   }
   ```

Please note that the 42 above needs to be unique for each scheduled notification, otherwise the `PendingIntents` will replace each other causing undesired effects!

4. **Cancel a notification** by rebuilding the associated `PendingIntent` and canceling it on the system `AlarmManager`. The following method cancels a notification:

```
public static void cancelNotification(Context context, String title, String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    // Cancel notification
    AlarmManager manager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
    manager.cancel(pending);
}
```

Note that the `42` above needs to match the number from step 3!

Chapter 70: AlarmManager

Section 70.1: How to Cancel an Alarm

If you want to cancel an alarm, and you don't have a reference to the original PendingIntent used to set the alarm, you need to recreate a PendingIntent exactly as it was when it was originally created.

An Intent is considered equal by the AlarmManager:

> if their action, data, type, class, and categories are the same. This does not compare any extra data included in the intents.

Usually the request code for each alarm is defined as a constant:

```
public static final int requestCode = 9999;
```

So, for a simple alarm set up like this:

```
Intent intent  = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
AlarmManager alarmManager = (AlarmManager)getSystemService(Context.ALARM_SERVICE);
alarmManager.setExact(AlarmManager.RTC_WAKEUP, targetTimeInMillis, pendingIntent);
```

Here is how you would create a new PendingIntent reference that you can use to cancel the alarm with a new AlarmManager reference:

```
Intent intent  = new Intent(this, AlarmReceiver.class);
intent.setAction("SomeAction");
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, requestCode, intent,
PendingIntent.FLAG_NO_CREATE);
AlarmManager alarmManager = (AlarmManager)getSystemService(Context.ALARM_SERVICE);
if(pendingIntent != null) {
    alarmManager.cancel(pendingIntent);
}
```

Section 70.2: Creating exact alarms on all Android versions

With more and more battery optimizations being put into the Android system over time, the methods of the `AlarmManager` have also significantly changed (to allow for more lenient timing). However, for some applications it is still required to be as exact as possible on all Android versions. The following helper uses the most accurate method available on all platforms to schedule a `PendingIntent`:

```
public static void setExactAndAllowWhileIdle(AlarmManager alarmManager, int type, long
triggerAtMillis, PendingIntent operation) {
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M){
        alarmManager.setExactAndAllowWhileIdle(type, triggerAtMillis, operation);
    } else if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT){
        alarmManager.setExact(type, triggerAtMillis, operation);
    } else {
        alarmManager.set(type, triggerAtMillis, operation);
    }
```

```
}
```

Section 70.3: API23+ Doze mode interferes with AlarmManager

Android 6 (API23) introduced Doze mode which interferes with AlarmManager. It uses certain maintenance windows to handle alarms, so even if you used `setExactAndAllowWhileIdle()` you cannot make sure that your alarm fires at the desired point of time.

You can turn this behavior off for your app using your phone's settings (`Settings/General/Battery & power saving/Battery usage/Ignore optimizations` or similar)

Inside your app you can check this setting ...

```
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
if (pm.isIgnoringBatteryOptimizations(packageName)) {
   // your app is ignoring Doze battery optimization
}
```

... and eventually show the respective settings dialog:

```
Intent intent = new Intent();
String packageName = getPackageName();
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
intent.setData(Uri.parse("package:" + packageName));
startActivity(intent);
```

Section 70.4: Run an intent at a later time

1. Create a receiver. This class will receive the intent and handle it how you wish.

```
public class AlarmReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        // Handle intent
        int reqCode = intent.getExtras().getInt("requestCode");
        ...
    }
}
```

2. Give an intent to AlarmManager. This example will trigger the intent to be sent to AlarmReceiver after 1 minute.

```
final int requestCode = 1337;
AlarmManager am = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(context, AlarmReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(context, requestCode, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
am.set( AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + 60000 , pendingIntent );
```

Chapter 71: Handler

Section 71.1: HandlerThreads and communication between Threads

As `Handlers` are used to send `Messages` and `Runnables` to a Thread's message queue it's easy to implement event based communication between multiple Threads. Every Thread that has a `Looper` is able to receive and process messages. A `HandlerThread` is a Thread that implements such a `Looper`, for example the main Thread (UI Thread) implements the features of a `HandlerThread`.

Creating a Handler for the current Thread

```
Handler handler = new Handler();
```

Creating a Handler for the main Thread (UI Thread)

```
Handler handler = new Handler(Looper.getMainLooper());
```

Send a Runnable from another Thread to the main Thread

```
new Thread(new Runnable() {
    public void run() {
        // this is executed on another Thread

        // create a Handler associated with the main Thread
        Handler handler = new Handler(Looper.getMainLooper());

        // post a Runnable to the main Thread
        handler.post(new Runnable() {
            public void run() {
                // this is executed on the main Thread
            }
        });
    }
}).start();
```

Creating a Handler for another HandlerThread and sending events to it

```
// create another Thread
HandlerThread otherThread = new HandlerThread("name");

// create a Handler associated with the other Thread
Handler handler = new Handler(otherThread.getLooper());

// post an event to the other Thread
handler.post(new Runnable() {
    public void run() {
        // this is executed on the other Thread
    }
});
```

Section 71.2: Use Handler to create a Timer (similar to javax.swing.Timer)

This can be useful if you're writing a game or something that needs to execute a piece of code every a few seconds.

```
import android.os.Handler;
```

```
public class Timer {
    private Handler handler;
    private boolean paused;

    private int interval;

    private Runnable task = new Runnable () {
        @Override
        public void run() {
            if (!paused) {
                runnable.run ();
                Timer.this.handler.postDelayed (this, interval);
            }
        }
    };

    private Runnable runnable;

    public int getInterval() {
        return interval;
    }

    public void setInterval(int interval) {
        this.interval = interval;
    }

    public void startTimer () {
        paused = false;
        handler.postDelayed (task, interval);
    }

    public void stopTimer () {
        paused = true;
    }

    public Timer (Runnable runnable, int interval, boolean started) {
        handler = new Handler ();
        this.runnable = runnable;
        this.interval = interval;
        if (started)
            startTimer ();
    }
}
```

Example usage:

```
Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)
```

This code will print "Hello" every second.

Section 71.3: Using a Handler to execute code after a delayed amount of time

Executing code after 1.5 seconds:

```
Handler handler = new Handler();
```

```
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        //The code you want to run after the time is up
    }
}, 1500); //the time you want to delay in milliseconds
```

Executing code repeatedly every 1 second:

```
Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        handler.postDelayed(this, 1000);
    }
}, 1000); //the time you want to delay in milliseconds
```

Section 71.4: Stop handler from execution

To stop the Handler from execution remove the callback attached to it using the runnable running inside it:

```
Runnable my_runnable = new Runnable() {
    @Override
    public void run() {
        // your code here
    }
};

public Handler handler = new Handler(); // use 'new Handler(Looper.getMainLooper());' if you want
this handler to control something in the UI
// to start the handler
public void start() {
    handler.postDelayed(my_runnable, 10000);
}

// to stop the handler
public void stop() {
    handler.removeCallbacks(my_runnable);
}

// to reset the handler
public void restart() {
    handler.removeCallbacks(my_runnable);
    handler.postDelayed(my_runnable, 10000);
}
```

Chapter 72: BroadcastReceiver

BroadcastReceiver (receiver) is an Android component which allows you to register for system or application events. All registered receivers for an event are notified by the Android runtime once this event happens.

for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured.

Applications can also initiate broadcasts—for example, to let other applications know that some data has been downloaded to the device and is available for them to use.

Section 72.1: Using LocalBroadcastManager

LocalBroadcastManager is used to send Broadcast Intents within an application, without exposing them to unwanted listeners.

Using *LocalBroadcastManager* is more efficient and safer than using `context.sendBroadcast()` directly, because you don't need to worry about any broadcasts faked by other Applications, which may pose a security hazard.

Here is a simple example of sending and receiving local broadcasts:

```
BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("Some Action")) {
            //Do something
        }
    }
});

LocalBroadcastManager manager = LocalBroadcastManager.getInstance(mContext);
manager.registerReceiver(receiver, new IntentFilter("Some Action"));

// onReceive() will be called as a result of this call:
manager.sendBroadcast(new Intent("Some Action"));//See also sendBroadcastSync

//Remember to unregister the receiver when you are done with it:
manager.unregisterReceiver(receiver);
```

Section 72.2: BroadcastReceiver Basics

BroadcastReceivers are used to receive broadcast Intents that are sent by the Android OS, other apps, or within the same app.

Each Intent is created with an *Intent Filter*, which requires a String *action*. Additional information can be configured in the Intent.

Likewise, BroadcastReceivers register to receive Intents with a particular Intent Filter. They can be registered programmatically:

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}, new IntentFilter("Some Action"));
```

or in the `AndroidManifest.xml` file:

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

To receive the Intent, set the Action to something documented by Android OS, by another app or API, or within your own application, using `sendBroadcast`:

```
mContext.sendBroadcast(new Intent("Some Action"));
```

Additionally, the Intent can contain information, such as Strings, primitives, and *Parcelables*, that can be viewed in `onReceive`.

Section 72.3: Introduction to Broadcast receiver

A Broadcast receiver is an Android component which allows you to register for system or application events.

A receiver can be registered via the `AndroidManifest.xml` file or dynamically via the `Context.registerReceiver()` method.

```
public class MyReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}
```

Here I have taken an example of `ACTION_BOOT_COMPLETED` which is fired by the system once the Android has completed the boot process.

You can register a receiver in manifest file like this:

```
<application
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name"
  android:theme="@style/AppTheme" >
   <receiver android:name="MyReceiver">
      <intent-filter>
         <action android:name="android.intent.action.BOOT_COMPLETED">
         </action>
      </intent-filter>
   </receiver>
</application>
```

Now device gets booted, `onReceive()` method will be called and then you can do your work (e.g. start a service, start an alarm).

Section 72.4: Using ordered broadcasts

Ordered broadcasts are used when you need to specify a priority for broadcast listeners.

In this example `firstReceiver` will receive broadcast always before than a `secondReceiver`:

```
final int highPriority = 2;
final int lowPriority = 1;
final String action = "action";

// intent filter for first receiver with high priority
final IntentFilter firstFilter = new IntentFilter(action);
first Filter.setPriority(highPriority);
final BroadcastReceiver firstReceiver = new MyReceiver();

// intent filter for second receiver with low priority
final IntentFilter secondFilter = new IntentFilter(action);
secondFilter.setPriority(lowPriority);
final BroadcastReceiver secondReceiver = new MyReceiver();

// register our receivers
context.registerReceiver(firstReceiver, firstFilter);
context.registerReceiver(secondReceiver, secondFilter);

// send ordered broadcast
context.sendOrderedBroadcast(new Intent(action), null);
```

Furthermore broadcast receiver can abort ordered broadcast:

```
@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}
```

in this case all receivers with lower priority will not receive a broadcast message.

Section 72.5: Sticky Broadcast

If we are using method sendStickyBroadcast(intent) the corresponding intent is sticky, meaning the intent you are sending stays around after broadcast is complete. A StickyBroadcast as the name suggests is a mechanism to read the data from a broadcast, after the broadcast is complete. This can be used in a scenario where you may want to check say in an `Activity's onCreate()` the value of a key in the intent before that Activity was launched.

```
Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);
```

Section 72.6: Enabling and disabling a Broadcast Receiver programmatically

To enable or disable a `BroadcastReceiver`, we need to get a reference to the `PackageManager` and we need a `ComponentName` object containing the class of the receiver we want to enable/disable:

```
ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();
```

Now we can call the following method to enable the `BroadcastReceiver`:

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

Or we can instead use `COMPONENT_ENABLED_STATE_DISABLED` to disable the receiver:

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP);
```

Section 72.7: Example of a LocalBroadcastManager

A BroadcastReceiver is basically a mechanism to relay Intents through the OS to perform specific actions. A classic definition being

> "A Broadcast receiver is an Android component which allows you to register for system or application events."

LocalBroadcastManager is a way to send or receive broadcasts within an application process. This mechanism has a lot of advantages

1. since the data remains inside the application process, the data cannot be leaked.
2. LocalBroadcasts are resolved faster, since the resolution of a normal broadcast happens at runtime throughout the OS.

A simple example of a LocalBroastManager is:

SenderActivity

```
Intent intent = new Intent("anEvent");
intent.putExtra("key", "This is an event");
LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

ReceiverActivity

1. Register a receiver

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,
              new IntentFilter("anEvent"));
```

2. A concrete object for performing action when the receiver is called

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // perform action here.
    }
};
```

3. unregister when the view is not visible any longer.

```
@Override
protected void onPause() {
    // Unregister since the activity is about to be closed.
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);
    super.onDestroy();
}
```

Section 72.8: Android stopped state

Starting with Android 3.1 all applications, upon installation, are placed in a stopped state. While in stopped state, the application will not run for any reason, except by a manual launch of an activity, or an **explicit** intent that addresses an activity ,service or broadcast.

When writing system app that installs APKs directly, please take into account that the newly installed APP won't receive any broadcasts until moved into a non stopped state.

An easy way to to activate an app is to sent a explicit broadcast to this app. as most apps implement `INSTALL_REFERRER`, we can use it as a hooking point

Scan the manifest of the installed app, and send an explicit broadcast to to each receiver:

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

Section 72.9: Communicate two activities through custom Broadcast receiver

You can communicate two activities so that Activity A can be notified of an event happening in Activity B.

Activity A

```
final String eventName = "your.package.goes.here.EVENT";

@Override
protected void onCreate(Bundle savedInstanceState) {
    registerEventReceiver();
    super.onCreate(savedInstanceState);
}

@Override
protected void onDestroy() {
    unregisterEventReceiver(eventReceiver);
    super.onDestroy();
}

private void registerEventReceiver() {
    IntentFilter eventFilter = new IntentFilter();
    eventFilter.addAction(eventName);
    registerReceiver(eventReceiver, eventFilter);
}

private BroadcastReceiver eventReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
         //This code will be executed when the broadcast in activity B is launched
    }
};
```

Activity B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
```

```
    Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}
```

Of course you can add more information to the broadcast adding extras to the Intent that is passed between the activities. Not added to keep the example as simple as possible.

Section 72.10: BroadcastReceiver to handle BOOT_COMPLETED events

Example below shows how to create a `BroadcastReceiver` which is able to receive `BOOT_COMPLETED` events. This way, you are able to start a `Service` or start an `Activity` as soon device was powered up.

Also, you can use `BOOT_COMPLETED` events to restore your alarms since they are destroyed when device is powered off.

NOTE: The user needs to have started the application at least once before you can receive the `BOOT_COMPLETED` action.

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.test.example" >
    ...
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    ...

    <application>
        ...

        <receiver android:name="com.test.example.MyCustomBroadcastReceiver">
        <intent-filter>
                <!-- REGISTER TO RECEIVE BOOT_COMPLETED EVENTS -->
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

MyCustomBroadcastReceiver.java

```
public class MyCustomBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if(action != null) {
            if (action.equals(Intent.ACTION_BOOT_COMPLETED) ) {
                // TO-DO: Code to handle BOOT COMPLETED EVENT
                // TO-DO: I can start an service.. display a notification... start an activity
            }
        }
    }
}
```

Section 72.11: Bluetooth Broadcast receiver

add permission in your manifest file

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

In your Fragment(or Activity)

- Add the receiver method

```
private BroadcastReceiver mBluetoothStatusChangedReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        final Bundle extras = intent.getExtras();
        final int bluetoothState = extras.getInt(Constants.BUNDLE_BLUETOOTH_STATE);
        switch(bluetoothState) {
            case BluetoothAdapter.STATE_OFF:
                // Bluetooth OFF
                break;
            case BluetoothAdapter.STATE_TURNING_OFF:
                // Turning OFF
                break;
            case BluetoothAdapter.STATE_ON:
                // Bluetooth ON
                break;
            case BluetoothAdapter.STATE_TURNING_ON:
                // Turning ON
                break;
    }
};
```

Register broadcast

- Call this method on onResume()

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
}
```

Unregister broadcast

- Call this method on onPause()

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // Beacon機能用
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

Chapter 73: UI Lifecycle

Section 73.1: Saving data on memory trimming

```
public class ExampleActivity extends Activity {

    private final static String EXAMPLE_ARG = "example_arg";
    private int mArg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_example);

        if(savedInstanceState != null) {
            mArg = savedInstanceState.getInt(EXAMPLE_ARG);
        }
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt(EXAMPLE_ARG, mArg);
    }
}
```

Explanation

So, what is happening here?

The Android system will always strive to clear as much memory as it can. So, if your activity is down to the background, and another foreground activity is demanding its share, the Android system will call `onTrimMemory()` on your activity.

But that doesn't mean that all your properties should vanish. What you should do is to save them into a Bundle object. Bundle object are much better handled memory wise. Inside a bundle every object is identified by unique text sequence - in the example above integer value variable `mArg` is hold under reference name `EXAMPLE_ARG`. And when the activity is recreated extract your old values from the Bundle object instead of recreating them from scratch

Chapter 74: HttpURLConnection

Section 74.1: Creating an HttpURLConnection

In order to create a new Android HTTP Client `HttpURLConnection`, call `openConnection()` on a URL instance. Since `openConnection()` returns a `URLConnection`, you need to explicitly cast the returned value.

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// do something with the connection
```

If you are creating a new `URL`, you also have to handle the exceptions associated with URL parsing.

```
try {
    URL url = new URL("http://example.com");
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // do something with the connection
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

Once the response body has been read and the connection is no longer required, the connection should be closed by calling `disconnect()`.

Here is an example:

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // do something with the connection
} finally {
    connection.disconnect();
}
```

Section 74.2: Sending an HTTP GET request

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // read the input stream
    // in this case, I simply read the first line of the stream
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} finally {
    connection.disconnect();
}
```

Please note that exceptions are not handled in the example above. A full example, including (a trivial) exception handling, would be:

```
URL url;
HttpURLConnection connection = null;
```

```
try {
    url = new URL("http://example.com");
    connection = (HttpURLConnection) url.openConnection();
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // read the input stream
    // in this case, I simply read the first line of the stream
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (connection != null) {
        connection.disconnect();
    }
}
```

Section 74.3: Reading the body of an HTTP GET request

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

    // use a string builder to bufferize the response body
    // read from the input strea.
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line).append('\n');
    }

    // use the string builder directly,
    // or convert it into a String
    String body = sb.toString();

    Log.d("HTTP-GET", body);

} finally {
    connection.disconnect();
}
```

Please note that exceptions are not handled in the example above.

Section 74.4: Sending an HTTP POST request with parameters

Use a HashMap to store the parameters that should be sent to the server through POST parameters:

```
HashMap<String, String> params;
```

Once the `params` HashMap is populated, create the StringBuilder that will be used to send them to the server:

```
    StringBuilder sbParams = new StringBuilder();
    int i = 0;
    for (String key : params.keySet()) {
        try {
```

```
            if (i != 0){
                sbParams.append("&");
            }
            sbParams.append(key).append("=")
                    .append(URLEncoder.encode(params.get(key), "UTF-8"));

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        i++;
    }
```

Then, create the HttpURLConnection, open the connection, and send the POST parameters:

```
try{
    String url = "http://www.example.com/test.php";
    URL urlObj = new URL(url);
    HttpURLConnection conn = (HttpURLConnection) urlObj.openConnection();
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Accept-Charset", "UTF-8");

    conn.setReadTimeout(10000);
    conn.setConnectTimeout(15000);

    conn.connect();

    String paramsString = sbParams.toString();

    DataOutputStream wr = new DataOutputStream(conn.getOutputStream());
    wr.writeBytes(paramsString);
    wr.flush();
    wr.close();
} catch (IOException e) {
  e.printStackTrace();
}
```

Then receive the result that the server sends back:

```
try {
  InputStream in = new BufferedInputStream(conn.getInputStream());
  BufferedReader reader = new BufferedReader(new InputStreamReader(in));
  StringBuilder result = new StringBuilder();
  String line;
  while ((line = reader.readLine()) != null) {
    result.append(line);
  }

  Log.d("test", "result from server: " + result.toString());

} catch (IOException e) {
  e.printStackTrace();
} finally {
    if (conn != null) {
        conn.disconnect();
    }
}
```

Section 74.5: A multi-purpose HttpURLConnection class to handle all types of HTTP requests

The following class can be used as a single class that can handle `GET`, `POST`, `PUT`, `PATCH`, and other requests:

```
class APIResponseObject{
    int responseCode;
    String response;

    APIResponseObject(int responseCode,String response)
    {
         this.responseCode = responseCode;
         this.response = response;
    }
}

public class APIAccessTask extends AsyncTask<String,Void,APIResponseObject> {
    URL requestUrl;
    Context context;
    HttpURLConnection urlConnection;
    List<Pair<String,String>> postData, headerData;
    String method;
    int responseCode = HttpURLConnection.HTTP_OK;

    interface OnCompleteListener{
        void onComplete(APIResponseObject result);
    }

    public OnCompleteListener delegate = null;

    APIAccessTask(Context context, String requestUrl, String method, OnCompleteListener delegate){
        this.context = context;
        this.delegate = delegate;
        this.method = method;
        try {
            this.requestUrl = new URL(requestUrl);
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
    }

    APIAccessTask(Context context, String requestUrl, String method, List<Pair<String,String>>
postData, OnCompleteListener delegate){
        this(context, requestUrl, method, delegate);
        this.postData = postData;
    }

    APIAccessTask(Context context, String requestUrl, String method, List<Pair<String,String>>
postData,
            List<Pair<String,String>> headerData, OnCompleteListener delegate ){
        this(context, requestUrl,method,postData,delegate);
        this.headerData = headerData;
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
```

```
    protected APIResponseObject doInBackground(String... params) {
        Log.d("debug", "url = "+ requestUrl);
        try {
            urlConnection = (HttpURLConnection) requestUrl.openConnection();

            if(headerData != null) {
                for (Pair pair : headerData) {
                    urlConnection.setRequestProperty(pair.first.toString(),pair.second.toString());
                }
            }

            urlConnection.setDoInput(true);
            urlConnection.setChunkedStreamingMode(0);
            urlConnection.setRequestMethod(method);
            urlConnection.connect();

            StringBuilder sb = new StringBuilder();

            if(!(method.equals("GET"))) {
                OutputStream out = new BufferedOutputStream(urlConnection.getOutputStream());
                BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(out, "UTF-8"));
                writer.write(getPostDataString(postData));
                writer.flush();
                writer.close();
                out.close();
            }

            urlConnection.connect();
            responseCode = urlConnection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                InputStream in = new BufferedInputStream(urlConnection.getInputStream());
                BufferedReader reader = new BufferedReader(new InputStreamReader(in, "UTF-8"));
                String line;

                while ((line = reader.readLine()) != null) {
                    sb.append(line);
                }
            }

            return new APIResponseObject(responseCode, sb.toString());
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPostExecute(APIResponseObject result) {
        delegate.onComplete(result);
        super.onPostExecute(result);
    }

    private String getPostDataString(List<Pair<String, String>> params) throws
UnsupportedEncodingException {
        StringBuilder result = new StringBuilder();
        boolean first = true;
        for(Pair<String,String> pair : params){
            if (first)
                first = false;
            else
                result.append("&");
```

```
            result.append(URLEncoder.encode(pair.first,"UTF-8"));
            result.append("=");
            result.append(URLEncoder.encode(pair.second, "UTF-8"));
        }
        return result.toString();
    }
}
```

Usage

Use any of the given constructors of the class depending on whether you need to send `POST` data or any extra headers.

The `onComplete()` method will be called when the data fetching is complete. The data is returned as an object of the `APIResponseObject` class, which has a status code stating the HTTP status code of the request and a string containing the response. You can parse this response in your class, i.e. XML or JSON.

Call `execute()` on the object of the class to execute the request, as shown in the following example:

```
class MainClass {
    String url = "https://example.com./api/v1/ex";
    String method = "POST";
    List<Pair<String,String>> postData = new ArrayList<>();

    postData.add(new Pair<>("email","whatever");
    postData.add(new Pair<>("password", "whatever");

    new APIAccessTask(MainActivity.this, url, method, postData,
            new APIAccessTask.OnCompleteListener() {
        @Override
        public void onComplete(APIResponseObject result) {
            if (result.responseCode == HttpURLConnection.HTTP_OK) {
                String str = result.response;
                // Do your XML/JSON parsing here
            }
        }
    }).execute();
}
```

Section 74.6: Use HttpURLConnection for multipart/form-data

Create custom class for calling multipart/form-data HttpURLConnection request

MultipartUtility.java

```
public class MultipartUtility {
        private final String boundary;
        private static final String LINE_FEED = "\r\n";
        private HttpURLConnection httpConn;
        private String charset;
        private OutputStream outputStream;
        private PrintWriter writer;

        /**
         * This constructor initializes a new HTTP POST request with content type
         * is set to multipart/form-data
         *
         * @param requestURL
         * @param charset
```

```
     * @throws IOException
     */
    public MultipartUtility(String requestURL, String charset)
            throws IOException {
        this.charset = charset;

        // creates a unique boundary based on time stamp
        boundary = "===" + System.currentTimeMillis() + "===";
        URL url = new URL(requestURL);
        httpConn = (HttpURLConnection) url.openConnection();
        httpConn.setUseCaches(false);
        httpConn.setDoOutput(true);    // indicates POST method
        httpConn.setDoInput(true);
        httpConn.setRequestProperty("Content-Type",
                "multipart/form-data; boundary=" + boundary);
        outputStream = httpConn.getOutputStream();
        writer = new PrintWriter(new OutputStreamWriter(outputStream, charset),
                true);
    }

    /**
     * Adds a form field to the request
     *
     * @param name  field name
     * @param value field value
     */
    public void addFormField(String name, String value) {
        writer.append("--" + boundary).append(LINE_FEED);
        writer.append("Content-Disposition: form-data; name=\"" + name + "\"")
                .append(LINE_FEED);
        writer.append("Content-Type: text/plain; charset=" + charset).append(
                LINE_FEED);
        writer.append(LINE_FEED);
        writer.append(value).append(LINE_FEED);
        writer.flush();
    }

    /**
     * Adds a upload file section to the request
     *
     * @param fieldName  name attribute in <input type="file" name="..." />
     * @param uploadFile a File to be uploaded
     * @throws IOException
     */
    public void addFilePart(String fieldName, File uploadFile)
            throws IOException {
        String fileName = uploadFile.getName();
        writer.append("--" + boundary).append(LINE_FEED);
        writer.append(
                "Content-Disposition: form-data; name=\"" + fieldName
                        + "\"; filename=\"" + fileName + "\"")
                .append(LINE_FEED);
        writer.append(
                "Content-Type: "
                        + URLConnection.guessContentTypeFromName(fileName))
                .append(LINE_FEED);
        writer.append("Content-Transfer-Encoding: binary").append(LINE_FEED);
        writer.append(LINE_FEED);
        writer.flush();

        FileInputStream inputStream = new FileInputStream(uploadFile);
        byte[] buffer = new byte[4096];
```

```
            int bytesRead = -1;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }
            outputStream.flush();
            inputStream.close();
            writer.append(LINE_FEED);
            writer.flush();
        }

        /**
         * Adds a header field to the request.
         *
         * @param name  - name of the header field
         * @param value - value of the header field
         */
        public void addHeaderField(String name, String value) {
            writer.append(name + ": " + value).append(LINE_FEED);
            writer.flush();
        }

        /**
         * Completes the request and receives response from the server.
         *
         * @return a list of Strings as response in case the server returned
         * status OK, otherwise an exception is thrown.
         * @throws IOException
         */
        public List<String> finish() throws IOException {
            List<String> response = new ArrayList<String>();
            writer.append(LINE_FEED).flush();
            writer.append("--" + boundary + "--").append(LINE_FEED);
            writer.close();

            // checks server's status code first
            int status = httpConn.getResponseCode();
            if (status == HttpURLConnection.HTTP_OK) {
                BufferedReader reader = new BufferedReader(new InputStreamReader(
                        httpConn.getInputStream()));
                String line = null;
                while ((line = reader.readLine()) != null) {
                    response.add(line);
                }
                reader.close();
                httpConn.disconnect();
            } else {
                throw new IOException("Server returned non-OK status: " + status);
            }
            return response;
        }
    }
```

Use it (Async way)

```
    MultipartUtility multipart = new MultipartUtility(requestURL, charset);

    // In your case you are not adding form data so ignore this
                /*This is to add parameter values */
                for (int i = 0; i < myFormDataArray.size(); i++) {
                    multipart.addFormField(myFormDataArray.get(i).getParamName(),
                            myFormDataArray.get(i).getParamValue());
```

```
                }


//add your file here.
                /*This is to add file content*/
                for (int i = 0; i < myFileArray.size(); i++) {
                    multipart.addFilePart(myFileArray.getParamName(),
                            new File(myFileArray.getFileName()));
                }

                List<String> response = multipart.finish();
                Debug.e(TAG, "SERVER REPLIED:");
                for (String line : response) {
                    Debug.e(TAG, "Upload Files Response:::" + line);
// get your server response here.
                    responseString = line;
                }
```

Section 74.7: Upload (POST) file using HttpURLConnection

Quite often it's necessary to send/upload a file to a remote server, for example, an image, video, audio or a backup of the application database to a remote private server. Assuming the server is expecting a POST request with the content, here's a simple example of how to complete this task in Android.

File uploads are sent using `multipart/form-data` POST requests. It's very easy to implement:

```
URL url = new URL(postTarget);
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

String auth = "Bearer " + oauthToken;
connection.setRequestProperty("Authorization", basicAuth);

String boundary = UUID.randomUUID().toString();
connection.setRequestMethod("POST");
connection.setDoOutput(true);
connection.setRequestProperty("Content-Type", "multipart/form-data;boundary=" + boundary);

DataOutputStream request = new DataOutputStream(uc.getOutputStream());

request.writeBytes("--" + boundary + "\r\n");
request.writeBytes("Content-Disposition: form-data; name=\"description\"\r\n\r\n");
request.writeBytes(fileDescription + "\r\n");

request.writeBytes("--" + boundary + "\r\n");
request.writeBytes("Content-Disposition: form-data; name=\"file\"; filename=\"" + file.fileName +
"\"\r\n\r\n");
request.write(FileUtils.readFileToByteArray(file));
request.writeBytes("\r\n");

request.writeBytes("--" + boundary + "--\r\n");
request.flush();
int respCode = connection.getResponseCode();

switch(respCode) {
    case 200:
        //all went ok - read response
        ...
        break;
    case 301:
    case 302:
```

```
    case 307:
        //handle redirect - for example, re-post to the new location
        ...
        break;
    ...
    default:
        //do something sensible
}
```

Of course, exceptions will need to be caught or declared as being thrown. A couple points to note about this code:

1. `postTarget` is the destination URL of the POST; `oauthToken` is the authentication token; `fileDescription` is the description of the file, which is sent as the value of field `description`; `file` is the file to be sent - it's of type `java.io.File` - if you have the file path, you can use `new File(filePath)` instead.
2. It sets `Authorization` header for an oAuth auth
3. It uses Apache Common `FileUtil` to read the file into a byte array - if you already have the content of the file in a byte array or in some other way in memory, then there's no need to read it.

Chapter 75: Callback URL

Section 75.1: Callback URL example with Instagram OAuth

One of the use cases of *callback URLs* is OAuth. Let us do this with an Instagram Login: If the user enters their credentials and clicks the *Login* button, Instagram will validate the credentials and return an `access_token`. We need that `access_token` in our app.

For our app to be able to listen to such links, we need to add a callback URL to our `Activity`. We can do this by adding an **`<intent-filter/>`** to our `Activity`, which will react to that callback URL. Assume that our callback URL is `appSchema://appName.com`. Then you have to add the following lines to your desired `Activity` in the *Manifest.xml* file:

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE"/>
<data android:host="appName.com" android:scheme="appSchema"/>
```

Explanation of the lines above:

- **`<category`** `android:name="android.intent.category.BROWSABLE"` **`/>`** makes the target activity allow itself to be started by a web browser to display data referenced by a link.
- **`<data`** `android:host="appName.com" android:scheme="appSchema"` **`/>`** specifies our schema and host of our callback URL.
- All together, these lines will cause the specific `Activity` to be opened whenever the callback URL is called in a browser.

Now, in order to get the contents of the URL in your `Activity`, you need to override the `onResume()` method as follows:

```
@Override
public void onResume() {
    // The following line will return "appSchema://appName.com".
    String CALLBACK_URL = getResources().getString(R.string.insta_callback);
    Uri uri = getIntent().getData();
    if (uri != null && uri.toString().startsWith(CALLBACK_URL)) {
        String access_token = uri.getQueryParameter("access_token");
    }
    // Perform other operations here.
}
```

Now you have retrieved the `access_token` from Instagram, that is used in various API endpoints of Instagram.

Chapter 76: Snackbar

Parameter	Description
view	View: The view to find a parent from.
text	CharSequence: The text to show. Can be formatted text.
resId	int: The resource id of the string resource to use. Can be formatted text.
duration	int: How long to display the message. This can be LENGTH_SHORT, LENGTH_LONG or LENGTH_INDEFINITE

Section 76.1: Creating a simple Snackbar

Creating a `Snackbar` can be done as follows:

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG).show();
```

The `view` is used to find a suitable parent to use to display the `Snackbar`. Typically this would be a `CoordinatorLayout` that you've defined in your XML, which enables adding functionality such as swipe to dismiss and automatically moving of other widgets (e.g. `FloatingActionButton`). If there's no `CoordinatorLayout` then the window decor's content view is used.

Very often we also add an action to the `Snackbar`. A common use case would be an "Undo" action.

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG)
        .setAction("UNDO", new View.OnClickListener() {
            @Override
               public void onClick(View view) {
                   // put your logic here

               }
         })
        .show();
```

You can create a `Snackbar` and show it later:

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
snackbar.show();
```

If you want to change the color of the `Snackbar`'s text:

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
View view = snackbar .getView();
TextView textView = (TextView) view.findViewById(android.support.design.R.id.snackbar_text);
textView.setTextColor(Color.parseColor("#FF4500"));
snackbar.show();
```

By default `Snackbar` dismisses on it's right swipe.This example demonstrates how to dismiss the snackBar on it's left swipe.

Section 76.2: Custom Snack Bar

Function to customize snackbar

```
public static Snackbar makeText(Context context, String message, int duration) {
        Activity activity = (Activity) context;
```

```
        View layout;
        Snackbar snackbar = Snackbar
                .make(activity.findViewById(android.R.id.content), message, duration);
        layout = snackbar.getView();
        //setting background color
        layout.setBackgroundColor(context.getResources().getColor(R.color.orange));
        android.widget.TextView text = (android.widget.TextView)
layout.findViewById(android.support.design.R.id.snackbar_text);
        //setting font color
        text.setTextColor(context.getResources().getColor(R.color.white));
        Typeface font = null;
        //Setting font
        font = Typeface.createFromAsset(context.getAssets(), "DroidSansFallbackanmol256.ttf");
        text.setTypeface(font);
        return snackbar;

    }
```

Call the function from fragment or activity

```
 SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();
```

Section 76.3: Custom Snackbar (no need view)

Creating an Snackbar without the need pass view to Snackbar, all layout create in android in android.R.id.content.

```
public class CustomSnackBar {

    public static final int STATE_ERROR = 0;
    public static final int STATE_WARNING = 1;
    public static final int STATE_SUCCESS = 2;
    public static final int VIEW_PARENT = android.R.id.content;

    public CustomSnackBar(View view, String message, int actionType) {
        super();

        Snackbar snackbar = Snackbar.make(view, message, Snackbar.LENGTH_LONG);
        View sbView = snackbar.getView();
        TextView textView = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_text);
        textView.setTextColor(Color.parseColor("#ffffff"));
        textView.setTextSize(TypedValue.COMPLEX_UNIT_SP, 14);
        textView.setGravity(View.TEXT_ALIGNMENT_CENTER);
        textView.setLayoutDirection(View.LAYOUT_DIRECTION_RTL);

        switch (actionType) {
            case STATE_ERROR:
                snackbar.getView().setBackgroundColor(Color.parseColor("#F12B2B"));
                break;
            case STATE_WARNING:
                snackbar.getView().setBackgroundColor(Color.parseColor("#000000"));
                break;
            case STATE_SUCCESS:
                snackbar.getView().setBackgroundColor(Color.parseColor("#7ED321"));
                break;
        }
        snackbar.show();
    }
}
```

for call class

new CustomSnackBar(findViewById(CustomSnackBar.VIEW_PARENT),"message", CustomSnackBar.STATE_ERROR);

Section 76.4: Snackbar with Callback

You can use Snackbar.Callback to listen if the snackbar was dismissed by user or timeout.

```
Snackbar.make(getView(), "Hi snackbar!", Snackbar.LENGTH_LONG).setCallback( new Snackbar.Callback()
{
                @Override
                public void onDismissed(Snackbar snackbar, int event) {
                    switch(event) {
                        case Snackbar.Callback.DISMISS_EVENT_ACTION:
                            Toast.makeText(getActivity(), "Clicked the action",
Toast.LENGTH_LONG).show();
                            break;
                        case Snackbar.Callback.DISMISS_EVENT_TIMEOUT:
                            Toast.makeText(getActivity(), "Time out", Toast.LENGTH_LONG).show();
                            break;
                    }
                }

                @Override
                public void onShown(Snackbar snackbar) {
                    Toast.makeText(getActivity(), "This is my annoying step-brother",
Toast.LENGTH_LONG).show();
                }
            }).setAction("Go!", new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                }
            }).show();
```

Section 76.5: Snackbar vs Toasts: Which one should I use?

Toasts are generally used when we want to display an information to the user regarding some action that has successfully (or not) happened and this action does not require the user to take any other action. Like when a message has been sent, for example:

```
Toast.makeText(this, "Message Sent!", Toast.LENGTH_SHORT).show();
```

Snackbars are also used to display an information. But this time, we can give the user an opportunity to take an action. For example, let's say the user deleted a picture by mistake and he wants to get it back. We can provide a Snackbar with the "Undo" action. Like this:

```
Snackbar.make(getCurrentFocus(), "Picture Deleted", Snackbar.LENGTH_SHORT)
        .setAction("Undo", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Return his picture
            }
        })
        .show();
```

Conclusion: Toasts are used when we don't need user interaction. Snackbars are used to allow users to take

another action or undo a previous one.

Section 76.6: Custom Snackbar

This example shows a white Snackbar with custom Undo icon.

```
Snackbar customBar = Snackbar.make(view , "Text to be displayed", Snackbar.LENGTH_LONG);
customBar.setAction("UNDO", new View.OnClickListener() {
    @Override
    public void onClick(View view) {
    //Put the logic for undo button here

    }
});

View sbView = customBar.getView();
//Changing background to White
sbView.setBackgroundColor(Color.WHITE));

TextView snackText = (TextView) sbView.findViewById(android.support.design.R.id.snackbar_text);
if (snackText!=null) {
    //Changing text color to Black
   snackText.setTextColor(Color.BLACK);
}

TextView actionText = (TextView) sbView.findViewById(android.support.design.R.id.snackbar_action);
if (actionText!=null) {
    // Setting custom Undo icon
    actionText.setCompoundDrawablesRelativeWithIntrinsicBounds(R.drawable.custom_undo, 0, 0, 0);
}
customBar.show();
```

Chapter 77: Widgets

Section 77.1: Manifest Declaration -

Declare the `AppWidgetProvider` class in your application's `AndroidManifest.xml` file. For example:

```
<receiver android:name="ExampleAppWidgetProvider" >
<intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
</intent-filter>
<meta-data android:name="android.appwidget.provider"
          android:resource="@xml/example_appwidget_info" />
</receiver>
```

Section 77.2: Metadata

Add the AppWidgetProviderInfo metadata in `res/xml`:

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
   android:minWidth="40dp"
   android:minHeight="40dp"
   android:updatePeriodMillis="86400000"
   android:previewImage="@drawable/preview"
   android:initialLayout="@layout/example_appwidget"
   android:configure="com.example.android.ExampleAppWidgetConfigure"
   android:resizeMode="horizontal|vertical"
   android:widgetCategory="home_screen">
</appwidget-provider>
```

Section 77.3: AppWidgetProvider Class

The most important `AppWidgetProvider` callback is **`onUpdate()`**. It is called every time an appwidget is added.

```
public class ExampleAppWidgetProvider extends AppWidgetProvider {

    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
        final int N = appWidgetIds.length;

        // Perform this loop procedure for each App Widget that belongs to this provider
        for (int i=0; i<N; i++) {
            int appWidgetId = appWidgetIds[i];

            // Create an Intent to launch ExampleActivity
            Intent intent = new Intent(context, ExampleActivity.class);
            PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent, 0);

            // Get the layout for the App Widget and attach an on-click listener
            // to the button
            RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.appwidget_provider_layout);
            views.setOnClickPendingIntent(R.id.button, pendingIntent);

            // Tell the AppWidgetManager to perform an update on the current app widget
            appWidgetManager.updateAppWidget(appWidgetId, views);
        }
    }
}
```

`onAppWidgetOptionsChanged()` is called when the widget is placed or resized.

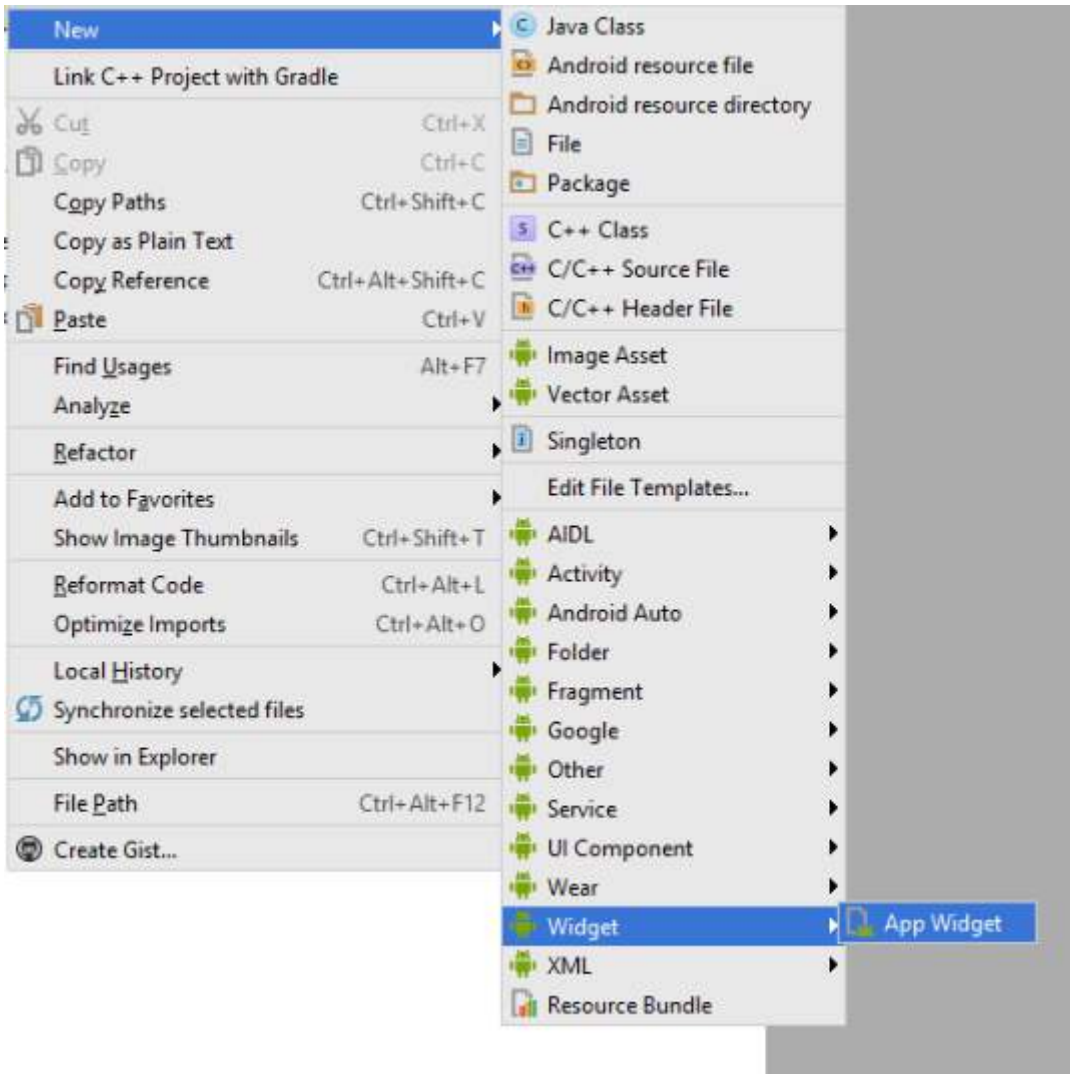
`onDeleted(Context, int[])` is called when the widget is deleted.

Section 77.4: Create/Integrate Basic Widget using Android Studio

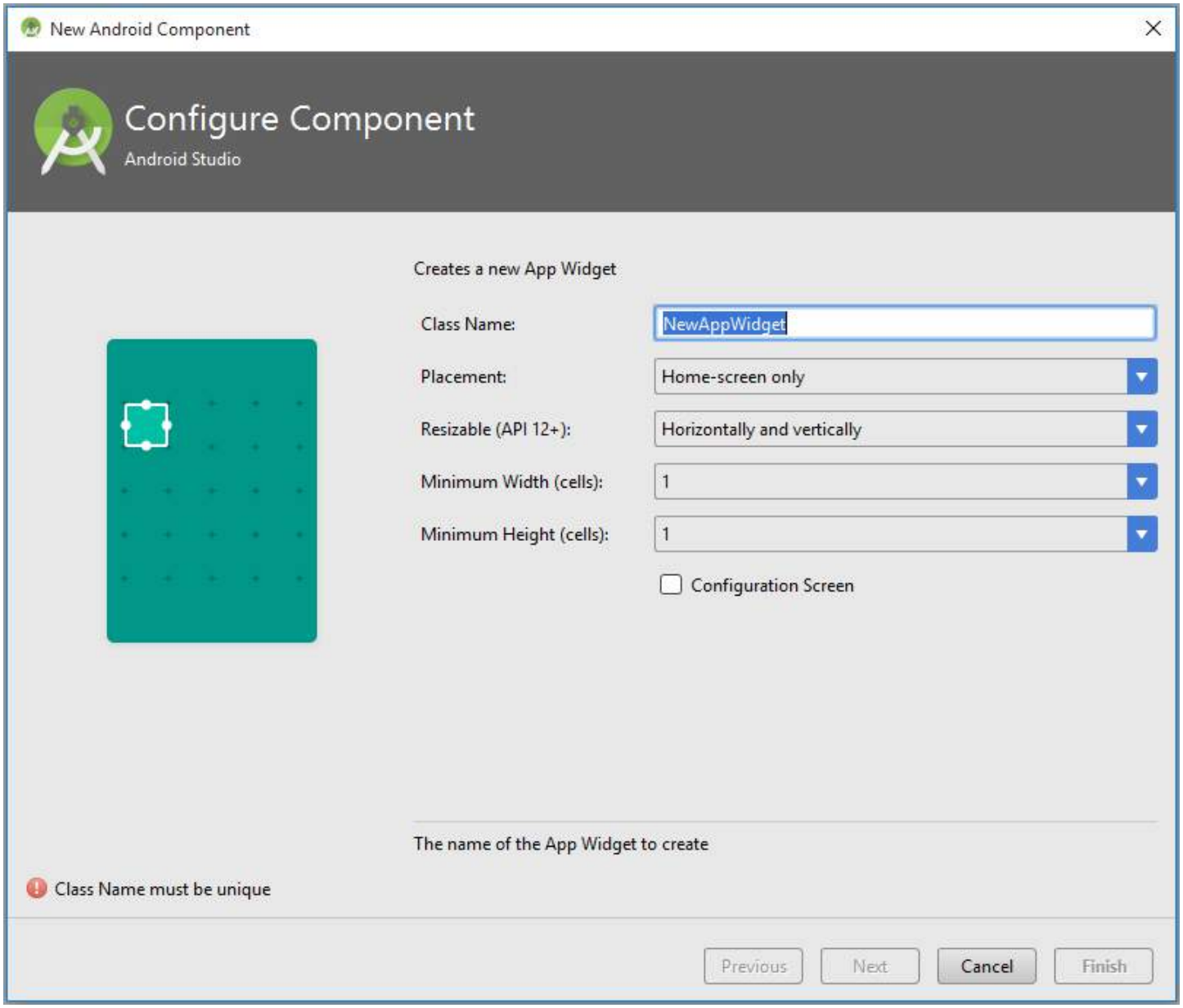
Latest Android Studio will create & integrate a Basic Widget to your Application in 2 steps.

Right on your Application ==> New ==> Widget ==> App Widget

.



It will show a Screen like below & fill the fields



Its Done.

It will **create & integrate a basic HelloWorld Widget**(Including Layout File , Meta Data File , Declaration in Manifest File etc.) to your Application.

Section 77.5: Two widgets with different layouts declaration

1. Declare two receivers in a manifest file:

```
<receiver
   android:name=".UVMateWidget"
   android:label="UVMate Widget 1x1">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
       android:name="android.appwidget.provider"
       android:resource="@xml/widget_1x1" />
</receiver>
<receiver
   android:name=".UVMateWidget2x2"
   android:label="UVMate Widget 2x2">
```

```
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
       android:name="android.appwidget.provider"
       android:resource="@xml/widget_2x2" />
</receiver>
```

2. Create two layouts
    - `@xml/widget_1x1`
    - `@xml/widget_2x2`
3. Declare the subclass `UVMateWidget2x2` from the `UVMateWidget` class with extended behavior:

```
package au.com.aershov.uvmate;

import android.content.Context;
import android.widget.RemoteViews;

public class UVMateWidget2x2 extends UVMateWidget {

    public RemoteViews getRemoteViews(Context context, int minWidth,
                                      int minHeight) {

        mUVMateHelper.saveWidgetSize(mContext.getString(R.string.app_ws_2x2));
        return new RemoteViews(context.getPackageName(), R.layout.widget_2x2);
    }
}
```

Chapter 78: Toast

Parameter	Details
context	The context to display your Toast in. `this` is commonly used in an Activity and `getActivity()` is commonly used in a Fragment
text	A CharSequence that specifies what text will be shown in the Toast. Any object that implements CharSequence can be used, including a String
resId	A resource ID that can be used to provide a resource String to display in the Toast
duration	Integer flag representing how long the Toast will show. Options are `Toast.LENGTH_SHORT` and `Toast.LENGTH_LONG`
gravity	Integer specifying the position, or "gravity" of the Toast. See options here
xOffset	Specifies the horizontal offset for the Toast position
yOffset	Specifies the vertical offset for the Toast position

A Toast provides simple feedback about an operation in a small popup and automatically disappears after a timeout. It only fills the amount of space required for the message and the current activity remains visible and interactive.

Section 78.1: Creating a custom Toast

If you don't want to use the default Toast view, you can provide your own using the `setView(View)` method on a `Toast` object.

First, create the XML layout you would like to use in your Toast.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/toast_layout_root"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="8dp"
   android:background="#111">

    <TextView android:id="@+id/title"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textColor="#FFF"/>

    <TextView android:id="@+id/description"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textColor="#FFF"/>

</LinearLayout>
```

Then, when creating your Toast, inflate your custom View from XML, and call `setView`

```
// Inflate the custom view from XML
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast_layout,
                               (ViewGroup) findViewById(R.id.toast_layout_root));

// Set the title and description TextViews from our custom layout
TextView title = (TextView) layout.findViewById(R.id.title);
title.setText("Toast Title");
```

```
TextView description = (TextView) layout.findViewById(R.id.description);
description.setText("Toast Description");

// Create and show the Toast object

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

Section 78.2: Set position of a Toast

A standard toast notification appears at the bottom of the screen aligned in horizontal centre. You can change this position with the `setGravity(int, int, int)`. This accepts three parameters: a Gravity constant, an x-position offset, and a y-position offset.

For example, if you decide that the toast should appear in the top-left corner, you can set the gravity like this:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Section 78.3: Showing a Toast Message

In Android, a Toast is a simple UI element that can be used to give contextual feedback to a user.

To display a simple Toast message, we can do the following.

```
// Declare the parameters to use for the Toast

Context context = getApplicationContext();
// in an Activity, you may also use "this"
// in a fragment, you can use getActivity()

CharSequence message = "I'm an Android Toast!";
int duration = Toast.LENGTH_LONG; // Toast.LENGTH_SHORT is the other option

// Create the Toast object, and show it!
Toast myToast = Toast.makeText(context, message, duration);
myToast.show();
```

Or, to show a Toast inline, without holding on to the Toast object you can:

```
Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
```

IMPORTANT: Make sure that the `show()` method is called from the UI thread. If you're trying to show a `Toast` from a different thread you can e.g. use `runOnUiThread` method of an `Activity`.

Failing to do so, meaning trying to modify the UI by creating a Toast, will throw a `RuntimeException` which will look like this:

```
java.lang.RuntimeException: Can't create handler inside thread that has not called Looper.prepare()
```

The simplest way of handling this exception is just by using runOnUiThread: syntax is shown below.

```
   runOnUiThread(new Runnable() {
        @Override
```

```
        public void run() {
            // Your code here
        }
    });
```

Section 78.4: Show Toast Message Above Soft Keyboard

By default, Android will display Toast messages at the bottom of the screen even if the keyboard is showing. This will show a Toast message just above the keyboard.

```
public void showMessage(final String message, final int length) {
    View root = findViewById(android.R.id.content);
    Toast toast = Toast.makeText(this, message, length);
    int yOffset = Math.max(0, root.getHeight() - toast.getYOffset());
    toast.setGravity(Gravity.TOP | Gravity.CENTER_HORIZONTAL, 0, yOffset);
    toast.show();
}
```

Section 78.5: Thread safe way of displaying Toast (Application Wide)

```
public class MainApplication extends Application {

    private static Context context; //application context

    private Handler mainThreadHandler;
    private Toast toast;

    public Handler getMainThreadHandler() {
        if (mainThreadHandler == null) {
            mainThreadHandler = new Handler(Looper.getMainLooper());
        }
        return mainThreadHandler;
    }

    @Override public void onCreate() {
        super.onCreate();
        context = this;
    }

    public static MainApplication getApp(){
        return (MainApplication) context;
    }

    /**
     * Thread safe way of displaying toast.
     * @param message
     * @param duration
     */
    public void showToast(final String message, final int duration) {
        getMainThreadHandler().post(new Runnable() {
            @Override
            public void run() {
                if (!TextUtils.isEmpty(message)) {
                    if (toast != null) {
                        toast.cancel(); //dismiss current toast if visible
                        toast.setText(message);
                    } else {
                        toast = Toast.makeText(App.this, message, duration);
```

```
                    }
                    toast.show();
                }
            }
        });
    }
```

Remember to add `MainApplication` in `manifest`.

Now call it from any thread to display a toast message.

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

Section 78.6: Thread safe way of displaying a Toast Message (For AsyncTask)

If you don't want to extend Application and keep your toast messages thread safe, make sure you show them in the post execute section of your AsyncTasks.

```
public class MyAsyncTask extends AsyncTask <Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... params) {
        // Do your background work here
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        // Show toast messages here
        Toast.makeText(context, "Ding! Your Toast is ready.",   Toast.LENGTH_SHORT).show();
    }

}
```

Chapter 79: Create Singleton Class for Toast Message

Parameter	details
context	Relevant context which needs to display your toast message. If you use this in the activity pass "this" keyword or If you use in fragement pass as "getActivity()".
view	Create a custom view and pass that view object to this.
gravity	Pass the gravity position of the toaster. All the position has added under the Gravity class as the static variables . The Most common positions are Gravity.TOP, Gravity.BOTTOM, Gravity.LEFT, Gravity.RIGHT.
xOffset	Horizontal offset of the toast message.
yOffset	Vertical offset of the toast message.
duration	Duration of the toast show. We can set either Toast.LENGTH_SHORT or Toast.LENGTH_LONG

Toast messages are the most simple way of providing feedback to the user. By default, Android provide gray color message toast where we can set the message and the duration of the message. If we need to create more customizable and reusable toast message, we can implement it by ourselves with the use of a custom layout. More importantly when we are implementing it, the use of Singelton design pattern will make it easy for maintaining and development of the custom toast message class.

Section 79.1: Create own singleton class for toast massages

Here is how to create your own singleton class for toast messages, If your application need to show success, warning and the danger messages for different use cases you can use this class after you have modified it to your own specifications.

```
    public class ToastGenerate {
        private static ToastGenerate ourInstance;

        public ToastGenerate (Context context) {
            this.context = context;
                                                }
        public static ToastGenerate getInstance(Context context) {
            if (ourInstance == null)
                ourInstance = new ToastGenerate(context);
            return ourInstance;
        }

       //pass message and message type to this method
        public void createToastMessage(String message,int type){

//inflate the custom layout
            LayoutInflater layoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

            LinearLayout toastLayout = (LinearLayout)
layoutInflater.inflate(R.layout.layout_custome_toast,null);
            TextView toastShowMessage = (TextView)
toastLayout.findViewById(R.id.textCustomToastTopic);

            switch (type){
                case 0:
                    //if the message type is 0 fail toaster method will call
                    createFailToast(toastLayout,toastShowMessage,message);
                    break;
                case 1:
```

```
                    //if the message type is 1 success toaster method will call
                    createSuccessToast(toastLayout,toastShowMessage,message);
                    break;

                case 2:
                    createWarningToast( toastLayout, toastShowMessage, message);
                    //if the message type is 2 warning toaster method will call
                    break;
                default:
                    createFailToast(toastLayout,toastShowMessage,message);

            }
        }

    //Failure toast message method
        private final void createFailToast(LinearLayout toastLayout,TextView toastMessage,String
message){

toastLayout.setBackgroundColor(context.getResources().getColor(R.color.button_alert_normal));
            toastMessage.setText(message);
            toastMessage.setTextColor(context.getResources().getColor(R.color.white));
            showToast(context,toastLayout);
        }

        //warning toast message method
        private final void createWarningToast( LinearLayout toastLayout, TextView toastMessage,
String message) {
            toastLayout.setBackgroundColor(context.getResources().getColor(R.color.warning_toast));
            toastMessage.setText(message);
            toastMessage.setTextColor(context.getResources().getColor(R.color.white));
            showToast(context, toastLayout);
        }
    //success toast message method
        private final void createSuccessToast(LinearLayout toastLayout,TextView toastMessage,String
message){
            toastLayout.setBackgroundColor(context.getResources().getColor(R.color.success_toast));

            toastMessage.setText(message);
            toastMessage.setTextColor(context.getResources().getColor(R.color.white));
            showToast(context,toastLayout);
        }

        private void showToast(View view){
            Toast toast = new Toast(context);
            toast.setGravity(Gravity.TOP,0,0); // show message in the top of the device
            toast.setDuration(Toast.LENGTH_SHORT);
            toast.setView(view);
            toast.show();
        }
    }
```

Chapter 80: Interfaces

Section 80.1: Custom Listener

Define interface

```
//In this interface, you can define messages, which will be send to owner.
public interface MyCustomListener {
    //In this case we have two messages,
    //the first that is sent when the process is successful.
    void onSuccess(List<Bitmap> bitmapList);
    //And The second message, when the process will fail.
    void onFailure(String error);
}
```

Create listener

In the next step we need to define an instance variable in the object that will send callback via `MyCustomListener`. And add setter for our listener.

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }
}
```

Implement listener

Now, in other class, we can create instance of `SampleClassB`.

```
public class SomeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
    }
}
```

next we can set our listener, to `sampleClass`, in two ways:

by implements `MyCustomListener` in our class:

```
public class SomeActivity extends Activity implements MyCustomListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
        sampleClass.setMyCustomListener(this);
    }

    @Override
    public void onSuccess(List<Bitmap> bitmapList) {

    }

    @Override
    public void onFailure(String error) {

    }
```

```
}
```

or just instantiate an anonymous inner class:

```
public class SomeActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
        sampleClass.setMyCustomListener(new MyCustomListener() {

            @Override
            public void onSuccess(List<Bitmap> bitmapList) {

            }

            @Override
            public void onFailure(String error) {

            }
        });
    }
}
```

Trigger listener

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }

    public void doSomething() {
        fetchImages();
    }

    private void fetchImages() {
        AsyncImagefetch imageFetch = new AsyncImageFetch();
        imageFetch.start(new Response<Bitmap>() {
            @Override
            public void onDone(List<Bitmap> bitmapList, Exception e) {
                //do some stuff if needed

                //check if listener is set or not.
                if(listener == null)
                    return;
                //Fire proper event. bitmapList or error message will be sent to
                //class which set listener.
                if(e == null)
                    listener.onSuccess(bitmapList);
                else
                    listener.onFailure(e.getMessage());
            }
        });
    }
}
```

Section 80.2: Basic Listener

The "listener" or "observer" pattern is the most common strategy for creating asynchronous callbacks in Android

development.

```
public class MyCustomObject {

  //1 - Define the interface
  public interface MyCustomObjectListener {
      public void onAction(String action);
  }

  //2 - Declare your listener object
  private MyCustomObjectListener listener;

  // and initialize it in the costructor
  public MyCustomObject() {
    this.listener = null;
 }

 //3 - Create your listener setter
 public void setCustomObjectListener(MyCustomObjectListener listener) {
    this.listener = listener;
 }

 // 4 - Trigger listener event
 public void makeSomething(){
    if (this.listener != null){
       listener.onAction("hello!");
 }
}
```

Now on your Activity:

```
public class MyActivity extends Activity {
   public final String TAG = "MyActivity";

  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main_activity);


      MyCustomObject mObj = new MyCustomObject();

      //5 - Implement listener callback
      mObj.setCustomObjectListener(new MyCustomObjectListener() {
        @Override
          public void onAction(String action) {
              Log.d(TAG, "Value: "+action);
          }
      });
   }
}
```

Chapter 81: Animators

Section 81.1: TransitionDrawable animation

This example displays a transaction for an image view with only two images.(can use more images as well one after the other for the first and second layer positions after each transaction as a loop)

- add a image array to `res/values/arrays.xml`

```
<resources>

    <array
       name="splash_images">
        <item>@drawable/spash_imge_first</item>
        <item>@drawable/spash_img_second</item>
    </array>

</resources>
```

```
    private Drawable[] backgroundsDrawableArrayForTransition;
    private TransitionDrawable transitionDrawable;

private void backgroundAnimTransAction() {

        // set res image array
        Resources resources = getResources();
        TypedArray icons = resources.obtainTypedArray(R.array.splash_images);

        @SuppressWarnings("ResourceType")
        Drawable drawable = icons.getDrawable(0);    // ending image

        @SuppressWarnings("ResourceType")
        Drawable drawableTwo = icons.getDrawable(1);   // starting image


        backgroundsDrawableArrayForTransition = new Drawable[2];
        backgroundsDrawableArrayForTransition[0] = drawable;
        backgroundsDrawableArrayForTransition[1] = drawableTwo;
        transitionDrawable = new TransitionDrawable(backgroundsDrawableArrayForTransition);

        // your image view here - backgroundImageView
        backgroundImageView.setImageDrawable(transitionDrawable);
        transitionDrawable.startTransition(4000);

        transitionDrawable.setCrossFadeEnabled(false); // call public methods


    }
```

Section 81.2: Fade in/out animation

In order to get a view to slowly fade in or out of view, use an `ObjectAnimator`. As seen in the code below, set a duration using `.setDuration(millis)` where the `millis` parameter is the duration (in milliseconds) of the animation. In the below code, the views will fade in / out over 500 milliseconds, or 1/2 second. To start the `ObjectAnimator`'s animation, call `.start()`. Once the animation is complete, `onAnimationEnd(Animator animation)` is called. Here is a good place to set your view's visibility to `View.GONE` or `View.VISIBLE`.

```
import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.ValueAnimator;

void fadeOutAnimation(View viewToFadeOut) {
    ObjectAnimator fadeOut = ObjectAnimator.ofFloat(viewToFadeOut, "alpha", 1f, 0f);

    fadeOut.setDuration(500);
    fadeOut.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationEnd(Animator animation) {
            // We wanna set the view to GONE, after it's fade out. so it actually disappear from the
layout & don't take up space.
            viewToFadeOut.setVisibility(View.GONE);
        }
    });

    fadeOut.start();
}

void fadeInAnimation(View viewToFadeIn) {
    ObjectAnimator fadeIn = ObjectAnimator.ofFloat(viewToFadeIn, "alpha", 0f, 1f);
    fadeIn.setDuration(500);

    fadeIn.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationStar(Animator animation) {
            // We wanna set the view to VISIBLE, but with alpha 0. So it appear invisible in the
layout.
            viewToFadeIn.setVisibility(View.VISIBLE);
            viewToFadeIn.setAlpha(0);
        }
    });

    fadeIn.start();
}
```

Section 81.3: ValueAnimator

`ValueAnimator` introduces a simple way to animate a value (of a particular type, e.g. **`int`**, **`float`**, etc.).

The usual way of using it is:

1. Create a `ValueAnimator` that will animate a value from `min` to `max`
2. Add an `UpdateListener` in which you will use the calculated animated value (which you can obtain with `getAnimatedValue()`)

There are two ways you can create the `ValueAnimator`:

*(the example code animates a **`float`** from `20f` to `40f` in `250ms`)*

1. From `xml` (put it in the `/res/animator/`):

```
<animator xmlns:android="http://schemas.android.com/apk/res/android"
   android:duration="250"
   android:valueFrom="20"
   android:valueTo="40"
   android:valueType="floatType"/>
```

```
ValueAnimator animator = (ValueAnimator) AnimatorInflater.loadAnimator(context,
```

```
        R.animator.example_animator);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // ... use the anim.getAnimatedValue()
    }
});
// set all the other animation-related stuff you want (interpolator etc.)
animator.start();
```

2. From the code:

```
ValueAnimator animator = ValueAnimator.ofFloat(20f, 40f);
animator.setDuration(250);
animator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
    @Override
    public void onAnimationUpdate(ValueAnimator anim) {
        // use the anim.getAnimatedValue()
    }
});
// set all the other animation-related stuff you want (interpolator etc.)
animator.start();
```

Section 81.4: Expand and Collapse animation of View

```
public class ViewAnimationUtils {

        public static void expand(final View v) {
            v.measure(LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT);
            final int targtetHeight = v.getMeasuredHeight();

            v.getLayoutParams().height = 0;
            v.setVisibility(View.VISIBLE);
            Animation a = new Animation()
            {
                @Override
                protected void applyTransformation(float interpolatedTime, Transformation t) {
                    v.getLayoutParams().height = interpolatedTime == 1
                            ? LayoutParams.WRAP_CONTENT
                            : (int)(targtetHeight * interpolatedTime);
                    v.requestLayout();
                }

                @Override
                public boolean willChangeBounds() {
                    return true;
                }
            };

            a.setDuration((int)(targtetHeight /
v.getContext().getResources().getDisplayMetrics().density));
            v.startAnimation(a);
        }

        public static void collapse(final View v) {
            final int initialHeight = v.getMeasuredHeight();

            Animation a = new Animation()
            {
                @Override
                protected void applyTransformation(float interpolatedTime, Transformation t) {
```

```
                    if(interpolatedTime == 1){
                        v.setVisibility(View.GONE);
                    }else{
                        v.getLayoutParams().height = initialHeight - (int)(initialHeight *
interpolatedTime);
                        v.requestLayout();
                    }
                }

                @Override
                public boolean willChangeBounds() {
                    return true;
                }
            };

            a.setDuration((int)(initialHeight /
v.getContext().getResources().getDisplayMetrics().density));
            v.startAnimation(a);
        }
    }
```

Section 81.5: ObjectAnimator

`ObjectAnimator` is a subclass of `ValueAnimator` with the added ability to set the calculated value to the property of a `target` `View`.

Just like in the `ValueAnimator`, there are two ways you can create the `ObjectAnimator`:

(the example code animates an `alpha` of a `View` from `0.4f` to `0.2f` in `250ms`)

1. From `xml` (put it in the `/res/animator`)

```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
                android:duration="250"
                android:propertyName="alpha"
                android:valueFrom="0.4"
                android:valueTo="0.2"
                android:valueType="floatType"/>
```

```
ObjectAnimator animator = (ObjectAnimator) AnimatorInflater.loadAnimator(context,
        R.animator.example_animator);
animator.setTarget(exampleView);
// set all the animation-related stuff you want (interpolator etc.)
animator.start();
```

2. From code:

```
ObjectAnimator animator = ObjectAnimator.ofFloat(exampleView, View.ALPHA, 0.4f, 0.2f);
animator.setDuration(250);
// set all the animation-related stuff you want (interpolator etc.)
animator.start();
```

Section 81.6: ViewPropertyAnimator

`ViewPropertyAnimator` is a simplified and optimized way to animate properties of a `View`.

Every single `View` has a `ViewPropertyAnimator` object available through the `animate()` method. You can use that to animate multiple properties at once with a simple call. Every single method of a `ViewPropertyAnimator` specifies the **target** value of a specific parameter that the `ViewPropertyAnimator` should animate to.

```
View exampleView = ...;
exampleView.animate()
        .alpha(0.6f)
        .translationY(200)
        .translationXBy(10)
        .scaleX(1.5f)
        .setDuration(250)
        .setInterpolator(new FastOutLinearInInterpolator());
```

Note: Calling `start()` on a `ViewPropertyAnimator` object is **NOT** mandatory. If you don't do that you're just letting the platform to handle the starting of the animation in the appropriate time (next animation handling pass). If you actually do that (call `start()`) you're making sure the animation is started immediately.

Section 81.7: Shake animation of an ImageView

Under res folder, create a new folder called "anim" to store your animation resources and put this on that folder.

shakeanimation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
   android:duration="100"
   android:fromDegrees="-15"
   android:pivotX="50%"
   android:pivotY="50%"
   android:repeatCount="infinite"
   android:repeatMode="reverse"
   android:toDegrees="15" />
```

Create a blank activity called Landing

activity_landing.xml

```
  <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/imgBell"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@mipmap/ic_notifications_white_48dp"/>

    </RelativeLayout>
```

And the method for animate the imageview on Landing.java

```
Context mContext;


 @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            mContext=this;
                setContentView(R.layout.activity_landing);

                AnimateBell();
```

```
        }

public void AnimateBell() {
        Animation shake = AnimationUtils.loadAnimation(mContext, R.anim.shakeanimation);
        ImageView imgBell= (ImageView) findViewById(R.id.imgBell);
        imgBell.setImageResource(R.mipmap.ic_notifications_active_white_48dp);
        imgBell.setAnimation(shake);
    }
```

Chapter 82: Location

Android Location APIs are used in a wide variety of apps for different purposes such as finding user location, notifying when a user has left a general area (Geofencing), and help interpret user activity (walking, running, driving, etc).

However, Android Location APIs are not the only means of acquiring user location. The following will give examples of how to use Android's `LocationManager` and other common location libraries.

Section 82.1: Fused location API

Example Using Activity w/ LocationRequest

```
/*
 * This example is useful if you only want to receive updates in this
 * activity only, and have no use for location anywhere else.
 */
public class LocationActivity extends AppCompatActivity implements
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
LocationListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();

        mLocationRequest = new LocationRequest()
                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) //GPS quality location points
                .setInterval(2000) //At least once every 2 seconds
                .setFastestInterval(1000); //At most once a second
    }

    @Override
    protected void onStart(){
        super.onStart();
        mGoogleApiClient.connect();
    }

    @Override
    protected void onResume(){
        super.onResume();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
            }
        }
    }
```

```
    @Override
    protected void onPause(){
        super.onPause();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
            }
        }
    }

    @Override
    protected void onStop(){
        super.onStop();
        mGoogleApiClient.disconnect();
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    }

    @Override
    public void onLocationChanged(Location location) {
        //Handle your location update code here
    }
}
```

Example Using Service w/ PendingIntent and BroadcastReceiver

ExampleActivity

Recommended reading: LocalBroadcastManager

```
/*
 * This example is useful if you have many different classes that should be
 * receiving location updates, but want more granular control over which ones
 * listen to the updates.
 *
 * For example, this activity will stop getting updates when it is not visible, but a database
 * class with a registered local receiver will continue to receive updates, until "stopUpdates()" is
called here.
 *
 */
public class ExampleActivity extends AppCompatActivity {

    private InternalLocationReceiver mInternalLocationReceiver;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);

        //Create internal receiver object in this method only.
        mInternalLocationReceiver = new InternalLocationReceiver(this);
    }

    @Override
    protected void onResume(){
        super.onResume();

        //Register to receive updates in activity only when activity is visible
        LocalBroadcastManager.getInstance(this).registerReceiver(mInternalLocationReceiver, new
IntentFilter("googleLocation"));
    }

    @Override
    protected void onPause(){
        super.onPause();

        //Unregister to stop receiving updates in activity when it is not visible.
        //NOTE: You will still receive updates even if this activity is killed.
        LocalBroadcastManager.getInstance(this).unregisterReceiver(mInternalLocationReceiver);
    }

    //Helper method to get updates
    private void requestUpdates(){
        startService(new Intent(this, LocationService.class).putExtra("request", true));
    }

    //Helper method to stop updates
    private void stopUpdates(){
        startService(new Intent(this, LocationService.class).putExtra("remove", true));
    }

    /*
     * Internal receiver used to get location updates for this activity.
     *
     * This receiver and any receiver registered with LocalBroadcastManager does
     * not need to be registered in the Manifest.
     *
     */
    private static class InternalLocationReceiver extends BroadcastReceiver{

        private ExampleActivity mActivity;

        InternalLocationReceiver(ExampleActivity activity){
            mActivity = activity;
        }

        @Override
        public void onReceive(Context context, Intent intent) {
            final ExampleActivity activity = mActivity;
            if(activity != null) {
                LocationResult result = intent.getParcelableExtra("result");
                //Handle location update here
            }
        }
    }
}
```

LocationService

NOTE: Don't forget to register this service in the Manifest!

```
public class LocationService extends Service implements
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    public void onCreate(){
        super.onCreate();
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();

        mLocationRequest = new LocationRequest()
                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) //GPS quality location points
                .setInterval(2000) //At least once every 2 seconds
                .setFastestInterval(1000); //At most once a second
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId){
        super.onStartCommand(intent, flags, startId);
        //Permission check for Android 6.0+
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            if (intent.getBooleanExtra("request", false)) {
                if (mGoogleApiClient.isConnected()) {
                    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, getPendingIntent());
                } else {
                    mGoogleApiClient.connect();
                }
            }
            else if(intent.getBooleanExtra("remove", false)){
                stopSelf();
            }
        }

        return START_STICKY;
    }

    @Override
    public void onDestroy(){
        super.onDestroy();
        if(mGoogleApiClient.isConnected()){
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
getPendingIntent());
            mGoogleApiClient.disconnect();
        }
    }

    private PendingIntent getPendingIntent(){

        //Example for IntentService
        //return PendingIntent.getService(this, 0, new Intent(this,
**YOUR_INTENT_SERVICE_CLASS_HERE**), PendingIntent.FLAG_UPDATE_CURRENT);
```

```
        //Example for BroadcastReceiver
        return PendingIntent.getBroadcast(this, 0, new Intent(this, LocationReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        //Permission check for Android 6.0+
        if(ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, getPendingIntent());
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

LocationReceiver

NOTE: Don't forget to register this receiver in the Manifest!

```
public class LocationReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if(LocationResult.hasResult(intent)){
            LocationResult locationResult = LocationResult.extractResult(intent);
            LocalBroadcastManager.getInstance(context).sendBroadcast(new
Intent("googleLocation").putExtra("result", locationResult));
        }
    }
}
```

Section 82.2: Get Address From Location using Geocoder

After you got the `Location` object from `FusedAPI`, you can easily acquire `Address` information from that object.

```
private Address getCountryInfo(Location location) {
    Address address = null;
    Geocoder geocoder = new Geocoder(getActivity(), Locale.getDefault());
    String errorMessage;
    List<Address> addresses = null;
    try {
        addresses = geocoder.getFromLocation(
```

```
                location.getLatitude(),
                location.getLongitude(),
                // In this sample, get just a single address.
                1);
    } catch (IOException ioException) {
        // Catch network or other I/O problems.
        errorMessage = "IOException>>" + ioException.getMessage();
    } catch (IllegalArgumentException illegalArgumentException) {
        // Catch invalid latitude or longitude values.
        errorMessage = "IllegalArgumentException>>" + illegalArgumentException.getMessage();
    }
    if (addresses != null && !addresses.isEmpty()) {
        address = addresses.get(0);
    }
    return country;
}
```

Section 82.3: Requesting location updates using LocationManager

As always, you need to make sure you have the required permissions.

```
public class MainActivity extends AppCompatActivity implements LocationListener{

    private LocationManager mLocationManager = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    }


    @Override
    protected void onResume() {
        super.onResume();

        try {
            mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
        }
        catch(SecurityException e){
            // The app doesn't have the correct permissions
        }
    }


    @Override
    protected void onPause() {
        try{
            mLocationManager.removeUpdates(this);
        }
        catch (SecurityException e){
            // The app doesn't have the correct permissions
        }

        super.onPause();
    }
```

```
    @Override
    public void onLocationChanged(Location location) {
        // We received a location update!
        Log.i("onLocationChanged", location.toString());
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {

    }

    @Override
    public void onProviderEnabled(String provider) {

    }

    @Override
    public void onProviderDisabled(String provider) {

    }
}
```

Section 82.4: Requesting location updates on a separate thread using LocationManager

As always, you need to make sure you have the required permissions.

```
public class MainActivity extends AppCompatActivity implements LocationListener{

    private LocationManager mLocationManager = null;
    HandlerThread mLocationHandlerThread = null;
    Looper mLocationHandlerLooper = null;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        mLocationHandlerThread = new HandlerThread("locationHandlerThread");
    }


    @Override
    protected void onResume() {
        super.onResume();

        mLocationHandlerThread.start();
        mLocationHandlerLooper = mLocationHandlerThread.getLooper();

        try {
            mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this,
mLocationHandlerLooper);
        }
        catch(SecurityException e){
            // The app doesn't have the correct permissions
        }
    }
```

```
    @Override
    protected void onPause() {
        try{
            mLocationManager.removeUpdates(this);
        }
        catch (SecurityException e){
            // The app doesn't have the correct permissions
        }

        mLocationHandlerLooper = null;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR2)
            mLocationHandlerThread.quitSafely();
        else
            mLocationHandlerThread.quit();

        mLocationHandlerThread = null;


        super.onPause();
    }




    @Override
    public void onLocationChanged(Location location) {
        // We received a location update on a separate thread!
        Log.i("onLocationChanged", location.toString());

        // You can verify which thread you're on by something like this:
        // Log.d("Which thread?", Thread.currentThread() == Looper.getMainLooper().getThread() ? "UI
Thread" : "New thread");
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {

    }

    @Override
    public void onProviderEnabled(String provider) {

    }

    @Override
    public void onProviderDisabled(String provider) {

    }
}
```

Section 82.5: Getting location updates in a BroadcastReceiver

First create a BroadcastReceiver class to handle the incoming Location updates:

```
public class LocationReceiver extends BroadcastReceiver implements Constants {

    @Override
     public void onReceive(Context context, Intent intent) {

        if (LocationResult.hasResult(intent)) {
```

```
            LocationResult locationResult = LocationResult.extractResult(intent);
            Location location = locationResult.getLastLocation();
            if (location != null) {
                // Do something with your location
            } else {
                Log.d(LocationReceiver.class.getSimpleName(), "*** location object is null ***");
            }
        }
    }
}
```

Then when you connect to the GoogleApiClient in the onConnected callback:

```
@Override
public void onConnected(Bundle connectionHint) {
    Intent backgroundIntent = new Intent(this, LocationReceiver.class);
    mBackgroundPendingIntent = backgroundPendingIntent.getBroadcast(getApplicationContext(),
LOCATION_REUEST_CODE, backgroundIntent, PendingIntent.FLAG_CANCEL_CURRENT);
    mFusedLocationProviderApi.requestLocationUpdates(mLocationClient, mLocationRequest,
backgroundPendingIntent);
}
```

Don't forget to remove the location update intent in the appropriate lifecycle callback:

```
@Override
public void onDestroy() {
    if (servicesAvailable && mLocationClient != null) {
        if (mLocationClient.isConnected()) {
            fusedLocationProviderApi.removeLocationUpdates(mLocationClient,
backgroundPendingIntent);
            // Destroy the current location client
            mLocationClient = null;
        } else {
            mLocationClient.unregisterConnectionCallbacks(this);
            mLocationClient = null;
        }
    }
    super.onDestroy();
}
```

Section 82.6: Register geofence

I have created `GeoFenceObserversationService` **singleton** class.

GeoFenceObserversationService.java:

```
public class GeoFenceObserversationService extends Service implements
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
ResultCallback<Status> {

    protected static final String TAG = "GeoFenceObserversationService";
    protected GoogleApiClient mGoogleApiClient;
    protected ArrayList<Geofence> mGeofenceList;
    private boolean mGeofencesAdded;
    private SharedPreferences mSharedPreferences;
    private static GeoFenceObserversationService mInstant;
    public static GeoFenceObserversationService getInstant(){
        return mInstant;
    }
```

```
    @Override
    public void onCreate() {
        super.onCreate();
        mInstant = this;
        mGeofenceList = new ArrayList<Geofence>();
        mSharedPreferences = getSharedPreferences(AppConstants.SHARED_PREFERENCES_NAME,
MODE_PRIVATE);
        mGeofencesAdded = mSharedPreferences.getBoolean(AppConstants.GEOFENCES_ADDED_KEY, false);

        buildGoogleApiClient();
    }


    @Override
    public void onDestroy() {
        mGoogleApiClient.disconnect();
        super.onDestroy();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        return START_STICKY;
    }

    protected void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnected(Bundle connectionHint) {
    }

    @Override
    public void onConnectionFailed(ConnectionResult result) {
    }

    @Override
    public void onConnectionSuspended(int cause) {

    }

    private GeofencingRequest getGeofencingRequest() {

        GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
        builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
        builder.addGeofences(mGeofenceList);
        return builder.build();
    }


    public void addGeofences() {
```

```
            if (!mGoogleApiClient.isConnected()) {
                Toast.makeText(this, getString(R.string.not_connected), Toast.LENGTH_SHORT).show();
                return;
            }

        populateGeofenceList();
        if(!mGeofenceList.isEmpty()){
            try {
                LocationServices.GeofencingApi.addGeofences(mGoogleApiClient,
getGeofencingRequest(), getGeofencePendingIntent()).setResultCallback(this);
            } catch (SecurityException securityException) {
                securityException.printStackTrace();
            }
        }

        }

    public void removeGeofences() {
        if (!mGoogleApiClient.isConnected()) {
            Toast.makeText(this, getString(R.string.not_connected), Toast.LENGTH_SHORT).show();
            return;
        }
        try {

LocationServices.GeofencingApi.removeGeofences(mGoogleApiClient,getGeofencePendingIntent()).setResu
ltCallback(this);
        } catch (SecurityException securityException) {
            securityException.printStackTrace();
        }
    }


    public void onResult(Status status) {

        if (status.isSuccess()) {
            mGeofencesAdded = !mGeofencesAdded;
            SharedPreferences.Editor editor = mSharedPreferences.edit();
            editor.putBoolean(AppConstants.GEOFENCES_ADDED_KEY, mGeofencesAdded);
            editor.apply();
        } else {
            String errorMessage = AppConstants.getErrorString(this,status.getStatusCode());
            Log.i("Geofence", errorMessage);
        }
    }

    private PendingIntent getGeofencePendingIntent() {
        Intent intent = new Intent(this, GeofenceTransitionsIntentService.class);
        return PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
    }

    private void populateGeofenceList() {
        mGeofenceList.clear();
        List<GeoFencingResponce> geoFenceList = getGeofencesList;
        if(geoFenceList!=null&&!geoFenceList.isEmpty()){
            for (GeoFencingResponce obj : geoFenceList){
                mGeofenceList.add(obj.getGeofence());
                Log.i(TAG,"Registered Geofences : " + obj.Id+"-"+obj.Name+"-"+obj.Lattitude+"-
"+obj.Longitude);
            }
        }
    }
}
```

AppConstant:

```
public static final String SHARED_PREFERENCES_NAME = PACKAGE_NAME + ".SHARED_PREFERENCES_NAME";
 public static final String GEOFENCES_ADDED_KEY = PACKAGE_NAME + ".GEOFENCES_ADDED_KEY";
 public static final String DETECTED_GEOFENCES = "detected_geofences";
 public static final String DETECTED_BEACONS = "detected_beacons";

 public static String getErrorString(Context context, int errorCode) {
     Resources mResources = context.getResources();
     switch (errorCode) {
         case GeofenceStatusCodes.GEOFENCE_NOT_AVAILABLE:
             return mResources.getString(R.string.geofence_not_available);
         case GeofenceStatusCodes.GEOFENCE_TOO_MANY_GEOFENCES:
             return mResources.getString(R.string.geofence_too_many_geofences);
         case GeofenceStatusCodes.GEOFENCE_TOO_MANY_PENDING_INTENTS:
             return mResources.getString(R.string.geofence_too_many_pending_intents);
         default:
             return mResources.getString(R.string.unknown_geofence_error);
     }
 }
```

Where I started Service ? From Application class

- `startService(new Intent(getApplicationContext(),GeoFenceObserversationService.class));`

How I registered Geofences ?

- `GeoFenceObserversationService.getInstant().addGeofences();`

Chapter 83: Theme, Style, Attribute

Section 83.1: Define primary, primary dark, and accent colors

You can customize your theme's color palette.

Using **framework** APIs

Version ≥ 5.0

```
<style name="AppTheme" parent="Theme.Material">
    <item name="android:colorPrimary">@color/primary</item>
    <item name="android:colorPrimaryDark">@color/primary_dark</item>
    <item name="android:colorAccent">@color/accent</item>
</style>
```

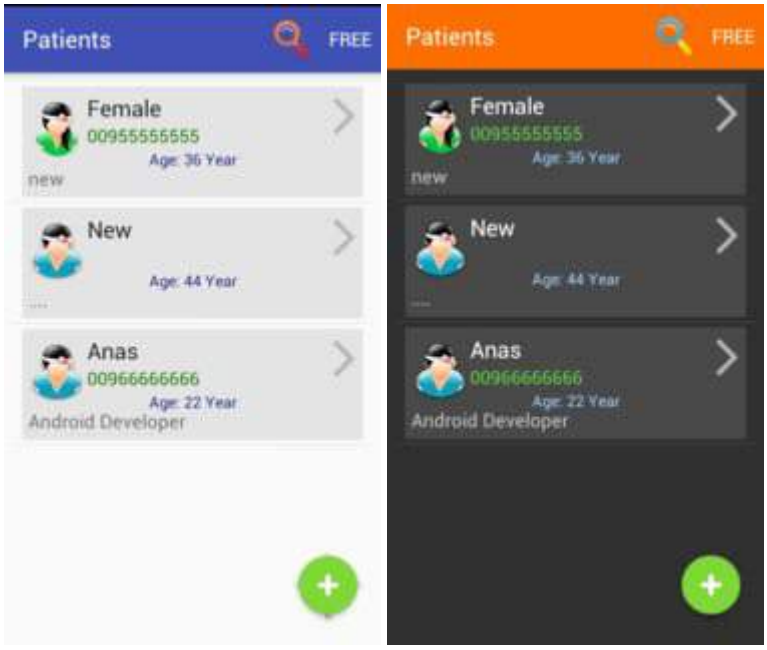
Using the **Appcompat support library** (and `AppCompatActivity`)

Version ≥ 2.1.x

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/primary</item>
    <item name="colorPrimaryDark">@color/primary_dark</item>
    <item name="colorAccent">@color/accent</item>
</style>
```

Section 83.2: Multiple Themes in one App

Using more than one theme in your Android application, you can add custom colors to every theme, to be like this:



First, we have to add our themes to `style.xml` like this:

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">

</style>

<!--  -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >
```

```
</style>
......
```

Above you can see **OneTheme** and **TwoTheme**.

Now, go to your `AndroidManifest.xml` and add this line: `android:theme="@style/OneTheme"` to your *application* tag, this will make **OneTheme** the default theme:

```
<application
       android:theme="@style/OneTheme"
       ...>
```

Create new xml file named `attrs.xml` and add this code :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <attr name="custom_red" format="color" />
    <attr name="custom_blue" format="color" />
    <attr name="custom_green" format="color" />
</resources>
<!-- add all colors you need (just color's name) -->
```

Go back to `style.xml` and add these colors with its values for each theme :

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="custom_red">#8b030c</item>
    <item name="custom_blue">#0f1b8b</item>
    <item name="custom_green">#1c7806</item>
</style>

<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >
    <item name="custom_red">#ff606b</item>
    <item name="custom_blue">#99cfff</item>
    <item name="custom_green">#62e642</item>
</style>
```

Now you have custom colors for each theme, let's add these color to our views.

Add **custom_blue** color to the TextView by using "?attr/" :

Go to your imageView and add this color :

```
<TextView>
    android:id="@+id/txte_view"
    android:textColor="?attr/custom_blue" />
```

Mow we can change the theme just by single line `setTheme(R.style.TwoTheme);` this line must be before `setContentView()` method in `onCreate()` method, like this `Activity.java` :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setTheme(R.style.TwoTheme);
    setContentView(R.layout.main_activity);
    ....
}
```

change theme for all activities at once

If we want to change the theme for all activities, we have to create new class named MyActivity extends `AppCompatActivity` class (or `Activity` class) and add line `setTheme(R.style.TwoTheme);` to **onCreate()** method:

```
public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (new MySettings(this).isDarkTheme())
            setTheme(R.style.TwoTheme);
    }
}
```

Finally, go to all your activities add make all of them extend the **MyActivity** base class:

```
public class MainActivity extends MyActivity {
    ....
}
```

In order to change the theme, just go to **MyActivity** and change `R.style.TwoTheme` to your theme (`R.style.OneTheme` , `R.style.ThreeTheme` ....).

Section 83.3: Navigation Bar Color (API 21+)

`Version ≥ 5.0`

This attribute is used to change the navigation bar (one, that contain Back, Home Recent button). Usually it is black, however it's color can be changed.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:navigationBarColor">@color/my_color</item>
</style>
```

Section 83.4: Use Custom Theme Per Activity

In themes.xml:

```
<style name="MyActivityTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

In AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/Theme.AppCompat">

    <activity
        android:name=".MyActivity"
        android:theme="@style/MyActivityTheme" />

</application>
```

Section 83.5: Light Status Bar (API 23+)

This attribute can change the background of the Status Bar icons (at the top of the screen) to white.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

Section 83.6: Use Custom Theme Globally

In themes.xml:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

In AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activity declarations here -->

</application>
```

Section 83.7: Overscroll Color (API 21+)

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

Section 83.8: Ripple Color (API 21+)

Version ≥ 5.0

The ripple animation is shown when user presses clickable views.

You can use the same ripple color used by your app assigning the `?android:colorControlHighlight` in your views. You can customize this color by changing the `android:colorControlHighlight` attribute in your theme:

This effect color can be changed:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

Or, if you are using a Material Theme:

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

Section 83.9: Translucent Navigation and Status Bars (API 19+)

The navigation bar (at the bottom of the screen) can be transparent. Here is the way to achieve it.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

The Status Bar (top of the screen) can be made transparent, by applying this attribute to the style:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

Section 83.10: Theme inheritance

When defining themes, one usually uses the theme provided by the system, and then changes modifies the look to fit his own application. For example, this is how the `Theme.AppCompat` theme is inherited:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

This theme now has all the properties of the standard `Theme.AppCompat` theme, except the ones we explicitly changed.

There is also a shortcut when inheriting, usually used when one inherits from his own theme:

```
<style name="AppTheme.Red">
    <item name="colorAccent">@color/red</item>
</style>
```

Since it already has `AppTheme.` in the start of it's name, it automatically inherits it, without needing to define the `parent` theme. This is useful when you need to create specific styles for a part (for example, a single Activity) of your app.

Chapter 84: MediaPlayer

Section 84.1: Basic creation and playing

MediaPlayer class can be used to control playback of audio/video files and streams.

Creation of MediaPlayer object can be of three types:

1. Media from local resource

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

2. From local URI (obtained from ContentResolver)

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

3. From external URL

```
String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Section 84.2: Media Player with Buffer progress and play position

```
public class SoundActivity extends Activity  {

    private MediaPlayer mediaPlayer;
    ProgressBar progress_bar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tool_sound);
        mediaPlayer = new MediaPlayer();
        mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        progress_bar = (ProgressBar) findViewById(R.id.progress_bar);

        btn_play_stop.setEnabled(false);
        btn_play_stop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(mediaPlayer.isPlaying()) {
                    mediaPlayer.pause();
                    btn_play_stop.setImageResource(R.drawable.ic_pause_black_24dp);
                } else {
                    mediaPlayer.start();
```

```
                    btn_play_stop.setImageResource(R.drawable.ic_play_arrow_black_24px);
                }
            }
        });

        mediaPlayer.setDataSource(proxyUrl);
        mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                observer.stop();
                progress_bar.setProgress(mp.getCurrentPosition());
                // TODO Auto-generated method stub
                mediaPlayer.stop();
                mediaPlayer.reset();
            }
        });
        mediaPlayer.setOnBufferingUpdateListener(new MediaPlayer.OnBufferingUpdateListener() {
            @Override
            public void onBufferingUpdate(MediaPlayer mp, int percent) {
                progress_bar.setSecondaryProgress(percent);
            }
        });
        mediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
            @Override
            public void onPrepared(MediaPlayer mediaPlayer) {
                btn_play_stop.setEnabled(true);

            }
        });
        observer = new MediaObserver();
        mediaPlayer.prepare();
        mediaPlayer.start();
        new Thread(observer).start();
    }


    private MediaObserver observer = null;

    private class MediaObserver implements Runnable {
        private AtomicBoolean stop = new AtomicBoolean(false);

        public void stop() {
            stop.set(true);
        }

        @Override
        public void run() {
            while (!stop.get()) {
                progress_bar.setProgress((int)((double)mediaPlayer.getCurrentPosition() /
(double)mediaPlayer.getDuration()*100));
                try {
                    Thread.sleep(200);
                } catch (Exception ex) {
                    Logger.log(ToolSoundActivity.this, ex);
                }

            }
        }
    }

    @Override
    protected void onDestroy() {
```

```
        super.onDestroy();
        mediaPlayer.stop();
    }
}
```

```
<LinearLayout
   android:gravity="bottom"
   android:layout_gravity="bottom"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="1"
   android:weightSum="1">

    <LinearLayout
       android:orientation="horizontal"
       android:layout_width="match_parent"
       android:layout_height="wrap_content">

        <ImageButton
           app:srcCompat="@drawable/ic_play_arrow_black_24px"
           android:layout_width="48dp"
           android:layout_height="48dp"
           android:id="@+id/btn_play_stop" />

        <ProgressBar
           android:padding="8dp"
           android:progress="0"
           android:id="@+id/progress_bar"
           style="@style/Widget.AppCompat.ProgressBar.Horizontal"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:layout_gravity="center" />

    </LinearLayout>

</LinearLayout>
```

Section 84.3: Getting system ringtones

This example demonstrates how to fetch the URI's of system ringtones (`RingtoneManager.TYPE_RINGTONE`):

```
private List<Uri> loadLocalRingtonesUris() {
        List<Uri> alarms = new ArrayList<>();
        try {
            RingtoneManager ringtoneMgr = new RingtoneManager(getActivity());
            ringtoneMgr.setType(RingtoneManager.TYPE_RINGTONE);

            Cursor alarmsCursor = ringtoneMgr.getCursor();
            int alarmsCount = alarmsCursor.getCount();
            if (alarmsCount == 0 && !alarmsCursor.moveToFirst()) {
                alarmsCursor.close();
                return null;
            }

            while (!alarmsCursor.isAfterLast() && alarmsCursor.moveToNext()) {
                int currentPosition = alarmsCursor.getPosition();
                alarms.add(ringtoneMgr.getRingtoneUri(currentPosition));
            }
```

```
        } catch (Exception ex) {
            ex.printStackTrace();
        }

        return alarms;
    }
```

The list depends on the types of requested ringtones. The possibilities are:

- `RingtoneManager.TYPE_RINGTONE`
- `RingtoneManager.TYPE_NOTIFICATION`
- `RingtoneManager.TYPE_ALARM`
- `RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM`

In order to get the Ringtones as `android.media.Ringtone` every `Uri` must be resolved by the `RingtoneManager`:

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

To play the sound, use the method:

```
public void setDataSource(Context context, Uri uri)
```

from `android.media.MediaPlayer`. `MediaPlayer` must be initialised and prepared according to the State diagram

Section 84.4: Asynchronous prepare

The `MediaPlayer$prepare()` is a blocking call and will freeze the UI till execution completes. To solve this problem, `MediaPlayer$prepareAsync()` can be used.

```
mMediaPlayer = ... // Initialize it here
mMediaPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener(){
    @Override
    public void onPrepared(MediaPlayer player) {
        // Called when the MediaPlayer is ready to play
        mMediaPlayer.start();
    }
}); // Set callback for when prepareAsync() finishes
mMediaPlayer.prepareAsync(); // Prepare asynchronously to not block the Main Thread
```

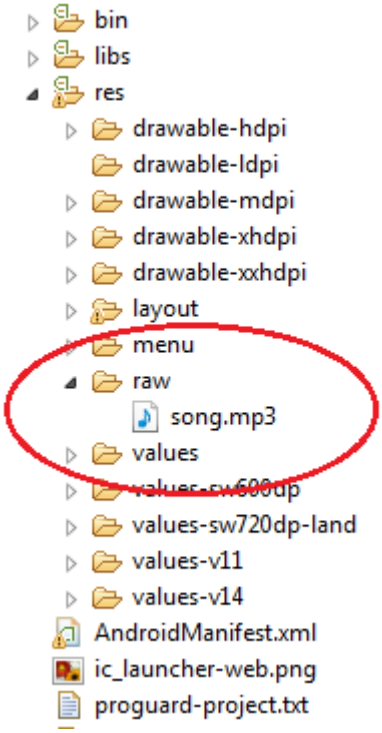
On synchronous operations, errors would normally be signaled with an exception or an error code, but whenever you use asynchronous resources, you should make sure your application is notified of errors appropriately. For MediaPlayer,

```
mMediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener(){
        @Override
        public boolean onError(MediaPlayer mp, int what, int extra) {
            // ... react appropriately ...
            // The MediaPlayer has moved to the Error state, must be reset!
            // Then return true if the error has been handled
        }
});
```

Section 84.5: Import audio into androidstudio and play it

This is an example how to get the play an audio file which you already have on your pc/laptop .First create a new

directory under res and name it as raw like this



copy the audio which you want to play into this folder .It may be a .mp3 or .wav file.

Now for example on button click you want to play this sound ,here is how it is done

```
public class MainActivity extends AppCompatActivity {
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.aboutapp_activity);

    MediaPlayer song=MediaPlayer.create(this, R.raw.song);

    Button button=(Button)findViewById(R.id.button);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
           song.start();
        }
    });
}
}
```

This will play the song only once when the button is clicked,if you want to replay the song on every button click write code like this

```
public class MainActivity extends AppCompatActivity {
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.aboutapp_activity);

  MediaPlayer song=MediaPlayer.create(this, R.raw.song);

  Button button=(Button)findViewById(R.id.button);
  button.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View view) {
        if (song.isPlaying()) {
            song.reset();
            song= MediaPlayer.create(getApplicationContext(), R.raw.song);
       }
         song.start();
     }
   });
 }
}
```

Section 84.6: Getting and setting system volume

Audio stream types

There are different profiles of ringtone streams. Each one of them has it's different volume.

Every example here is written for `AudioManager.STREAM_RING` stream type. However this is not the only one. The available stream types are:

- `STREAM_ALARM`
- `STREAM_DTMF`
- `STREAM_MUSIC`
- `STREAM_NOTIFICATION`
- `STREAM_RING`
- `STREAM_SYSTEM`
- `STREAM_VOICE_CALL`

Setting volume

To get the volume of specific profile, call:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);
```

This value is very little useful, when the maximum value for the stream is not known:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);
```

The ratio of those two value will give a relative volume (0 < volume < 1):

```
float volume = ((float) currentVolume) / streamMaxVolume
```

Adjusting volume by one step

To make the volume for the stream higher by one step, call:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);
```

To make the volume for the stream lower by one step, call:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

Setting MediaPlayer to use specific stream type

There is a helper function from `MediaPlayer` class to do this.
Just call `void setAudioStreamType(int streamtype)`:

```
MediaPlayer mMedia = new MediaPlayer();
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

Chapter 85: Android Sound and Media

Section 85.1: How to pick image and video for api >19

Here is a tested code for image and video.It will work for all APIs less than 19 and greater than 19 as well.

Image:

```
if (Build.VERSION.SDK_INT <= 19) {
                        Intent i = new Intent();
                        i.setType("image/*");
                        i.setAction(Intent.ACTION_GET_CONTENT);
                        i.addCategory(Intent.CATEGORY_OPENABLE);
                        startActivityForResult(i, 10);
                    } else if (Build.VERSION.SDK_INT > 19) {
                        Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                        startActivityForResult(intent, 10);
                    }
```

Video:

```
if (Build.VERSION.SDK_INT <= 19) {
                        Intent i = new Intent();
                        i.setType("video/*");
                        i.setAction(Intent.ACTION_GET_CONTENT);
                        i.addCategory(Intent.CATEGORY_OPENABLE);
                        startActivityForResult(i, 20);
                    } else if (Build.VERSION.SDK_INT > 19) {
                        Intent intent = new Intent(Intent.ACTION_PICK,
android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI);
                        startActivityForResult(intent, 20);
                    }
```

.

```
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == Activity.RESULT_OK) {

            if (requestCode == 10) {
                Uri selectedImageUri = data.getData();
                String selectedImagePath = getRealPathFromURI(selectedImageUri);
            } else if (requestCode == 20) {
                Uri selectedVideoUri = data.getData();
                String selectedVideoPath = getRealPathFromURI(selectedVideoUri);
            }

    public String getRealPathFromURI(Uri uri) {
            if (uri == null) {
                return null;
            }
            String[] projection = {MediaStore.Images.Media.DATA};
            Cursor cursor = getActivity().getContentResolver().query(uri, projection, null, null,
null);
            if (cursor != null) {
                int column_index = cursor
                        .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
```

```
                cursor.moveToFirst();
                return cursor.getString(column_index);
            }
            return uri.getPath();
        }
```

Section 85.2: Play sounds via SoundPool

```
public class PlaySound extends Activity implements OnTouchListener {
    private SoundPool soundPool;
    private int soundID;
    boolean loaded = false;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            View view = findViewById(R.id.textView1);
            view.setOnTouchListener(this);
            // Set the hardware buttons to control the music
            this.setVolumeControlStream(AudioManager.STREAM_MUSIC);
            // Load the sound
            soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC, 0);
            soundPool.setOnLoadCompleteListener(new OnLoadCompleteListener() {
                    @Override
                    public void onLoadComplete(SoundPool soundPool, int sampleId,
                                    int status) {
                            loaded = true;
                    }
            });
            soundID = soundPool.load(this, R.raw.sound1, 1);

    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    // Getting the user sound settings
                    AudioManager audioManager = (AudioManager) getSystemService(AUDIO_SERVICE);
                    float actualVolume = (float) audioManager
                                    .getStreamVolume(AudioManager.STREAM_MUSIC);
                    float maxVolume = (float) audioManager
                                    .getStreamMaxVolume(AudioManager.STREAM_MUSIC);
                    float volume = actualVolume / maxVolume;
                    // Is the sound loaded already?
                    if (loaded) {
                            soundPool.play(soundID, volume, volume, 1, 0, 1f);
                            Log.e("Test", "Played sound");
                    }
            }
            return false;
    }
}
```

Chapter 86: MediaSession

Section 86.1: Receiving and handling button events

This example creates a `MediaSession` object when a `Service` is started. The `MediaSession` object is released when the `Service` gets destroyed:

```
public final class MyService extends Service {
    private static MediaSession s_mediaSession;

    @Override
    public void onCreate() {
        // Instantiate new MediaSession object.
        configureMediaSession();
    }

    @Override
    public void onDestroy() {
        if (s_mediaSession != null)
            s_mediaSession.release();
    }
}
```

The following method instantiates and configures the `MediaSession` button callbacks:

```
private void configureMediaSession {
    s_mediaSession = new MediaSession(this, "MyMediaSession");

    // Overridden methods in the MediaSession.Callback class.
    s_mediaSession.setCallback(new MediaSession.Callback() {
        @Override
        public boolean onMediaButtonEvent(Intent mediaButtonIntent) {
            Log.d(TAG, "onMediaButtonEvent called: " + mediaButtonIntent);
            KeyEvent ke = mediaButtonIntent.getParcelableExtra(Intent.EXTRA_KEY_EVENT);
            if (ke != null && ke.getAction() == KeyEvent.ACTION_DOWN) {
                int keyCode = ke.getKeyCode();
                Log.d(TAG, "onMediaButtonEvent Received command: " + ke);
            }
            return super.onMediaButtonEvent(mediaButtonIntent);
        }

        @Override
        public void onSkipToNext() {
            Log.d(TAG, "onSkipToNext called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToNext called",
Toast.LENGTH_SHORT).show();
            skipToNextPlaylistItem(); // Handle this button press.
            super.onSkipToNext();
        }

        @Override
        public void onSkipToPrevious() {
            Log.d(TAG, "onSkipToPrevious called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToPrevious called",
Toast.LENGTH_SHORT).show();
            skipToPreviousPlaylistItem(); // Handle this button press.
            super.onSkipToPrevious();
        }
```

```
        @Override
        public void onPause() {
            Log.d(TAG, "onPause called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onPause called", Toast.LENGTH_SHORT).show();
            mpPause(); // Pause the player.
            super.onPause();
        }

        @Override
        public void onPlay() {
            Log.d(TAG, "onPlay called (media button pressed)");
            mpStart(); // Start player/playback.
            super.onPlay();
        }

        @Override
        public void onStop() {
            Log.d(TAG, "onStop called (media button pressed)");
            mpReset(); // Stop and/or reset the player.
            super.onStop();
        }
    });

    s_mediaSession.setFlags(MediaSession.FLAG_HANDLES_MEDIA_BUTTONS |
MediaSession.FLAG_HANDLES_TRANSPORT_CONTROLS);
    s_mediaSession.setActive(true);
}
```

The following method sends meta data (stored in a HashMap) to the device using A2DP:

```
void sendMetaData(@NonNull final HashMap<String, String> hm) {
    // Return if Bluetooth A2DP is not in use.
    if (!((AudioManager) getSystemService(Context.AUDIO_SERVICE)).isBluetoothA2dpOn()) return;

    MediaMetadata metadata = new MediaMetadata.Builder()
            .putString(MediaMetadata.METADATA_KEY_TITLE, hm.get("Title"))
            .putString(MediaMetadata.METADATA_KEY_ALBUM, hm.get("Album"))
            .putString(MediaMetadata.METADATA_KEY_ARTIST, hm.get("Artist"))
            .putString(MediaMetadata.METADATA_KEY_AUTHOR, hm.get("Author"))
            .putString(MediaMetadata.METADATA_KEY_COMPOSER, hm.get("Composer"))
            .putString(MediaMetadata.METADATA_KEY_WRITER, hm.get("Writer"))
            .putString(MediaMetadata.METADATA_KEY_DATE, hm.get("Date"))
            .putString(MediaMetadata.METADATA_KEY_GENRE, hm.get("Genre"))
            .putLong(MediaMetadata.METADATA_KEY_YEAR, tryParse(hm.get("Year")))
            .putLong(MediaMetadata.METADATA_KEY_DURATION, tryParse(hm.get("Raw Duration")))
            .putLong(MediaMetadata.METADATA_KEY_TRACK_NUMBER, tryParse(hm.get("Track Number")))
            .build();

    s_mediaSession.setMetadata(metadata);
}
```

The following method sets the PlaybackState. It also sets which button actions the `MediaSession` will respond to:

```
private void setPlaybackState(@NonNull final int stateValue) {
    PlaybackState state = new PlaybackState.Builder()
            .setActions(PlaybackState.ACTION_PLAY | PlaybackState.ACTION_SKIP_TO_NEXT
                    | PlaybackState.ACTION_PAUSE | PlaybackState.ACTION_SKIP_TO_PREVIOUS
                    | PlaybackState.ACTION_STOP | PlaybackState.ACTION_PLAY_PAUSE)
            .setState(stateValue, PlaybackState.PLAYBACK_POSITION_UNKNOWN, 0)
            .build();
```

```
    s_mediaSession.setPlaybackState(state);
}
```

Chapter 87: MediaStore

Section 87.1: Fetch Audio/MP3 files from specific folder of device or fetch all files

First, add the following permissions to the manifest of your project in order to enable device storage access:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Then, create the file *AudioModel.class* and put the following model class into it in order to allow getting and setting list items:

```
public class AudioModel {
    String aPath;
    String aName;
    String aAlbum;
    String aArtist;

    public String getaPath() {
        return aPath;
    }
    public void setaPath(String aPath) {
        this.aPath = aPath;
    }
    public String getaName() {
        return aName;
    }
    public void setaName(String aName) {
        this.aName = aName;
    }
    public String getaAlbum() {
        return aAlbum;
    }
    public void setaAlbum(String aAlbum) {
        this.aAlbum = aAlbum;
    }
    public String getaArtist() {
        return aArtist;
    }
    public void setaArtist(String aArtist) {
        this.aArtist = aArtist;
    }
}
```

Next, use the following method to read all MP3 files from a folder of your device or to read all files of your device:

```
public List<AudioModel> getAllAudioFromDevice(final Context context) {
    final List<AudioModel> tempAudioList = new ArrayList<>();

    Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    String[] projection = {MediaStore.Audio.AudioColumns.DATA, MediaStore.Audio.AudioColumns.TITLE,
MediaStore.Audio.AudioColumns.ALBUM, MediaStore.Audio.ArtistColumns.ARTIST,};
    Cursor c = context.getContentResolver().query(uri, projection, MediaStore.Audio.Media.DATA + "
like ? ", new String[]{"%utm%"}, null);

    if (c != null) {
        while (c.moveToNext()) {
```

```
            AudioModel audioModel = new AudioModel();
            String path = c.getString(0);
            String name = c.getString(1);
            String album = c.getString(2);
            String artist = c.getString(3);

            audioModel.setaName(name);
            audioModel.setaAlbum(album);
            audioModel.setaArtist(artist);
            audioModel.setaPath(path);

            Log.e("Name :" + name, " Album :" + album);
            Log.e("Path :" + path, " Artist :" + artist);

            tempAudioList.add(audioModel);
        }
        c.close();
    }

    return tempAudioList;
}
```

The code above will return a list of all MP3 files with the music's name, path, artist, and album. For more details please refer to the Media.Store.Audio documentation.

In order to read files of a specific folder, use the following query (you need to replace the folder name):

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ? ",
    new String[]{"%yourFolderName%"}, // Put your device folder / file location here.
    null);
```

If you want to retrieve all files from your device, then use the following query:

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);
```

Note: Don't forget to enable storage access permissions.

Now, all you have to do is to call the method above in order to get the MP3 files:

```
getAllAudioFromDevice(this);
```

Example with Activity

```
public class ReadAudioFilesActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_audio_list);

        /**
         * This will return a list of all MP3 files. Use the list to display data.
         */
        getAllAudioFromDevice(this);
    }
```

```
    // Method to read all the audio/MP3 files.
    public List<AudioModel> getAllAudioFromDevice(final Context context) {
        final List<AudioModel> tempAudioList = new ArrayList<>();

        Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        String[] projection =
{MediaStore.Audio.AudioColumns.DATA,MediaStore.Audio.AudioColumns.TITLE
,MediaStore.Audio.AudioColumns.ALBUM, MediaStore.Audio.ArtistColumns.ARTIST,};
        Cursor c = context.getContentResolver().query(uri, projection, MediaStore.Audio.Media.DATA
+ " like ? ", new String[]{"%utm%"}, null);

        if (c != null) {
            while (c.moveToNext()) {
                // Create a model object.
                AudioModel audioModel = new AudioModel();

                String path = c.getString(0);   // Retrieve path.
                String name = c.getString(1);   // Retrieve name.
                String album = c.getString(2);  // Retrieve album name.
                String artist = c.getString(3); // Retrieve artist name.

                // Set data to the model object.
                audioModel.setaName(name);
                audioModel.setaAlbum(album);
                audioModel.setaArtist(artist);
                audioModel.setaPath(path);

                Log.e("Name :" + name, " Album :" + album);
                Log.e("Path :" + path, " Artist :" + artist);

                // Add the model object to the list .
                tempAudioList.add(audioModel);
            }
            c.close();
        }

        // Return the list.
        return tempAudioList;
    }
}
```

Chapter 88: Multidex and the Dex Method Limit

DEX means Android app's (APK) executable bytecode files in the form of Dalvik Executable (DEX) files, which contain the compiled code used to run your app.

The Dalvik Executable specification limits the total number of methods that can be referenced within a single DEX file to 65,536 (64K)—including Android framework methods, library methods, and methods in your own code.

To overcome this limit requires configure your app build process to generate more than one DEX file, known as a Multidex.

Section 88.1: Enabling Multidex

In order to enable a multidex configuration you need:

- to change your Gradle build configuration
- to use a `MultiDexApplication` or enable the MultiDex in your `Application` class

Gradle configuration

In `app/build.gradle` add these parts:

```
android {
    compileSdkVersion 24
    buildToolsVersion "24.0.1"

    defaultConfig {
        ...
        minSdkVersion 14
        targetSdkVersion 24
        ...

        // Enabling multidex support.
        multiDexEnabled true
    }
    ...
}

dependencies {
  compile 'com.android.support:multidex:1.0.1'
}
```

Enable MultiDex in your Application

Then proceed with one of three options:

- Multidex by extending Application
- Multidex by extending MultiDexApplication
- Multidex by using MultiDexApplication directly

When these configuration settings are added to an app, the Android build tools construct a primary dex (classes.dex) and supporting (classes2.dex, classes3.dex) as needed.
The build system will then package them into an APK file for distribution.

Section 88.2: Multidex by extending Application

Use this option if your project requires an `Application` subclass.

Specify this `Application` subclass using the `android:name` property in the manifest file inside the `application` tag.

In the `Application` subclass, add the `attachBaseContext()` method override, and in that method call `MultiDex.install()`:

```
package com.example;

import android.app.Application;
import android.content.Context;

/**
 * Extended application that support multidex
 */
public class MyApplication extends Application {

    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
}
```

Ensure that the `Application` subclass is specified in the `application` tag of your AndroidManifest.xml:

```
<application
   android:name="com.example.MyApplication"
   android:icon="@drawable/ic_launcher"
   android:label="@string/app_name">
</application>
```

Section 88.3: Multidex by extending MultiDexApplication

This is very similar to using an `Application` subclass and overriding the `attachBaseContext()` method.

However, using this method, you don't need to override `attachBaseContext()` as this is already done in the `MultiDexApplication` superclass.

Extend `MultiDexApplication` instead of `Application`:

```
package com.example;

import android.support.multidex.MultiDexApplication;
import android.content.Context;

/**
 * Extended MultiDexApplication
 */
public class MyApplication extends MultiDexApplication {

     // No need to override attachBaseContext()

     //.........
}
```

Add this class to your AndroidManifest.xml exactly as if you were extending Application:

```
<application
   android:name="com.example.MyApplication"
   android:icon="@drawable/ic_launcher"
   android:label="@string/app_name">
</application>
```

Section 88.4: Multidex by using MultiDexApplication directly

Use this option if you don't need an `Application` subclass.

This is the simplest option, but this way you can't provide your own `Application` subclass. If an `Application` subclass is needed, you will have to switch to one of the other options to do so.

For this option, simply specify the fully-qualified class name `android.support.multidex.MultiDexApplication` for the `android:name` property of the `application` tag in the AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.android.multidex.myapplication">
    <application
       ...
       android:name="android.support.multidex.MultiDexApplication">
        ...
    </application>
</manifest>
```

Section 88.5: Counting Method References On Every Build (Dexcount Gradle Plugin)

The dexcount plugin counts methods and class resource count after a successful build.

Add the plugin in the `app/build.gradle`:

```
apply plugin: 'com.android.application'

buildscript {
    repositories {
        mavenCentral() // or jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

Apply the plugin in the `app/build.gradle` file:

```
apply plugin: 'com.getkeepsafe.dexcount'
```

Look for the output data generated by the plugin in:

../app/build/outputs/dexcount

Especially useful is the .html chart in:

../app/build/outputs/dexcount/debugChart/index.html

Chapter 89: Data Synchronization with Sync Adapter

Section 89.1: Dummy Sync Adapter with Stub Provider

SyncAdapter

```
/**
 * Define a sync adapter for the app.
 * <p/>
 * <p>This class is instantiated in {@link SyncService}, which also binds SyncAdapter to the system.
 * SyncAdapter should only be initialized in SyncService, never anywhere else.
 * <p/>
 * <p>The system calls onPerformSync() via an RPC call through the IBinder object supplied by
 * SyncService.
 */
class SyncAdapter extends AbstractThreadedSyncAdapter {
    /**
     * Constructor. Obtains handle to content resolver for later use.
     */
    public SyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
    }

    /**
     * Constructor. Obtains handle to content resolver for later use.
     */
    public SyncAdapter(Context context, boolean autoInitialize, boolean allowParallelSyncs) {
        super(context, autoInitialize, allowParallelSyncs);
    }

    @Override
    public void onPerformSync(Account account, Bundle extras, String authority,
                              ContentProviderClient provider, SyncResult syncResult) {
        //Jobs you want to perform in background.
        Log.e("" + account.name, "Sync Start");
    }

}
```

Sync Service

```
/**
 * Define a Service that returns an IBinder for the
 * sync adapter class, allowing the sync adapter framework to call
 * onPerformSync().
 */
public class SyncService extends Service {
    // Storage for an instance of the sync adapter
    private static SyncAdapter sSyncAdapter = null;
    // Object to use as a thread-safe lock
    private static final Object sSyncAdapterLock = new Object();

    /*
     * Instantiate the sync adapter object.
     */
    @Override
    public void onCreate() {
        /*
         * Create the sync adapter as a singleton.
         * Set the sync adapter as syncable
```

```
         * Disallow parallel syncs
         */
        synchronized (sSyncAdapterLock) {
            if (sSyncAdapter == null) {
                sSyncAdapter = new SyncAdapter(getApplicationContext(), true);
            }
        }
    }

    /**
     * Return an object that allows the system to invoke
     * the sync adapter.
     */
    @Override
    public IBinder onBind(Intent intent) {
        /*
         * Get the object that allows external processes
         * to call onPerformSync(). The object is created
         * in the base class code when the SyncAdapter
         * constructors call super()
         */
        return sSyncAdapter.getSyncAdapterBinder();
    }
}
```

Authenticator

```
public class Authenticator extends AbstractAccountAuthenticator {
    // Simple constructor
    public Authenticator(Context context) {
        super(context);
    }

    // Editing properties is not supported
    @Override
    public Bundle editProperties(
            AccountAuthenticatorResponse r, String s) {
        throw new UnsupportedOperationException();
    }

    // Don't add additional accounts
    @Override
    public Bundle addAccount(
            AccountAuthenticatorResponse r,
            String s,
            String s2,
            String[] strings,
            Bundle bundle) throws NetworkErrorException {
        return null;
    }

    // Ignore attempts to confirm credentials
    @Override
    public Bundle confirmCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            Bundle bundle) throws NetworkErrorException {
        return null;
    }

    // Getting an authentication token is not supported
    @Override
    public Bundle getAuthToken(
```

```
            AccountAuthenticatorResponse r,
            Account account,
            String s,
            Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }

    // Getting a label for the auth token is not supported
    @Override
    public String getAuthTokenLabel(String s) {
        throw new UnsupportedOperationException();
    }

    // Updating user credentials is not supported
    @Override
    public Bundle updateCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            String s, Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }

    // Checking features for the account is not supported
    @Override
    public Bundle hasFeatures(
            AccountAuthenticatorResponse r,
            Account account, String[] strings) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }
}
```

Authenticator Service

```
/**
 * A bound Service that instantiates the authenticator
 * when started.
 */
public class AuthenticatorService extends Service {
    // Instance field that stores the authenticator object
    private Authenticator mAuthenticator;
    @Override
    public void onCreate() {
        // Create a new authenticator object
        mAuthenticator = new Authenticator(this);
    }
    /*
     * When the system binds to this Service to make the RPC call
     * return the authenticator's IBinder.
     */
    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }
}
```

AndroidManifest.xml additions

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_SYNC_SETTINGS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />

     <service
       android:name=".syncAdapter.SyncService"
```

```
        android:exported="true">
        <intent-filter>
            <action android:name="android.content.SyncAdapter" />
        </intent-filter>
        <meta-data
            android:name="android.content.SyncAdapter"
            android:resource="@xml/syncadapter" />
    </service>

    <service android:name=".authenticator.AuthenticatorService">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator" />
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator" />
    </service>

    <provider
        android:name=".provider.StubProvider"
        android:authorities="com.yourpackage.provider"
        android:exported="false"
        android:syncable="true" />
```

res/xml/authenticator.xml

```
<?xml version="1.0" encoding="utf-8"?>
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:smallIcon="@mipmap/ic_launcher" />
```

res/xml/syncadapter.xml

```
<?xml version="1.0" encoding="utf-8"?>
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.yourpackage.android"
    android:allowParallelSyncs="false"
    android:contentAuthority="com.yourpackage.provider"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"
    android:userVisible="false" />
```

StubProvider

```
/*
 * Define an implementation of ContentProvider that stubs out
 * all methods
 */
public class StubProvider extends ContentProvider {
    /*
     * Always return true, indicating that the
     * provider loaded correctly.
     */
    @Override
    public boolean onCreate() {
        return true;
    }

    /*
     * Return no type for MIME type
     */
    @Override
    public String getType(Uri uri) {
        return null;
```

```
    }

    /*
     * query() always returns no results
     *
     */
    @Override
    public Cursor query(
            Uri uri,
            String[] projection,
            String selection,
            String[] selectionArgs,
            String sortOrder) {
        return null;
    }

    /*
     * insert() always returns null (no URI)
     */
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }

    /*
     * delete() always returns "no rows affected" (0)
     */
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        return 0;
    }

    /*
     * update() always returns "no rows affected" (0)
     */
    public int update(
            Uri uri,
            ContentValues values,
            String selection,
            String[] selectionArgs) {
        return 0;
    }
}
```

Call this function on successful login to create an account with the logged-in user ID

```
public Account CreateSyncAccount(Context context, String accountName) {
        // Create the account type and default account
        Account newAccount = new Account(
                accountName, "com.yourpackage");
        // Get an instance of the Android account manager
        AccountManager accountManager =
                (AccountManager) context.getSystemService(
                        ACCOUNT_SERVICE);
        /*
         * Add the account and account type, no password or user data
         * If successful, return the Account object, otherwise report an error.
         */
        if (accountManager.addAccountExplicitly(newAccount, null, null)) {
            /*
             * If you don't set android:syncable="true" in
             * in your <provider> element in the manifest,
             * then call context.setIsSyncable(account, AUTHORITY, 1)
```

```
            * here.
            */
        } else {
            /*
            * The account exists or some other error occurred. Log this, report it,
            * or handle it internally.
            */
        }
        return newAccount;
    }
```

Forcing a Sync

```
Bundle bundle = new Bundle();
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED, true);
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_FORCE, true);
bundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL, true);
ContentResolver.requestSync(null, MyContentProvider.getAuthority(), bundle);
```

Chapter 90: PorterDuff Mode

PorterDuff is described as a way of combining images as if they were "irregular shaped pieces of cardboard" overlayed on each other, as well as a scheme for blending the overlapping parts

Section 90.1: Creating a PorterDuff ColorFilter

`PorterDuff.Mode` is used to create a `PorterDuffColorFilter`. A color filter modifies the color of each pixel of a visual resource.

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

The above filter will tint the non-transparent pixels to blue color.

The color filter can be applied to a `Drawable`:

```
drawable.setColorFilter(filter);
```

It can be applied to an `ImageView`:

```
imageView.setColorFilter(filter);
```

Also, it can be applied to a `Paint`, so that the color that is drawn using that paint, is modified by the filter:

```
paint.setColorFilter(filter);
```

Section 90.2: Creating a PorterDuff XferMode

An `Xfermode` (think "transfer" mode) works as a transfer step in drawing pipeline. When an `Xfermode` is applied to a `Paint`, the pixels drawn with the paint are combined with underlying pixels (already drawn) as per the mode:

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

Now we have a blue tint paint. Any shape drawn will tint the already existing, non-transparent pixels blue in the area of the shape.

Section 90.3: Apply a radial mask (vignette) to a bitmap using PorterDuffXfermode

```
/**
 * Apply a radial mask (vignette, i.e. fading to black at the borders) to a bitmap
 * @param imageToApplyMaskTo Bitmap to modify
 */
public static void radialMask(final Bitmap imageToApplyMaskTo) {
    Canvas canvas = new Canvas(imageToApplyMaskTo);

    final float centerX = imageToApplyMaskTo.getWidth() * 0.5f;
    final float centerY = imageToApplyMaskTo.getHeight() * 0.5f;
    final float radius = imageToApplyMaskTo.getHeight() * 0.7f;

    RadialGradient gradient = new RadialGradient(centerX, centerY, radius,
            0x00000000, 0xFF000000, android.graphics.Shader.TileMode.CLAMP);
```

```
    Paint p = new Paint();
    p.setShader(gradient);
    p.setColor(0xFF000000);
    p.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.DST_OUT));
    canvas.drawRect(0, 0, imageToApplyMaskTo.getWidth(), imageToApplyMaskTo.getHeight(), p);
}
```

Chapter 91: Menu

Parameter	Description
`inflate(int menuRes, Menu menu)`	Inflate a menu hierarchy from the specified XML resource.
`getMenuInflater ()`	Returns a `MenuInflater` with this context.
`onCreateOptionsMenu (Menu menu)`	Initialize the contents of the Activity's standard options menu. You should place your menu items in to menu.
`onOptionsItemSelected (MenuItem item)`	This method is called whenever an item in your options menu is selected

Section 91.1: Options menu with dividers

In Android there is a default options menu, which can take a number of options. If a larger number of options needs to be displayed, then it makes sense to group those options in order to maintain clarity. Options can be grouped by putting dividers (i.e. horizontal lines) between them. In order to allow for dividers, the following theme can be used:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:dropDownListViewStyle">@style/PopupMenuListView</item>
</style>
<style name="PopupMenuListView" parent="@style/Widget.AppCompat.ListView.DropDown">
    <item name="android:divider">@color/black</item>
    <item name="android:dividerHeight">1dp</item>
</style>
```

By changing the theme, dividers can be added to a menu.

Section 91.2: Apply custom font to Menu

```
public static void applyFontToMenu(Menu m, Context mContext){
    for(int i=0;i<m.size();i++) {
        applyFontToMenuItem(m.getItem(i),mContext);
    }
}
public static void applyFontToMenuItem(MenuItem mi, Context mContext) {
    if(mi.hasSubMenu())
        for(int i=0;i<mi.getSubMenu().size();i++) {
            applyFontToMenuItem(mi.getSubMenu().getItem(i),mContext);
        }
    Typeface font = Typeface.createFromAsset(mContext.getAssets(), "fonts/yourCustomFont.ttf");
    SpannableString mNewTitle = new SpannableString(mi.getTitle());
    mNewTitle.setSpan(new CustomTypefaceSpan("", font, mContext), 0, mNewTitle.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);
    mi.setTitle(mNewTitle);
}
```

and then in the Activity:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
```

```
    return true;
}
```

Section 91.3: Creating a Menu in an Activity

To define your own menu, create an XML file inside your project's `res/menu/` directory and build the menu with the following elements:

- **`<menu>`** : Defines a Menu, which holds all the menu items.
- **`<item>`** : Creates a MenuItem, which represents a single item in a menu. We can also create a nested element in order to create a submenu.

Step 1:
Create your own xml file as the following:

In `res/menu/main_menu.xml`:

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
       android:id="@+id/aboutMenu"
       android:title="About" />
    <item
       android:id="@+id/helpMenu"
       android:title="Help" />
    <item
       android:id="@+id/signOutMenu"
       android:title="Sign Out" />
</menu>
```

Step 2:

To specify the options menu, override `onCreateOptionsMenu()` in your *activity*.

In this method, you can inflate your menu resource (defined in your XML file i.e., `res/menu/main_menu.xml`)

```
 @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
```

When the user selects an item from the options menu, the system calls your *activity's* overridden `onOptionsItemSelected()` method.

- This method passes the MenuItem selected.
- You can identify the item by calling `getItemId()`, which returns the unique ID for the menu item (defined by the `android:id attribute` in the menu resource - `res/menu/main_menu.xml`)*/

```
 @Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.aboutMenu:
            Log.d(TAG, "Clicked on About!");
```

```
            // Code for About goes here
            return true;
        case R.id.helpMenu:
            Log.d(TAG, "Clicked on Help!");
            // Code for Help goes here
            return true;
        case R.id.signOutMenu:
            Log.d(TAG, "Clicked on Sign Out!");
            // SignOut method call goes here
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Wrapping up!

Your `Activity` code should look like below:

```
public class MainActivity extends AppCompatActivity {

    private static final String TAG = "mytag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.aboutMenu:
                Log.d(TAG, "Clicked on About!");
                // Code for About goes here
                return true;
            case R.id.helpMenu:
                Log.d(TAG, "Clicked on Help!");
                // Code for Help goes here
                return true;
            case R.id.signOutMenu:
                Log.d(TAG, "User signed out");
                // SignOut method call goes here
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
}
```

Screenshot of how your own Menu looks:

Chapter 92: Picasso

Picasso is an image library for Android. It's created and maintained by Square. It simplifies the process of displaying images from external locations. The library handles every stage of the process, from the initial HTTP request to the caching of the image. In many cases, only a few lines of code are required to implement this neat library.

Section 92.1: Adding Picasso Library to your Android Project

From the official documentation:

Gradle.

```
dependencies {
 compile "com.squareup.picasso:picasso:2.5.2"
}
```

Maven:

```
<dependency>
  <groupId>com.squareup.picasso</groupId>
  <artifactId>picasso</artifactId>
  <version>2.5.2</version>
</dependency>
```

Section 92.2: Circular Avatars with Picasso

Here is an example Picasso Circle Transform class based on the original, with the addition of a thin border, and also includes functionality for an optional separator for stacking:

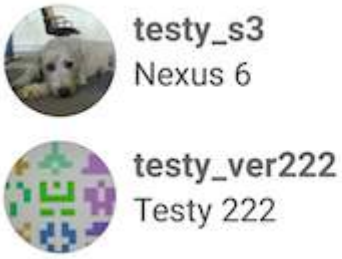
```
import android.graphics.Bitmap;
import android.graphics.BitmapShader;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;

import com.squareup.picasso.Transformation;

public class CircleTransform implements Transformation {

    boolean mCircleSeparator = false;

    public CircleTransform(){
    }

    public CircleTransform(boolean circleSeparator){
        mCircleSeparator = circleSeparator;
    }

    @Override
    public Bitmap transform(Bitmap source) {
        int size = Math.min(source.getWidth(), source.getHeight());

        int x = (source.getWidth() - size) / 2;
        int y = (source.getHeight() - size) / 2;

        Bitmap squaredBitmap = Bitmap.createBitmap(source, x, y, size, size);

        if (squaredBitmap != source) {
            source.recycle();
```

```
        }

        Bitmap bitmap = Bitmap.createBitmap(size, size, source.getConfig());

        Canvas canvas = new Canvas(bitmap);
        BitmapShader shader = new BitmapShader(squaredBitmap, BitmapShader.TileMode.CLAMP,
BitmapShader.TileMode.CLAMP);
        Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG | Paint.DITHER_FLAG |
Paint.FILTER_BITMAP_FLAG);
        paint.setShader(shader);

        float r = size/2f;
        canvas.drawCircle(r, r, r-1, paint);

        // Make the thin border:
        Paint paintBorder = new Paint();
        paintBorder.setStyle(Style.STROKE);
        paintBorder.setColor(Color.argb(84,0,0,0));
        paintBorder.setAntiAlias(true);
        paintBorder.setStrokeWidth(1);
        canvas.drawCircle(r, r, r-1, paintBorder);

        // Optional separator for stacking:
        if (mCircleSeparator) {
            Paint paintBorderSeparator = new Paint();
            paintBorderSeparator.setStyle(Style.STROKE);
            paintBorderSeparator.setColor(Color.parseColor("#ffffff"));
            paintBorderSeparator.setAntiAlias(true);
            paintBorderSeparator.setStrokeWidth(4);
            canvas.drawCircle(r, r, r+1, paintBorderSeparator);
        }

        squaredBitmap.recycle();
        return bitmap;
    }

    @Override
    public String key() {
        return "circle";
    }
}
```

Here is how to use it when loading an image (assuming **`this`** is an Activity Context, and `url` is a String with the url of the image to load):

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform())
    .into(ivAvatar);
```

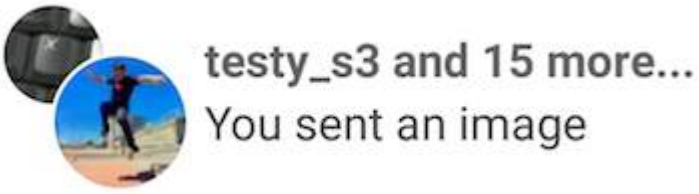
Result:



For use with the separator, give **true** to the constructor for the top image:

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform(true))
    .into(ivAvatar);
```

Result (two ImageViews in a FrameLayout):



Section 92.3: Placeholder and Error Handling

Picasso supports both download and error placeholders as optional features. Its also provides callbacks for handling the download result.

```
Picasso.with(context)
  .load("YOUR IMAGE URL HERE")
  .placeholder(Your Drawable Resource)   //this is optional the image to display while the url image
is downloading
  .error(Your Drawable Resource)         //this is also optional if some error has occurred in
downloading the image this image would be displayed
  .into(imageView, new Callback(){
     @Override
        public void onSuccess() {}

        @Override
        public void onError() {}
   });
```

A request will be retried three times before the error placeholder is shown.

Section 92.4: Re-sizing and Rotating

```
Picasso.with(context)
 .load("YOUR IMAGE URL HERE")
 .placeholder(DRAWABLE RESOURCE)   // optional
 .error(DRAWABLE RESOURCE)         // optional
 .resize(width, height)            // optional
 .rotate(degree)                   // optional
 .into(imageView);
```

Section 92.5: Disable cache in Picasso

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

Section 92.6: Using Picasso as ImageGetter for Html.fromHtml

Using Picasso as ImageGetter for Html.fromHtml

```
public class PicassoImageGetter implements Html.ImageGetter {

private TextView textView;

private Picasso picasso;

public PicassoImageGetter(@NonNull Picasso picasso, @NonNull TextView textView) {
    this.picasso = picasso;
    this.textView = textView;
}

@Override
public Drawable getDrawable(String source) {
    Log.d(PicassoImageGetter.class.getName(), "Start loading url " + source);

    BitmapDrawablePlaceHolder drawable = new BitmapDrawablePlaceHolder();

    picasso
            .load(source)
            .error(R.drawable.connection_error)
            .into(drawable);

    return drawable;
}

private class BitmapDrawablePlaceHolder extends BitmapDrawable implements Target {

    protected Drawable drawable;

    @Override
    public void draw(final Canvas canvas) {
        if (drawable != null) {
            checkBounds();
            drawable.draw(canvas);
        }
    }

    public void setDrawable(@Nullable Drawable drawable) {
        if (drawable != null) {
            this.drawable = drawable;
            checkBounds();
        }
    }

    private void checkBounds() {
        float defaultProportion = (float) drawable.getIntrinsicWidth() / (float)
drawable.getIntrinsicHeight();
```

```
        int width = Math.min(textView.getWidth(), drawable.getIntrinsicWidth());
        int height = (int) ((float) width / defaultProportion);

        if (getBounds().right != textView.getWidth() || getBounds().bottom != height) {

            setBounds(0, 0, textView.getWidth(), height); //set to full width

            int halfOfPlaceHolderWidth = (int) ((float) getBounds().right / 2f);
            int halfOfImageWidth = (int) ((float) width / 2f);

            drawable.setBounds(
                    halfOfPlaceHolderWidth - halfOfImageWidth, //centering an image
                    0,
                    halfOfPlaceHolderWidth + halfOfImageWidth,
                    height);

            textView.setText(textView.getText()); //refresh text
        }
    }

    //------------------------------------------------------------------//

    @Override
    public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
        setDrawable(new BitmapDrawable(Application.getContext().getResources(), bitmap));
    }

    @Override
    public void onBitmapFailed(Drawable errorDrawable) {
        setDrawable(errorDrawable);
    }

    @Override
    public void onPrepareLoad(Drawable placeHolderDrawable) {
        setDrawable(placeHolderDrawable);
    }

    //------------------------------------------------------------------//

}
}
```

The usage is simple:

```
Html.fromHtml(textToParse, new PicassoImageGetter(picasso, textViewTarget), null);
```

Section 92.7: Cancelling Image Requests using Picasso

In certain cases we need to cancel an image download request in Picasso before the download has completed.

This could happen for various reasons, for example if the parent view transitioned to some other view before the image download could be completed.

In this case, you can cancel the image download request using the `cancelRequest()` method:

```
ImageView imageView;

//......
```

```
Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

Section 92.8: Loading Image from external Storage

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
    .load(new File(imagePath))
    .into(imageView);
```

Section 92.9: Downloading image as Bitmap using Picasso

If you want to Download image as `Bitmap` using `Picasso` following code will help you:

```
Picasso.with(mContext)
        .load(ImageUrl)
        .into(new Target() {
            @Override
            public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
               // Todo: Do something with your bitmap here
            }

            @Override
            public void onBitmapFailed(Drawable errorDrawable) {
            }

            @Override
            public void onPrepareLoad(Drawable placeHolderDrawable) {
            }
        });
```

Section 92.10: Try offline disk cache first, then go online and fetch the image

first add the OkHttp to the gradle build file of the app module

```
compile 'com.squareup.picasso:picasso:2.5.2'
compile 'com.squareup.okhttp:okhttp:2.4.0'
compile 'com.jakewharton.picasso:picasso2-okhttp3-downloader:1.0.2'
```

Then make a class extending Application

```
import android.app.Application;

import com.squareup.picasso.OkHttpDownloader;
import com.squareup.picasso.Picasso;

public class Global extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        Picasso.Builder builder = new Picasso.Builder(this);
        builder.downloader(new OkHttpDownloader(this,Integer.MAX_VALUE));
        Picasso built = builder.build();
        built.setIndicatorsEnabled(true);
        built.setLoggingEnabled(true);
```

```
        Picasso.setSingletonInstance(built);

    }
}
```

add it to the Manifest file as follows :

```
<application
       android:name=".Global"
       .. >

</application>
```

Normal Usage

```
Picasso.with(getActivity())
.load(imageUrl)
.networkPolicy(NetworkPolicy.OFFLINE)
.into(imageView, new Callback() {
    @Override
    public void onSuccess() {
        //Offline Cache hit
    }

    @Override
    public void onError() {
        //Try again online if cache failed
        Picasso.with(getActivity())
                .load(imageUrl)
                .error(R.drawable.header)
                .into(imageView, new Callback() {
            @Override
            public void onSuccess() {
                 //Online download
            }

            @Override
            public void onError() {
                Log.v("Picasso","Could not fetch image");
            }
        });
    }
});
```

Link to original answer

Chapter 93: RoboGuice

Section 93.1: Simple example

RoboGuice is a framework that brings the simplicity and ease of Dependency Injection to Android, using Google's own Guice library.

```
@ContentView(R.layout.main)
class RoboWay extends RoboActivity {
    @InjectView(R.id.name)              TextView name;
    @InjectView(R.id.thumbnail)         ImageView thumbnail;
    @InjectResource(R.drawable.icon)    Drawable icon;
    @InjectResource(R.string.app_name) String myName;
    @Inject                             LocationManager loc;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        name.setText( "Hello, " + myName );
    }
}
```

Section 93.2: Installation for Gradle Projects

Add the following pom to the dependencies section of your gradle build file :

```
project.dependencies {
    compile 'org.roboguice:roboguice:3.+'
    provided 'org.roboguice:roboblender:3.+'
}
```

Section 93.3: @ContentView annotation

The @ContentView annotation can be used to further alleviate development of activities and replace the setContentView statement :

```
@ContentView(R.layout.myactivity_layout)
public class MyActivity extends RoboActivity {
    @InjectView(R.id.text1) TextView textView;

    @Override
    protected void onCreate( Bundle savedState ) {
        textView.setText("Hello!");
    }
}
```

Section 93.4: @InjectResource annotation

You can inject any type of resource, Strings, Animations, Drawables, etc.

To inject your first resource into an activity, you'll need to:

- Inherit from RoboActivity
- Annotate your resources with @InjectResource

Example

```
@InjectResource(R.string.app_name) String name;

@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;

@InjectResource(R.anim.my_animation) Animation myAnimation;
```

Section 93.5: @InjectView annotation

You can inject any view using the @InjectView annotation:

You'll need to:

- Inherit from RoboActivity
- Set your content view
- Annotate your views with @InjectView

Example

```
@InjectView(R.id.textView1) TextView textView1;

@InjectView(R.id.textView2) TextView textView2;

@InjectView(R.id.imageView1) ImageView imageView1;
```

Section 93.6: Introduction to RoboGuice

`RoboGuice` is a framework that brings the simplicity and ease of Dependency Injection to Android, using Google's own Guice library.

RoboGuice 3 slims down your application code. Less code means fewer opportunities for bugs. It also makes your code easier to follow -- no longer is your code littered with the mechanics of the Android platform, but now it can focus on the actual logic unique to your application.

To give you an idea, take a look at this simple example of a typical Android `Activity`:

```
class AndroidWay extends Activity {
        TextView name;
        ImageView thumbnail;
        LocationManager loc;
        Drawable icon;
        String myName;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            name      = (TextView) findViewById(R.id.name);
            thumbnail = (ImageView) findViewById(R.id.thumbnail);
            loc       = (LocationManager) getSystemService(Activity.LOCATION_SERVICE);
            icon      = getResources().getDrawable(R.drawable.icon);
            myName    = getString(R.string.app_name);
            name.setText( "Hello, " + myName );
        }
    }
```

This example is 19 lines of code. If you're trying to read through `onCreate()`, you have to skip over 5 lines of boilerplate initialization to find the only one that really matters: `name.setText()`. And complex activities can end up with a lot more of this sort of initialization code.

Compare this to the same app, written using `RoboGuice`:

```
@ContentView(R.layout.main)
    class RoboWay extends RoboActivity {
        @InjectView(R.id.name)             TextView name;
        @InjectView(R.id.thumbnail)        ImageView thumbnail;
        @InjectResource(R.drawable.icon)   Drawable icon;
        @InjectResource(R.string.app_name) String myName;
        @Inject                            LocationManager loc;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            name.setText( "Hello, " + myName );
        }
    }
```

RoboGuice's goal is to make your code be about your app, rather than be about all the initialization and lifecycle code you typically have to maintain in Android.

Annotations:

@ContentView annotation:

The @ContentView annotation can be used to further alleviate development of activities and replace the setContentView statement :

```
@ContentView(R.layout.myactivity_layout)
    public class MyActivity extends RoboActivity {
        @InjectView(R.id.text1) TextView textView;

        @Override
        protected void onCreate( Bundle savedState ) {
            textView.setText("Hello!");
        }
    }
```

@InjectResource annotation:

First you need an Activity that inherits from RoboActivity. Then, assuming that you have an animation my_animation.xml in your res/anim folder, you can now reference it with an annotation:

```
public class MyActivity extends RoboActivity {
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // the rest of your code
}
```

@Inject annotation:

You make sure your activity extends from RoboActivity and annotate your System service member with @Inject. Roboguice will do the rest.

```
class MyActivity extends RoboActivity {
    @Inject Vibrator vibrator;
    @Inject NotificationManager notificationManager;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        // we can use the instances directly!
        vibrator.vibrate(1000L); // RoboGuice took care of the getSystemService(VIBRATOR_SERVICE)
        notificationManager.cancelAll();
```

In addition to Views, Resources, Services, and other android-specific things, RoboGuice can inject Plain Old Java Objects. By default Roboguice will call a no argument constructor on your POJO

```
class MyActivity extends RoboActivity {
    @Inject Foo foo; // this will basically call new Foo();
}
```

Chapter 94: ACRA

Parameter	Description
@ReportCrashes	Defines the ACRA settings such as where it is to be reported, custom content, etc
formUri	the path to the file that reports the crash

Section 94.1: ACRAHandler

Example Application-extending class for handling the reporting:

```
@ReportsCrashes(

        formUri = "https://backend-of-your-choice.com/",//Non-password protected.
        customReportContent = { /* */ReportField.APP_VERSION_NAME,
ReportField.PACKAGE_NAME,ReportField.ANDROID_VERSION, ReportField.PHONE_MODEL,ReportField.LOGCAT },
        mode = ReportingInteractionMode.TOAST,
        resToastText = R.string.crash

)
public class ACRAHandler extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);

        final ACRAConfiguration config = new ConfigurationBuilder(this)

                .build();

        // Initialise ACRA
        ACRA.init(this, config);


    }

}
```

Section 94.2: Example manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   <!-- etc -->

>



    <!-- Internet is required. READ_LOGS are to ensure that the Logcat is transmitted-->
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_LOGS"/>

    <application
       android:allowBackup="true"
       android:name=".ACRAHandler"<!-- Activates ACRA on startup -->
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

```
        <!-- Activities -->
    </application>

</manifest>
```

Section 94.3: Installation

Maven

```
<dependency>
    <groupId>ch.acra</groupId>
    <artifactId>acra</artifactId>
    <version>4.9.2</version>
    <type>aar</type>
</dependency>
```

Gradle

```
compile 'ch.acra:acra:4.9.2'
```

Chapter 95: Parcelable

Parcelable is an Android specific interface where you implement the serialization yourself. It was created to be far more efficient that Serializable, and to get around some problems with the default Java serialization scheme.

Section 95.1: Making a custom object Parcelable

```
/**
 * Created by Alex Sullivan on 7/21/16.
 */
public class Foo implements Parcelable
{
    private final int myFirstVariable;
    private final String mySecondVariable;
    private final long myThirdVariable;

    public Foo(int myFirstVariable, String mySecondVariable, long myThirdVariable)
    {
        this.myFirstVariable = myFirstVariable;
        this.mySecondVariable = mySecondVariable;
        this.myThirdVariable = myThirdVariable;
    }

    // Note that you MUST read values from the parcel IN THE SAME ORDER that
    // values were WRITTEN to the parcel! This method is our own custom method
    // to instantiate our object from a Parcel. It is used in the Parcelable.Creator variable we
declare below.
    public Foo(Parcel in)
    {
        this.myFirstVariable = in.readInt();
        this.mySecondVariable = in.readString();
        this.myThirdVariable = in.readLong();
    }

    // The describe contents method can normally return 0. It's used when
    // the parceled object includes a file descriptor.
    @Override
    public int describeContents()
    {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags)
    {
        dest.writeInt(myFirstVariable);
        dest.writeString(mySecondVariable);
        dest.writeLong(myThirdVariable);
    }

    // Note that this seemingly random field IS NOT OPTIONAL. The system will
    // look for this variable using reflection in order to instantiate your
    // parceled object when read from an Intent.
    public static final Parcelable.Creator<Foo> CREATOR = new Parcelable.Creator<Foo>()
    {
        // This method is used to actually instantiate our custom object
        // from the Parcel. Convention dictates we make a new constructor that
        // takes the parcel in as its only argument.
        public Foo createFromParcel(Parcel in)
        {
```

```
            return new Foo(in);
        }

        // This method is used to make an array of your custom object.
        // Declaring a new array with the provided size is usually enough.
        public Foo[] newArray(int size)
        {
            return new Foo[size];
        }
    };
}
```

Section 95.2: Parcelable object containing another Parcelable object

An example of a class that contains a parcelable class inside:

```
public class Repository implements Parcelable {
    private String name;
    private Owner owner;
    private boolean isPrivate;

    public Repository(String name, Owner owner, boolean isPrivate) {
        this.name = name;
        this.owner = owner;
        this.isPrivate = isPrivate;
    }

    protected Repository(Parcel in) {
        name = in.readString();
        owner = in.readParcelable(Owner.class.getClassLoader());
        isPrivate = in.readByte() != 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeParcelable(owner, flags);
        dest.writeByte((byte) (isPrivate ? 1 : 0));
    }

    @Override
    public int describeContents() {
        return 0;
    }

    public static final Creator<Repository> CREATOR = new Creator<Repository>() {
        @Override
        public Repository createFromParcel(Parcel in) {
            return new Repository(in);
        }

        @Override
        public Repository[] newArray(int size) {
            return new Repository[size];
        }
    };

    //getters and setters

    public String getName() {
```

```
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Owner getOwner() {
        return owner;
    }

    public void setOwner(Owner owner) {
        this.owner = owner;
    }

     public boolean isPrivate() {
        return isPrivate;
    }

    public void setPrivate(boolean isPrivate) {
        this.isPrivate = isPrivate;
    }
}
```

Owner is just a normal parcelable class.

Section 95.3: Using Enums with Parcelable

```
/**
 * Created by Nick Cardoso on 03/08/16.
 * This is not a complete parcelable implementation, it only highlights the easiest
 * way to read and write your Enum values to your parcel
 */
public class Foo implements Parcelable {

    private final MyEnum myEnumVariable;
    private final MyEnum mySaferEnumVariableExample;

    public Foo(Parcel in) {

        //the simplest way
        myEnumVariable = MyEnum.valueOf( in.readString() );

        //with some error checking
        try {
            mySaferEnumVariableExample= MyEnum.valueOf( in.readString() );
        } catch (IllegalArgumentException e) { //bad string or null value
            mySaferEnumVariableExample= MyEnum.DEFAULT;
        }

    }

    ...

    @Override
    public void writeToParcel(Parcel dest, int flags) {

        //the simple way
        dest.writeString(myEnumVariable.name());

        //avoiding NPEs with some error checking
```

```
        dest.writeString(mySaferEnumVariableExample == null? null :
mySaferEnumVariableExample.name());

    }

}

public enum MyEnum {
    VALUE_1,
    VALUE_2,
    DEFAULT
}
```

This is preferable to (for example) using an ordinal, because inserting new values into your enum will not affect previously stored values

Chapter 96: Retrofit2

The official Retrofit page describes itself as

A type-safe REST client for Android and Java.

Retrofit turns your REST API into a Java interface. It uses annotations to describe HTTP requests, URL parameter replacement and query parameter support is integrated by default. Additionally, it provides functionality for multipart request body and file uploads.

Section 96.1: A Simple GET Request

We are going to be showing how to make a `GET` request to an API that responds with a `JSON` object or a `JSON` array. The first thing we need to do is add the Retrofit and `GSON` Converter dependencies to our module's gradle file.

Add the dependencies for retrofit library as described in the Remarks section.

Example of expected JSON object:

```
{
    "deviceId": "56V56C14SF5B4SF",
    "name": "Steven",
    "eventCount": 0
}
```

Example of JSON array:

```
[
    {
        "deviceId": "56V56C14SF5B4SF",
        "name": "Steven",
        "eventCount": 0
    },
    {
        "deviceId": "35A80SF3QDV7M9F",
        "name": "John",
        "eventCount": 2
    }
]
```

Example of corresponding model class:

```
public class Device
{
    @SerializedName("deviceId")
    public String id;

    @SerializedName("name")
    public String name;

    @SerializedName("eventCount")
    public int eventCount;
}
```

The `@SerializedName` annotations here are from the `GSON` library and allows us to `serialize` and `deserialize` this class to `JSON` using the serialized name as the keys. Now we can build the interface for the API that will actually

fetch the data from the server.

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

There's a lot going on here in a pretty compact space so let's break it down:

- The `@GET` annotation comes from Retrofit and tells the library that we're defining a GET request.
- The path in the parentheses is the endpoint that our GET request should hit (we'll set the base url a little later).
- The curly-brackets allow us to replace parts of the path at run time so we can pass arguments.
- The function we're defining is called `getDevice` and takes the device id we want as an argument.
- The `@PATH` annotation tells Retrofit that this argument should replace the "deviceId" placeholder in the path.
- The function returns a `Call` object of type `Device`.

Creating a wrapper class:

Now we will make a little wrapper class for our API to keep the Retrofit initialization code wrapped up nicely.

```
public class DeviceAPIHelper
{
    public final DeviceAPI api;

    private DeviceAPIHelper ()
    {

        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("http://example.com/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();

        api = retrofit.create(DeviceAPI.class);
    }
}
```

This class creates a GSON instance to be able to parse the JSON response, creates a Retrofit instance with our base url and a GSONConverter and then creates an instance of our API.

Calling the API:

```
// Getting a JSON object
Call<Device> callObject = api.getDevice(deviceID);
callObject.enqueue(new Callback<Response<Device>>()
{
    @Override
    public void onResponse (Call<Device> call, Response<Device> response)
    {
        if (response.isSuccessful())
        {
           Device device = response.body();
        }
    }
```

```
    @Override
    public void onFailure (Call<Device> call, Throwable t)
    {
        Log.e(TAG, t.getLocalizedMessage());
    }
});

// Getting a JSON array
Call<List<Device>> callArray = api.getDevices();
callArray.enqueue(new Callback<Response<List<Device>>()
{
    @Override
    public void onResponse (Call<List<Device>> call, Response<List<Device>> response)
    {
        if (response.isSuccessful())
        {
           List<Device> devices = response.body();
        }
    }

    @Override
    public void onFailure (Call<List<Device>> call, Throwable t)
    {
        Log.e(TAG, t.getLocalizedMessage());
    }
});
```

This uses our API interface to create a `Call<Device>` object and to create a `Call<List<Device>>` respectively. Calling `enqueue` tells Retrofit to make that call on a background thread and return the result to the callback that we're creating here.

Note: Parsing a JSON array of primitive objects (like *String, Integer, Boolean*, and *Double*) is similar to parsing a JSON array. However, you don't need your own model class. You can get the array of Strings for example by having the return type of the call as `Call<List<String>>`.

Section 96.2: Debugging with Stetho

Add the following dependencies to your application.

```
compile 'com.facebook.stetho:stetho:1.5.0'
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

In your Application class' `onCreate` method, call the following.

```
Stetho.initializeWithDefaults(this);
```

When creating your `Retrofit` instance, create a custom OkHttp instance.

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

Then set this custom OkHttp instance in the Retrofit instance.

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

Now connect your phone to your computer, launch the app, and type `chrome://inspect` into your Chrome browser. Retrofit network calls should now show up for you to inspect.

Section 96.3: Add logging to Retrofit2

Retrofit requests can be logged using an intercepter. There are several levels of detail available: NONE, BASIC, HEADERS, BODY. See Github project here.

1. Add dependency to build.gradle:

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

2. Add logging interceptor when creating Retrofit:

```
HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
        .addInterceptor(loggingInterceptor)
        .build();
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://example.com/")
        .client(okHttpClient)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build();
```

Exposing the logs in the Terminal(Android Monitor) is something that should be avoided in the release version as it may lead to unwanted exposing of critical information such as Auth Tokens etc.

To avoid the logs being exposed in the run time, check the following condition

```
if(BuildConfig.DEBUG){
     //your interfector code here
    }
```

For example:

```
HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
if(BuildConfig.DEBUG){
     //print the logs in this case
    loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
}else{
    loggingInterceptor.setLevel(LoggingInterceptor.Level.NONE);
}

OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
      .addInterceptor(loggingInterceptor)
      .build();

Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com/")
      .client(okHttpClient)
      .addConverterFactory(GsonConverterFactory.create(gson))
      .build();
```

Section 96.4: A simple POST request with GSON

Sample JSON:

```
{
    "id": "12345",
    "type": "android"
}
```

Define your request:

```
public class GetDeviceRequest {

    @SerializedName("deviceId")
    private String mDeviceId;

    public GetDeviceRequest(String deviceId) {
        this.mDeviceId = deviceId;
    }

    public String getDeviceId() {
        return mDeviceId;
    }

}
```

Define your service (endpoints to hit):

```
public interface Service {

    @POST("device")
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);

}
```

Define your singleton instance of the network client:

```
public class RestClient {

    private static Service REST_CLIENT;

    static {
        setupRestClient();
    }

    private static void setupRestClient() {

        // Define gson
        Gson gson = new Gson();

        // Define our client
        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("http://example.com/")
                .addConverterFactory(GsonConverterFactory.create(gson))
                .build();

        REST_CLIENT = retrofit.create(Service.class);
    }

    public static Retrofit getRestClient() {
        return REST_CLIENT;
    }
```

```
}
```

Define a simple model object for the device:

```
public class Device {

    @SerializedName("id")
    private String mId;

    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }

    public String getType() {
        return mType;
    }

}
```

Define controller to handle the requests for the device

```
public class DeviceController {

    // Other initialization code here...

    public void getDeviceFromAPI() {

        // Define our request and enqueue
        Call<Device> call = RestClient.getRestClient().getDevice(new GetDeviceRequest("12345"));

        // Go ahead and enqueue the request
        call.enqueue(new Callback<Device>() {
            @Override
            public void onSuccess(Response<Device> deviceResponse) {
                // Take care of your device here
                if (deviceResponse.isSuccess()) {
                    // Handle success
                    //delegate.passDeviceObject();
                }
            }

            @Override
            public void onFailure(Throwable t) {
                // Go ahead and handle the error here
            }

        });
```

Section 96.5: Download a file from Server using Retrofit2

Interface declaration for downloading a file

```
public interface ApiInterface {
    @GET("movie/now_playing")
    Call<MovieResponse> getNowPlayingMovies(@Query("api_key") String apiKey, @Query("page") int
page);
```

```
    // option 1: a resource relative to your base URL
    @GET("resource/example.zip")
    Call<ResponseBody> downloadFileWithFixedUrl();

    // option 2: using a dynamic URL
    @GET
    Call<ResponseBody> downloadFileWithDynamicUrl(@Url String fileUrl);
}
```

The option 1 is used for downloading a file from Server which is having fixed URL. and option 2 is used to pass a dynamic value as full URL to request call. This can be helpful when downloading files, which are dependent of parameters like user or time.

Setup retrofit for making api calls

```
public class ServiceGenerator {

    public static final String API_BASE_URL = "http://your.api-base.url/";

    private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

    private static Retrofit.Builder builder =
            new Retrofit.Builder()
            .baseUrl(API_BASE_URL)
            .addConverterFactory(GsonConverterFactory.create());

    public static <S> S createService(Class<S> serviceClass){
        Retrofit retrofit = builder.client(httpClient.build()).build();
        return retrofit.create(serviceClass);
    }

}
```

Now, make implementation of api for downloading file from server

```
private void downloadFile(){
        ApiInterface apiInterface = ServiceGenerator.createService(ApiInterface.class);

        Call<ResponseBody> call = apiInterface.downloadFileWithFixedUrl();

        call.enqueue(new Callback<ResponseBody>() {
            @Override
            public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
                if (response.isSuccessful()){
                    boolean writtenToDisk = writeResponseBodyToDisk(response.body());

                    Log.d("File download was a success? ", String.valueOf(writtenToDisk));
                }
            }

            @Override
            public void onFailure(Call<ResponseBody> call, Throwable t) {

            }
        });
    }
```

And after getting response in the callback, code some standard IO for saving file to disk. Here is the code:

```
private boolean writeResponseBodyToDisk(ResponseBody body) {
```

```
        try {
            // todo change the file location/name according to your needs
            File futureStudioIconFile = new File(getExternalFilesDir(null) + File.separator +
"Future Studio Icon.png");

            InputStream inputStream = null;
            OutputStream outputStream = null;

            try {
                byte[] fileReader = new byte[4096];

                long fileSize = body.contentLength();
                long fileSizeDownloaded = 0;

                inputStream = body.byteStream();
                outputStream = new FileOutputStream(futureStudioIconFile);

                while (true) {
                    int read = inputStream.read(fileReader);

                    if (read == -1) {
                        break;
                    }

                    outputStream.write(fileReader, 0, read);

                    fileSizeDownloaded += read;

                    Log.d("File Download: " , fileSizeDownloaded + " of " + fileSize);
                }

                outputStream.flush();

                return true;
            } catch (IOException e) {
                return false;
            } finally {
                if (inputStream != null) {
                    inputStream.close();
                }

                if (outputStream != null) {
                    outputStream.close();
                }
            }
        } catch (IOException e) {
            return false;
        }
    }
```

Note we have specified **ResponseBody** as return type, otherwise Retrofit will try to parse and convert it, which doesn't make sense when you are downloading file.

If you want more on Retrofit stuffs, got to this link as it is very useful. [1]:
https://futurestud.io/blog/retrofit-getting-started-and-android-client

Section 96.6: Upload multiple file using Retrofit as multipart

Once you have setup the Retrofit environment in your project, you can use the following example that demonstrates how to upload multiple files using Retrofit:

```
private void mulipleFileUploadFile(Uri[] fileUri) {
    OkHttpClient okHttpClient = new OkHttpClient();
    OkHttpClient clientWith30sTimeout = okHttpClient.newBuilder()
            .readTimeout(30, TimeUnit.SECONDS)
            .build();

    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(API_URL_BASE)
            .addConverterFactory(new MultiPartConverter())
            .client(clientWith30sTimeout)
            .build();

    WebAPIService service = retrofit.create(WebAPIService.class); //here is the interface which you
have created for the call service
    Map<String, okhttp3.RequestBody> maps = new HashMap<>();

    if (fileUri!=null && fileUri.length>0) {
        for (int i = 0; i < fileUri.length; i++) {
            String filePath = getRealPathFromUri(fileUri[i]);
            File file1 = new File(filePath);

            if (filePath != null && filePath.length() > 0) {
                if (file1.exists()) {
                    okhttp3.RequestBody requestFile =
okhttp3.RequestBody.create(okhttp3.MediaType.parse("multipart/form-data"), file1);
                    String filename = "imagePath" + i; //key for upload file like : imagePath0
                    maps.put(filename + "\"; filename=\"" + file1.getName(), requestFile);
                }
            }
        }
    }

    String descriptionString = " string request";//
    //hear is the your json request
    Call<String> call = service.postFile(maps, descriptionString);
    call.enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call,
                               Response<String> response) {
            Log.i(LOG_TAG, "success");
            Log.d("body==>", response.body().toString() + "");
            Log.d("isSuccessful==>", response.isSuccessful() + "");
            Log.d("message==>", response.message() + "");
            Log.d("raw==>", response.raw().toString() + "");
            Log.d("raw().networkResponse()", response.raw().networkResponse().toString() + "");
        }

        @Override
        public void onFailure(Call<String> call, Throwable t) {
            Log.e(LOG_TAG, t.getMessage());
        }
    });
}

public String getRealPathFromUri(final Uri uri) { // function for file path from uri,
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(mContext, uri)) {
        // ExternalStorageProvider
        if (isExternalStorageDocument(uri)) {
            final String docId = DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];
```

```
            if ("primary".equalsIgnoreCase(type)) {
                return Environment.getExternalStorageDirectory() + "/" + split[1];
            }
        }
        // DownloadsProvider
        else if (isDownloadsDocument(uri)) {

            final String id = DocumentsContract.getDocumentId(uri);
            final Uri contentUri = ContentUris.withAppendedId(
                    Uri.parse("content://downloads/public_downloads"), Long.valueOf(id));

            return getDataColumn(mContext, contentUri, null, null);
        }
        // MediaProvider
        else if (isMediaDocument(uri)) {
            final String docId = DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];

            Uri contentUri = null;
            if ("image".equals(type)) {
                contentUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
            } else if ("video".equals(type)) {
                contentUri = MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
            } else if ("audio".equals(type)) {
                contentUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
            }

            final String selection = "_id=?";
            final String[] selectionArgs = new String[]{
                    split[1]
            };

            return getDataColumn(mContext, contentUri, selection, selectionArgs);
        }
    }
    // MediaStore (and general)
    else if ("content".equalsIgnoreCase(uri.getScheme())) {

        // Return the remote address
        if (isGooglePhotosUri(uri))
            return uri.getLastPathSegment();

        return getDataColumn(mContext, uri, null, null);
    }
    // File
    else if ("file".equalsIgnoreCase(uri.getScheme())) {
        return uri.getPath();
    }

    return null;
}
```

Following is the interface

```
public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}
```

Section 96.7: Retrofit with OkHttp interceptor

This example shows how to use a request interceptor with OkHttp. This has numerous use cases such as:

- Adding universal `header` to the request. E.g. authenticating a request
- Debugging networked applications
- Retrieving raw `response`
- Logging network transaction etc.
- Set custom user agent

```
Retrofit.Builder builder = new Retrofit.Builder()
        .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
        .addConverterFactory(GsonConverterFactory.create())
        .baseUrl("https://api.github.com/");

if (!TextUtils.isEmpty(githubToken)) {
    // `githubToken`: Access token for GitHub
    OkHttpClient client = new OkHttpClient.Builder().addInterceptor(new Interceptor() {
        @Override public Response intercept(Chain chain) throws IOException {
            Request request = chain.request();
            Request newReq = request.newBuilder()
                    .addHeader("Authorization", format("token %s", githubToken))
                    .build();
            return chain.proceed(newReq);
        }
    }).build();

    builder.client(client);
}

return builder.build().create(GithubApi.class);
```

See OkHttp topic for more details.

Section 96.8: Header and Body: an Authentication Example

The `@Header` and `@Body` annotations can be placed into the method signatures and Retrofit will automatically create them based on your models.

```
public interface MyService {
     @POST("authentication/user")
     Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
@Header("Authorization") String basicToken);
}
```

AuthenticaionRequest is our model, a POJO, containing the information the server requires. For this example, our server wants the client key and secret.

```
public class AuthenticationRequest {
     String clientKey;
     String clientSecret;
}
```

Notice that in `@Header("Authorization")` we are specifying we are populating the Authorization header. The other headers will be populated automatically since Retrofit can infer what they are based on the type of objects we are sending and expecting in return.

We create our Retrofit service somewhere. We make sure to use HTTPS.

```
Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https:// some example site")
            .client(client)
            .build();
MyService myService = retrofit.create(MyService.class)
```

Then we can use our service.

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

Section 96.9: Uploading a file via Multipart

Declare your interface with Retrofit2 annotations:

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ") RequestBody
photo);
}
```

Where `RestApiDefaultResponse` is a custom class containing the response.

Building the implementation of your API and enqueue the call:

```
Retrofit retrofit = new Retrofit.Builder()
                                .addConverterFactory(GsonConverterFactory.create())
                                .baseUrl("http://<yourhost>/")
                                .client(okHttpClient)
                                .build();

BackendApiClient apiClient = retrofit.create(BackendApiClient.class);
RequestBody reqBody = RequestBody.create(MediaType.parse("image/jpeg"), photoFile);
Call<RestApiDefaultResponse> call = apiClient.uploadPhoto(reqBody);
call.enqueue(<your callback function>);
```

Section 96.10: Retrofit 2 Custom Xml Converter

Adding dependencies into the build.gradle file.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ....
}
```

Then create Converter Factory

```
public class XStreamXmlConverterFactory extends Converter.Factory {

    /** Create an instance using a default {@link com.thoughtworks.xstream.XStream} instance for
conversion. */
    public static XStreamXmlConverterFactory create() {
        return create(new XStream());
    }

    /** Create an instance using {@code xStream} for conversion. */
    public static XStreamXmlConverterFactory create(XStream xStream) {
        return new XStreamXmlConverterFactory(xStream);
    }

    private final XStream xStream;

    private XStreamXmlConverterFactory(XStream xStream) {
        if (xStream == null) throw new NullPointerException("xStream == null");
        this.xStream = xStream;
    }

    @Override
    public Converter<ResponseBody, ?> responseBodyConverter(Type type, Annotation[] annotations,
Retrofit retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        Class<?> cls = (Class<?>) type;

        return new XStreamXmlResponseBodyConverter<>(cls, xStream);
    }

    @Override
    public Converter<?, RequestBody> requestBodyConverter(Type type,
          Annotation[] parameterAnnotations, Annotation[] methodAnnotations, Retrofit retrofit) {

        if (!(type instanceof Class)) {
            return null;
        }

        return new XStreamXmlRequestBodyConverter<>(xStream);
    }
}
```

create a class to handle the body request.

```
final class XStreamXmlResponseBodyConverter <T> implements Converter<ResponseBody, T> {

    private final Class<T> cls;
    private final XStream xStream;

    XStreamXmlResponseBodyConverter(Class<T> cls, XStream xStream) {
        this.cls = cls;
        this.xStream = xStream;
    }

    @Override
    public T convert(ResponseBody value) throws IOException {

        try {
```

```
            this.xStream.processAnnotations(cls);
            Object object =  this.xStream.fromXML(value.byteStream());
            return (T) object;

        }finally {
            value.close();
        }
    }
}
```

create a class to handle the body response.

```
final class XStreamXmlRequestBodyConverter<T> implements Converter<T, RequestBody> {

    private static final MediaType MEDIA_TYPE = MediaType.parse("application/xml; charset=UTF-8");
    private static final String CHARSET = "UTF-8";

    private final XStream xStream;

    XStreamXmlRequestBodyConverter(XStream xStream) {
        this.xStream = xStream;
    }

    @Override
    public RequestBody convert(T value) throws IOException {

        Buffer buffer = new Buffer();

        try {
            OutputStreamWriter osw = new OutputStreamWriter(buffer.outputStream(), CHARSET);
            xStream.toXML(value, osw);
            osw.flush();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        return RequestBody.create(MEDIA_TYPE, buffer.readByteString());
    }
}
```

So, this point we can send and receive any XML , We just need create XStream Annotations for the entities.

Then create a Retrofit instance:

```
XStream xs = new XStream(new DomDriver());
xs.autodetectAnnotations(true);

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .addConverterFactory(XStreamXmlConverterFactory.create(xs))
    .client(client)
    .build();
```

Section 96.11: Reading XML form URL with Retrofit 2

We will use retrofit 2 and SimpleXmlConverter to get xml data from url and parse to Java class.

Add dependency to Gradle script:

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

Create interface

Also create xml class wrapper in our case Rss class

```
    public interface ApiDataInterface{

        // path to xml link on web site

        @GET (data/read.xml)

        Call<Rss> getData();

}
```

Xml read function

```
private void readXmlFeed() {
        try {

            // base url - url of web site
            Retrofit retrofit = new Retrofit.Builder()
                    .baseUrl(http://www.google.com/)
                    .client(new OkHttpClient())
                    .addConverterFactory(SimpleXmlConverterFactory.create())
                    .build();

            ApiDataInterface apiService = retrofit.create(ApiDataInterface.class);

            Call<Rss> call = apiService.getData();
            call.enqueue(new Callback<Rss>() {

                @Override
                public void onResponse(Call<Rss> call, Response<Rss> response) {

                    Log.e("Response success", response.message());

                }

                @Override
                public void onFailure(Call<Rss> call, Throwable t) {
                    Log.e("Response fail", t.getMessage());
                }
            });

        } catch (Exception e) {
            Log.e("Exception", e.getMessage());
        }


    }
```

This is example of Java class with SimpleXML annotations

More about annotations SimpleXmlDocumentation

```
@Root (name = "rss")
```

```
public class Rss
{


    public Rss() {

    }



    public Rss(String title, String description, String link, List<Item> item, String language) {

        this.title = title;
        this.description = description;
        this.link = link;
        this.item = item;
        this.language = language;

    }

    @Element (name = "title")
    private String title;

    @Element(name = "description")
    private String description;

    @Element(name = "link")
    private String link;

    @ElementList (entry="item", inline=true)
    private List<Item> item;

    @Element(name = "language")
    private String language;
```

Chapter 97: ButterKnife

Butterknife is a view binding tool that uses annotations to generate boilerplate code for us. This tool is developed by Jake Wharton at Square and is essentially used to save typing repetitive lines of code like `findViewById(R.id.view)` when dealing with views thus making our code look a lot cleaner.

To be clear, Butterknife is **not a dependency injection library**. Butterknife injects code at compile time. It is very similar to the work done by Android Annotations.

Section 97.1: Configuring ButterKnife in your project

Configure your project-level `build.gradle` to include the `android-apt` plugin:

```
buildscript {
   repositories {
      mavenCentral()
   }

   dependencies {
      classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
   }
}
```

Then, apply the `android-apt` plugin in your module-level `build.gradle` and add the ButterKnife dependencies:

```
apply plugin: 'android-apt'

android {
    ...
}

dependencies {
      compile 'com.jakewharton:butterknife:8.5.1'
      annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}
```

Note: If you are using the new Jack compiler with version 2.2.0 or newer you do not need the `android-apt` plugin and can instead replace apt with `annotationProcessor` when declaring the compiler dependency.

In order to use ButterKnife annotations you shouldn't forget about binding them in `onCreate()` of your Activities or `onCreateView()` of your Fragments:

```
class ExampleActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Binding annotations
        ButterKnife.bind(this);
        // ...
  }

}

// Or
class ExampleFragment extends Fragment {
```

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View view = inflater.inflate(getContentView(), container, false);
        // Binding annotations
        ButterKnife.bind(this, view);
        // ...
        return view;
  }

}
```

Snapshots of the development version are available in Sonatype's snapshots repository.

Below are the additional steps you'd have to take to use ButterKnife in a library project

To use ButterKnife in a library project, add the plugin to your project-level `build.gradle`:

```
buildscript {
    dependencies {
        classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
    }
}
```

...and then apply to your module by adding these lines on the top of your library-level `build.gradle`:

```
apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'
```

Now make sure you use `R2` instead of `R` inside all ButterKnife annotations.

```
class ExampleActivity extends Activity {

    // Bind xml resource to their View
    @BindView(R2.id.user) EditText username;
    @BindView(R2.id.pass) EditText password;

    // Binding resources from drawable,strings,dimens,colors
    @BindString(R.string.choose) String choose;
    @BindDrawable(R.drawable.send) Drawable send;
    @BindColor(R.color.cyan) int cyan;
    @BindDimen(R.dimen.margin) Float generalMargin;

    // Listeners
    @OnClick(R.id.submit)
    public void submit(View view) {
    // TODO submit data to server...
    }

    // bind with butterknife in onCreate
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
        // TODO continue
  }
```

```
}
```

Section 97.2: Unbinding views in ButterKnife

Fragments have a different view lifecycle than activities. When binding a fragment in onCreateView, set the views to null in onDestroyView. Butter Knife returns an Unbinder instance when you call bind to do this for you. Call its unbind method in the appropriate lifecycle callback.

An example:

```
public class MyFragment extends Fragment {
  @BindView(R.id.textView) TextView textView;
  @BindView(R.id.button) Button button;
  private Unbinder unbinder;

  @Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    View view = inflater.inflate(R.layout.my_fragment, container, false);
    unbinder = ButterKnife.bind(this, view);
    // TODO Use fields...
    return view;
  }

  @Override public void onDestroyView() {
    super.onDestroyView();
    unbinder.unbind();
  }
}
```

> Note: Calling unbind() in onDestroyView() is not required, but recommended as it saves quite a bit of memory if your app has a large backstack.

Section 97.3: Binding Listeners using ButterKnife

OnClick Listener:

```
@OnClick(R.id.login)
public void login(View view) {
  // Additional logic
}
```

All arguments to the listener method are optional:

```
@OnClick(R.id.login)
public void login() {
   // Additional logic
}
```

Specific type will be automatically casted:

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
  button.setText("Hello!");
}
```

Multiple IDs in a single binding for common event handling:

```
@OnClick({ R.id.door1, R.id.door2, R.id.door3 })
public void pickDoor(DoorView door) {
  if (door.hasPrizeBehind()) {
    Toast.makeText(this, "You win!", LENGTH_SHORT).show();
  } else {
    Toast.makeText(this, "Try again", LENGTH_SHORT).show();
  }
}
```

Custom Views can bind to their own listeners by not specifying an ID:

```
public class CustomButton extends Button {
  @OnClick
  public void onClick() {
    // TODO
  }
}
```

Section 97.4: Android Studio ButterKnife Plugin

Android ButterKnife Zelezny

> Plugin for generating ButterKnife injections from selected layout XMLs in activities/fragments/adapters.

Note : Make sure that you make the right click for ***your_xml_layou***`(R.layout.your_xml_layou)` else the *Generate menu* will not contain Butterknife injector option.

```java
/**
 * Main UI for setting up GridWichterle.
 *
 * @author Michal Matl (michal.matl@inmite.eu)
 */
public class SettingsActivity extends FragmentActivity {

    private Config mConfig;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        ButterKnife.inject(this);

        Intent intent = new Intent(this, GridOverlayService.class);
        startService(intent);

        setupViews();
    }
```

Link : Jetbrains Plugin Android ButterKnife Zelezny

Section 97.5: Binding Views using ButterKnife

we can annotate fields with `@BindView` and a view ID for Butter Knife to find and automatically cast the corresponding view in our layout.

Binding Views

Binding Views in Activity

```java
class ExampleActivity extends Activity {
  @BindView(R.id.title) TextView title;
  @BindView(R.id.subtitle) TextView subtitle;
  @BindView(R.id.footer) TextView footer;

  @Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.simple_activity);
    ButterKnife.bind(this);
    // TODO Use fields...
  }
}
```

Binding Views in Fragments

```
public class FancyFragment extends Fragment {
  @BindView(R.id.button1) Button button1;
  @BindView(R.id.button2) Button button2;
  private Unbinder unbinder;

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
{
    View view = inflater.inflate(R.layout.fancy_fragment, container, false);
    unbinder = ButterKnife.bind(this, view);
    // TODO Use fields...
    return view;
  }

  // in fragments or non activity bindings we need to unbind the binding when view is about to be
destroyed
  @Override
    public void onDestroy() {
        super.onDestroy();
        unbinder.unbind();
    }
}
```

Binding Views in Dialogs

We can use `ButterKnife.findById` to find views on a View, Activity, or Dialog. It uses generics to infer the return type and automatically performs the cast.

```
View view = LayoutInflater.from(context).inflate(R.layout.thing, null);
TextView firstName = ButterKnife.findById(view, R.id.first_name);
TextView lastName = ButterKnife.findById(view, R.id.last_name);
ImageView photo = ButterKnife.findById(view, R.id.photo);
```

Binding Views in ViewHolder

```
static class ViewHolder {
    @BindView(R.id.title) TextView name;
    @BindView(R.id.job_title) TextView jobTitle;

    public ViewHolder(View view) {
      ButterKnife.bind(this, view);
    }
  }
```

Binding Resources

Apart from being useful for binding views, one could also use ButterKnife to bind resources such as those defined within `strings.xml`, `drawables.xml`, `colors.xml`, `dimens.xml`, etc.

```
public class ExampleActivity extends Activity {

    @BindString(R.string.title) String title;
    @BindDrawable(R.drawable.graphic) Drawable graphic;
    @BindColor(R.color.red) int red; // int or ColorStateList field
    @BindDimen(R.dimen.spacer) Float spacer; // int (for pixel size) or float (for exact value)
field

    @Override
    public void onCreate(Bundle savedInstanceState) {

        // ...
```

```
        ButterKnife.bind(this);
    }

}
```

Binding View Lists

You can group multiple views into a List or array. This is very helpful when we need to perform one action on multiple views at once.

```
@BindViews({ R.id.first_name, R.id.middle_name, R.id.last_name })
List<EditText> nameViews;

//The apply method allows you to act on all the views in a list at once.
ButterKnife.apply(nameViews, DISABLE);
ButterKnife.apply(nameViews, ENABLED, false);


//We can use Action and Setter interfaces allow specifying simple behavior.
static final ButterKnife.Action<View> DISABLE = new ButterKnife.Action<View>() {
  @Override public void apply(View view, int index) {
    view.setEnabled(false);
  }
};
static final ButterKnife.Setter<View, Boolean> ENABLED = new ButterKnife.Setter<View, Boolean>() {
  @Override public void set(View view, Boolean value, int index) {
    view.setEnabled(value);
  }
};
```

Optional Bindings

By default, both `@Bind` and listener bindings are required. An exception is thrown if the target view cannot be found. But if we are not sure if a view will be there or not then we can add a `@Nullable` annotation to fields or the `@Optional` annotation to methods to suppress this behavior and create an optional binding.

```
@Nullable
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;

@Optional
@OnClick(R.id.maybe_missing)
void onMaybeMissingClicked() {
  // TODO ...
}
```

Chapter 98: Volley

Volley is an Android HTTP library that was introduced by Google to make networking calls much simpler. By default all the Volley network calls are made asynchronously, handling everything in a background thread and returning the results in the foreground with use of callbacks. As fetching data over a network is one of the most common tasks that is performed in any app, the Volley library was made to ease Android app development.

Section 98.1: Using Volley for HTTP requests

Add the gradle dependency in app-level build.gradle

```
compile 'com.android.volley:volley:1.0.0'
```

Also, add the android.permission.INTERNET permission to your app's manifest.

**Create Volley RequestQueue instance singleton in your Application **

```
public class InitApplication extends Application {

private RequestQueue queue;
private static InitApplication sInstance;

private static final String TAG = InitApplication.class.getSimpleName();


@Override
public void onCreate() {
    super.onCreate();

    sInstance = this;

    Stetho.initializeWithDefaults(this);

}

public static synchronized InitApplication getInstance() {
    return sInstance;
}

public <T> void addToQueue(Request<T> req, String tag) {
    req.setTag(TextUtils.isEmpty(tag) ? TAG : tag);
    getQueue().add(req);
}

public <T> void addToQueue(Request<T> req) {
    req.setTag(TAG);
    getQueue().add(req);
}

public void cancelPendingRequests(Object tag) {
    if (queue != null) {
        queue.cancelAll(tag);
    }
}

public RequestQueue getQueue() {
    if (queue == null) {
        queue = Volley.newRequestQueue(getApplicationContext());
        return queue;
```

```
    }
    return queue;
}
}
```

Now, you can use the volley instance using the getInstance() method and add a new request in the queue using `InitApplication.getInstance().addToQueue(request);`

A simple example to request JsonObject from server is

```
JsonObjectRequest myRequest = new JsonObjectRequest(Method.GET,
        url, null,
        new Response.Listener<JSONObject>() {

            @Override
            public void onResponse(JSONObject response) {
                Log.d(TAG, response.toString());
            }
        }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {
                Log.d(TAG, "Error: " + error.getMessage());
            }
});

myRequest.setRetryPolicy(new DefaultRetryPolicy(
        MY_SOCKET_TIMEOUT_MS,
        DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
        DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
```

To handle Volley timeouts you need to use a `RetryPolicy`. A retry policy is used in case a request cannot be completed due to network failure or some other cases.

Volley provides an easy way to implement your `RetryPolicy` for your requests. By default, Volley sets all socket and connection timeouts to 5 seconds for all requests. `RetryPolicy` is an interface where you need to implement your logic of how you want to retry a particular request when a timeout occurs.

The constructor takes the following three parameters:

- `initialTimeoutMs` - Specifies the socket timeout in milliseconds for every retry attempt.
- `maxNumRetries` - The number of times retry is attempted.
- `backoffMultiplier` - A multiplier which is used to determine exponential time set to socket for every retry attempt.

Section 98.2: Basic StringRequest using GET method

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...

// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
    @Override
```

```
    public void onResponse(String response) {
        // Display the first 500 characters of the response string.
        mTextView.setText("Response is: "+ response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        mTextView.setText("That didn't work!");
    }
});
// Add the request to the RequestQueue.
queue.add(stringRequest);
```

Section 98.3: Adding custom design time attributes to NetworkImageView

There are several additional attributes that the Volley `NetworkImageView` adds to the standard `ImageView`. However, these attributes can only be set in code. The following is an example of how to make an extension class that will pick up the attributes from your XML layout file and apply them to the `NetworkImageView` instance for you.

In your `~/res/xml` directory, add a file named `attrx.xml`:

```
<resources>
    <declare-styleable name="MoreNetworkImageView">
        <attr name="defaultImageResId" format="reference"/>
        <attr name="errorImageResId" format="reference"/>
    </declare-styleable>
</resources>
```

Add a new class file to your project:

```
package my.namespace;

import android.content.Context;
import android.content.res.TypedArray;
import android.support.annotation.NonNull;
import android.util.AttributeSet;

import com.android.volley.toolbox.NetworkImageView;

public class MoreNetworkImageView extends NetworkImageView {
    public MoreNetworkImageView(@NonNull final Context context) {
        super(context);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet attrs)
{
        this(context, attrs, 0);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet attrs,
final int defStyle) {
        super(context, attrs, defStyle);

        final TypedArray attributes = context.obtainStyledAttributes(attrs,
R.styleable.MoreNetworkImageView, defStyle, 0);

        // load defaultImageResId from XML
        int defaultImageResId =
```

```
attributes.getResourceId(R.styleable.MoreNetworkImageView_defaultImageResId, 0);
        if (defaultImageResId > 0) {
            setDefaultImageResId(defaultImageResId);
        }

        // load errorImageResId from XML
        int errorImageResId =
attributes.getResourceId(R.styleable.MoreNetworkImageView_errorImageResId, 0);
        if (errorImageResId > 0) {
            setErrorImageResId(errorImageResId);
        }
    }
}
```

An example layout file showing the use of the custom attributes:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="wrap_content"
 android:layout_height="fill_parent">

  <my.namespace.MoreNetworkImageView
   android:layout_width="64dp"
   android:layout_height="64dp"
   app:errorImageResId="@drawable/error_img"
   app:defaultImageResId="@drawable/default_img"
   tools:defaultImageResId="@drawable/editor_only_default_img"/>
    <!--
     Note: The "tools:" prefix does NOT work for custom attributes in Android Studio 2.1 and
     older at least, so in this example the defaultImageResId would show "default_img" in the
     editor, not the "editor_only_default_img" drawable even though it should if it was
     supported as an editor-only override correctly like standard Android properties.
   -->

</android.support.v7.widget.CardView>
```

Section 98.4: Adding custom headers to your requests [e.g. for basic auth]

If you need to add custom headers to your volley requests, you can't do this after initialisation, as the headers are saved in a private variable.

Instead, you need to override the `getHeaders()` method of `Request.class` as such:

```
new JsonObjectRequest(REQUEST_METHOD, REQUEST_URL, REQUEST_BODY, RESP_LISTENER, ERR_LISTENER) {
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        HashMap<String, String> customHeaders = new Hashmap<>();

        customHeaders.put("KEY_0", "VALUE_0");
        ...
        customHeaders.put("KEY_N", "VALUE_N");

        return customHeaders;
    }
};
```

Explanation of the parameters:

- `REQUEST_METHOD` - Either of the `Request.Method.*` constants.
- `REQUEST_URL` - The full URL to send your request to.
- `REQUEST_BODY` - A `JSONObject` containing the POST-Body to be sent (or null).
- `RESP_LISTENER` - A `Response.Listener<?>` object, whose `onResponse(T data)` method is called upon successful completion.
- `ERR_LISTENER` - A `Response.ErrorListener` object, whose `onErrorResponse(VolleyError e)` method is called upon a unsuccessful request.

If you want to build a custom request, you can add the headers in it as well:

```
public class MyCustomRequest extends Request {
    ...
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        HashMap<String, String> customHeaders = new Hashmap<>();

        customHeaders.put("KEY_0", "VALUE_0");
        ...
        customHeaders.put("KEY_N", "VALUE_N");

        return customHeaders;
    }
    ...
}
```

Section 98.5: Remote server authentication using StringRequest through POST method

For the sake of this example, let us assume that we have a server for handling the POST requests that we will be making from our Android app:

```
// User input data.
String email = "my@email.com";
String password = "123";

// Our server URL for handling POST requests.
String URL = "http://my.server.com/login.php";

// When we create a StringRequest (or a JSONRequest) for sending
// data with Volley, we specify the Request Method as POST, and
// the URL that will be receiving our data.
StringRequest stringRequest =
    new StringRequest(Request.Method.POST, URL,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            // At this point, Volley has sent the data to your URL
            // and has a response back from it. I'm going to assume
            // that the server sends an "OK" string.
            if (response.equals("OK")) {
                // Do login stuff.
            } else {
                // So the server didn't return an "OK" response.
                // Depending on what you did to handle errors on your
                // server, you can decide what action to take here.
            }
```

```
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // This is when errors related to Volley happen.
            // It's up to you what to do if that should happen, but
            // it's usually not a good idea to be too clear as to
            // what happened here to your users.
        }
    }) {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            // Here is where we tell Volley what it should send in
            // our POST request. For this example, we want to send
            // both the email and the password.

            // We will need key ids for our data, so our server can know
            // what is what.
            String key_email = "email";
            String key_password = "password";

            Map<String, String> map = new HashMap<String, String>();
            // map.put(key, value);
            map.put(key_email, email);
            map.put(key_password, password);
            return map;
        }
    };

    // This is a policy that we need to specify to tell Volley, what
    // to do if it gets a timeout, how many times to retry, etc.
    stringRequest.setRetryPolicy(new RetryPolicy() {
            @Override
            public int getCurrentTimeout() {
                // Here goes the timeout.
                // The number is in milliseconds, 5000 is usually enough,
                // but you can up or low that number to fit your needs.
                return 50000;
            }
            @Override
            public int getCurrentRetryCount() {
                // The maximum number of attempts.
                // Again, the number can be anything you need.
                return 50000;
            }
            @Override
            public void retry(VolleyError error) throws VolleyError {
                // Here you could check if the retry count has gotten
                // to the maximum number, and if so, send a VolleyError
                // message or similar. For the sake of the example, I'll
                // show a Toast.
                Toast.makeText(getContext(), error.toString(), Toast.LENGTH_LONG).show();
            }
    });

    // And finally, we create a Volley Queue. For this example, I'm using
    // getContext(), because I was working with a Fragment. But context could
    // be "this", "getContext()", etc.
    RequestQueue requestQueue = Volley.newRequestQueue(getContext());
    requestQueue.add(stringRequest);
```

```
} else {
    // If, for example, the user inputs an email that is not currently
    // on your remote DB, here's where we can inform the user.
    Toast.makeText(getContext(), "Wrong email", Toast.LENGTH_LONG).show();
}
```

Section 98.6: Cancel a request

```
// assume a Request and RequestQueue have already been initialized somewhere above

public static final String TAG = "SomeTag";

// Set the tag on the request.
request.setTag(TAG);

// Add the request to the RequestQueue.
mRequestQueue.add(request);

// To cancel this specific request
request.cancel();

// ... then, in some future life cycle event, for example in onStop()
// To cancel all requests with the specified tag in RequestQueue
mRequestQueue.cancelAll(TAG);
```

Section 98.7: Request JSON

```
final TextView mTxtDisplay = (TextView) findViewById(R.id.txtDisplay);
ImageView mImageView;
String url = "http://ip.jsontest.com/";

final JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
    @Override
    public void onResponse(JSONObject response) {
        mTxtDisplay.setText("Response: " + response.toString());
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // ...
    }
});

requestQueue.add(jsObjRequest);
```

Section 98.8: Use JSONArray as request body

The default requests integrated in volley don't allow to pass a `JSONArray` as request body in a `POST` request. Instead, you can only pass a `JSON` object as a parameter.

However, instead of passing a `JSON` object as a parameter to the request constructor, you need to override the `getBody()` method of the `Request.class`. You should pass `null` as third parameter as well:

```
JSONArray requestBody = new JSONArray();

new JsonObjectRequest(Request.Method.POST, REQUEST_URL, null, RESP_LISTENER, ERR_LISTENER) {
    @Override
```

```
    public byte[] getBody() {
        try {
            return requestBody.toString().getBytes(PROTOCOL_CHARSET);
        } catch (UnsupportedEncodingException uee) {
            // error handling
            return null;
        }
    }
};
```

Explanation of the parameters:

- `REQUEST_URL` - The full URL to send your request to.
- `RESP_LISTENER` - A `Response.Listener<?>` object, whose `onResponse(T data)` method is called upon successful completion.
- `ERR_LISTENER` - A `Response.ErrorListener` object, whose `onErrorResponse(VolleyError e)` method is called upon an unsuccessful request.

Section 98.9: Boolean variable response from server with json request in volley

you can custom class below one

```
private final String PROTOCOL_CONTENT_TYPE = String.format("application/json; charset=%s",
PROTOCOL_CHARSET);

    public BooleanRequest(int method, String url, String requestBody, Response.Listener<Boolean>
listener, Response.ErrorListener errorListener) {
        super(method, url, errorListener);
        this.mListener = listener;
        this.mErrorListener = errorListener;
        this.mRequestBody = requestBody;
    }

    @Override
    protected Response<Boolean> parseNetworkResponse(NetworkResponse response) {
        Boolean parsed;
        try {
            parsed = Boolean.valueOf(new String(response.data,
HttpHeaderParser.parseCharset(response.headers)));
        } catch (UnsupportedEncodingException e) {
            parsed = Boolean.valueOf(new String(response.data));
        }
        return Response.success(parsed, HttpHeaderParser.parseCacheHeaders(response));
    }

    @Override
    protected VolleyError parseNetworkError(VolleyError volleyError) {
        return super.parseNetworkError(volleyError);
    }

    @Override
    protected void deliverResponse(Boolean response) {
        mListener.onResponse(response);
    }

    @Override
    public void deliverError(VolleyError error) {
        mErrorListener.onErrorResponse(error);
```

```
    }

    @Override
    public String getBodyContentType() {
        return PROTOCOL_CONTENT_TYPE;
    }

    @Override
    public byte[] getBody() throws AuthFailureError {
        try {
            return mRequestBody == null ? null : mRequestBody.getBytes(PROTOCOL_CHARSET);
        } catch (UnsupportedEncodingException uee) {
            VolleyLog.wtf("Unsupported Encoding while trying to get the bytes of %s using %s",
                    mRequestBody, PROTOCOL_CHARSET);
            return null;
        }
    }
}
```

use this with your activity

```
try {
        JSONObject jsonBody;
        jsonBody = new JSONObject();
        jsonBody.put("Title", "Android Demo");
        jsonBody.put("Author", "BNK");
        jsonBody.put("Date", "2015/08/28");
        String requestBody = jsonBody.toString();
        BooleanRequest booleanRequest = new BooleanRequest(0, url, requestBody, new
Response.Listener<Boolean>() {
            @Override
            public void onResponse(Boolean response) {
                Toast.makeText(mContext, String.valueOf(response), Toast.LENGTH_SHORT).show();
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(mContext, error.toString(), Toast.LENGTH_SHORT).show();
            }
        });
        // Add the request to the RequestQueue.
        queue.add(booleanRequest);
    } catch (JSONException e) {
        e.printStackTrace();
    }
```

Section 98.10: Helper Class for Handling Volley Errors

```
public class VolleyErrorHelper {
        /**
         * Returns appropriate message which is to be displayed to the user
         * against the specified error object.
         *
         * @param error
         * @param context
         * @return
         */

        public static String getMessage (Object error , Context context){
            if(error instanceof TimeoutError){
```

```
                return context.getResources().getString(R.string.timeout);
            }else if (isServerProblem(error)){
                return handleServerError(error ,context);

            }else if(isNetworkProblem(error)){
                return context.getResources().getString(R.string.nointernet);
            }
            return context.getResources().getString(R.string.generic_error);

        }

        private static String handleServerError(Object error, Context context) {

            VolleyError er = (VolleyError)error;
            NetworkResponse response = er.networkResponse;
            if(response != null){
                switch (response.statusCode){

                    case 404:
                    case 422:
                    case 401:
                        try {
                            // server might return error like this { "error": "Some error occurred"
}
                            // Use "Gson" to parse the result
                            HashMap<String, String> result = new Gson().fromJson(new
String(response.data),
                                    new TypeToken<Map<String, String>>() {
                                    }.getType());

                            if (result != null && result.containsKey("error")) {
                                return result.get("error");
                            }

                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                        // invalid request
                        return ((VolleyError) error).getMessage();

                    default:
                        return context.getResources().getString(R.string.timeout);
                }
            }

            return context.getResources().getString(R.string.generic_error);
        }

        private static boolean isServerProblem(Object error) {
            return (error instanceof ServerError || error instanceof AuthFailureError);
        }

        private static boolean isNetworkProblem (Object error){
            return (error instanceof NetworkError || error instanceof NoConnectionError);
        }
```

Chapter 99: Date and Time Pickers

Section 99.1: Date Picker Dialog

It is a dialog which prompts user to select date using `DatePicker`. The dialog requires context, initial year, month and day to show the dialog with starting date. When the user selects the date it callbacks via `DatePickerDialog.OnDateSetListener`.

```
public void showDatePicker(Context context,int initialYear, int initialMonth, int initialDay) {
       DatePickerDialog datePickerDialog = new DatePickerDialog(context,
                new DatePickerDialog.OnDateSetListener() {
                    @Override
                    public void onDateSet(DatePicker datepicker,int year ,int month, int day) {
                    //this condition is necessary to work properly on all android versions
                    if(view.isShown()){
                            //You now have the selected year, month and day
                    }

}
                }, initialYear, initialMonth , initialDay);

        //Call show() to simply show the dialog
        datePickerDialog.show();

    }
```

Please note that month is a int starting from 0 for January to 11 for December

Section 99.2: Material DatePicker

add below dependencies to `build.gradle` file in dependency section. (this is an unOfficial library for date picker)

```
  compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

Now we have to open `DatePicker` on Button click event.

So create one Button on xml file like below.

```
    <Button
          android:id="@+id/dialog_bt_date"
          android:layout_below="@+id/resetButton"
          android:layout_width="wrap_content"
          android:layout_height="40dp"
          android:textColor="#FF000000"
          android:gravity="center"
          android:text="DATE"/>
```

and in MainActivity use this way.

```
public class MainActivity extends AppCompatActivity implements DatePickerDialog.OnDateSetListener{

    Button button;
    Calendar calendar ;
    DatePickerDialog datePickerDialog ;
    int Year, Month, Day ;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        calendar = Calendar.getInstance();

        Year = calendar.get(Calendar.YEAR) ;
        Month = calendar.get(Calendar.MONTH);
        Day = calendar.get(Calendar.DAY_OF_MONTH);


        Button dialog_bt_date = (Button)findViewById(R.id.dialog_bt_date);
        dialog_bt_date.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                datePickerDialog = DatePickerDialog.newInstance(MainActivity.this, Year, Month,
Day);

                datePickerDialog.setThemeDark(false);

                datePickerDialog.showYearPickerFirst(false);

                datePickerDialog.setAccentColor(Color.parseColor("#0072BA"));

                datePickerDialog.setTitle("Select Date From DatePickerDialog");

                datePickerDialog.show(getFragmentManager(), "DatePickerDialog");

            }
        });
    }

    @Override
    public void onDateSet(DatePickerDialog view, int Year, int Month, int Day) {

        String date = "Selected Date : " + Day + "-" + Month + "-" + Year;

        Toast.makeText(MainActivity.this, date, Toast.LENGTH_LONG).show();
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.abc_main_menu, menu);
        return true;
    }

}
```

Output :

Chapter 100: Localized Date/Time in Android

Section 100.1: Custom localized date format with DateUtils.formatDateTime()

DateUtils.formatDateTime() allows you to supply a time, and based on the flags you provide, it creates a localized datetime string. The flags allow you to specify whether to include specific elements (like the weekday).

Date date = new Date(); String localizedDate = DateUtils.formatDateTime(context, date.getTime(), DateUtils.FORMAT_SHOW_DATE | DateUtils.FORMAT_SHOW_WEEKDAY);

formatDateTime() automatically takes care about proper date formats.

Section 100.2: Standard date/time formatting in Android

Format a date:

Date date = new Date(); DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM); String localizedDate = df.format(date)

Format a date and time. Date is in short format, time is in long format:

Date date = new Date(); DateFormat df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG); String localizedDate = df.format(date)

Section 100.3: Fully customized date/time

Date date = new Date(); df = new SimpleDateFormat("HH:mm", Locale.US); String localizedDate = df.format(date)

Commonly used patterns:

- HH: hour (0-23)
- hh: hour (1-12)
- a: AM/PM marker
- mm: minute (0-59)
- ss: second
- dd: day in month (1-31)
- MM: month
- yyyy: year

Chapter 101: Time Utils

Section 101.1: To check within a period

This example will help to verify the given time is within a period or not.

*To check the time is today, We can use **DateUtils** class*

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

To check the time is within a week,

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

To check the time is within a year,

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

To check the time is within a number day of day including today,

```
public static boolean isWithinDay(long timeInMillis, int day) {
    long diff = System.currentTimeMillis() - timeInMillis;

    float dayCount = (float) (diff / DateUtils.DAY_IN_MILLIS);

    return dayCount < day;
}
```

Note : DateUtils is **android.text.format.DateUtils**

Section 101.2: Convert Date Format into Milliseconds

To Convert you date in dd/MM/yyyy format into milliseconds you call this function with data as String

```
public long getMilliFromDate(String dateFormat) {
    Date date = new Date();
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
    try {
        date = formatter.parse(dateFormat);
    } catch (ParseException e) {
        e.printStackTrace();
    }
    System.out.println("Today is " + date);
    return date.getTime();
}
```

This method converts milliseconds to Time-stamp Format date :

```
public String getTimeStamp(long timeinMillies) {
    String date = null;
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"); // modify format
```

```
    date = formatter.format(new Date(timeinMillies));
    System.out.println("Today is " + date);

    return date;
}
```

This Method will convert given specific day,month and year into milliseconds. It will be very help when using `Timpicker` or `Datepicker`

```
public static long getTimeInMillis(int day, int month, int year) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day);
    return calendar.getTimeInMillis();
}
```

It will return milliseconds from date

```
public static String getNormalDate(long timeInMillies) {
    String date = null;
    SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
    date = formatter.format(timeInMillies);
    System.out.println("Today is " + date);
    return date;
}
```

It will return current date

```
public static String getCurrentDate() {
    Calendar c = Calendar.getInstance();
    System.out.println("Current time => " + c.getTime());
    SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
    String formattedDate = df.format(c.getTime());
    return formattedDate;
}
```

Note : Java Provides numbers of date format support Date Pattern

Section 101.3: GetCurrentRealTime

This calculate current device time and add/subtract difference between real and device time

```
public static Calendar getCurrentRealTime() {

    long bootTime = networkTime - SystemClock.elapsedRealtime();
    Calendar calInstance = Calendar.getInstance();
    calInstance.setTimeZone(getUTCTimeZone());
    long currentDeviceTime = bootTime + SystemClock.elapsedRealtime();
    calInstance.setTimeInMillis(currentDeviceTime);
    return calInstance;
}
```

get UTC based timezone.

```
public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}
```

Chapter 102: In-app Billing

Section 102.1: Consumable In-app Purchases

Consumable Managed Products are products that can be bought multiple times such as in-game currency, game lives, power-ups, etc.

In this example, we are going to implement 4 different consumable **managed products** `"item1"`, `"item2"`, `"item3"`, `"item4"`.

> **Steps in summary:**
>
> 1. Add the In-app Billing library to your project (AIDL File).
> 2. Add the required permission in `AndroidManifest.xml` file.
> 3. Deploy a signed apk to Google Developers Console.
> 4. Define your products.
> 5. Implement the code.
> 6. Test In-app Billing (optional).

Step 1:

First of all, we will need to add the AIDL file to your project as clearly explained in Google Documentation here.

`IInAppBillingService.aidl` is an Android Interface Definition Language (AIDL) file that defines the interface to the In-app Billing Version 3 service. You will use this interface to make billing requests by invoking IPC method calls.

Step 2:

After adding the AIDL file, add BILLING permission in `AndroidManifest.xml`:

```
<!-- Required permission for implementing In-app Billing -->
<uses-permission android:name="com.android.vending.BILLING" />
```

Step 3:

Generate a signed apk, and upload it to Google Developers Console. This is required so that we can start defining our in-app products there.

Step 4:

Define all your products with different productID, and set a price to each one of them. There are 2 types of products (Managed Products and Subscriptions). As we already said, we are going to implement 4 different consumable **managed products** `"item1"`, `"item2"`, `"item3"`, `"item4"`.

Step 5:

After doing all the steps above, you are now ready to start implementing the code itself in your own activity.

MainActivity:

```
public class MainActivity extends Activity {

    IInAppBillingService inAppBillingService;
    ServiceConnection serviceConnection;
```

```
    // productID for each item. You should define them in the Google Developers Console.
    final String item1 = "item1";
    final String item2 = "item2";
    final String item3 = "item3";
    final String item4 = "item4";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Instantiate the views according to your layout file.
        final Button buy1 = (Button) findViewById(R.id.buy1);
        final Button buy2 = (Button) findViewById(R.id.buy2);
        final Button buy3 = (Button) findViewById(R.id.buy3);
        final Button buy4 = (Button) findViewById(R.id.buy4);

        // setOnClickListener() for each button.
        // buyItem() here is the method that we will implement to launch the PurchaseFlow.
        buy1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item1);
            }
        });

        buy2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item2);
            }
        });

        buy3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item3);
            }
        });

        buy4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item4);
            }
        });

        // Attach the service connection.
        serviceConnection = new ServiceConnection() {
            @Override
            public void onServiceDisconnected(ComponentName name) {
                inAppBillingService = null;
            }

            @Override
            public void onServiceConnected(ComponentName name, IBinder service) {
                inAppBillingService = IInAppBillingService.Stub.asInterface(service);
            }
        };

        // Bind the service.
        Intent serviceIntent = new Intent("com.android.vending.billing.InAppBillingService.BIND");
```

```
        serviceIntent.setPackage("com.android.vending");
        bindService(serviceIntent, serviceConnection, BIND_AUTO_CREATE);

        // Get the price of each product, and set the price as text to
        // each button so that the user knows the price of each item.
        if (inAppBillingService != null) {
            // Attention: You need to create a new thread here because
            // getSkuDetails() triggers a network request, which can
            // cause lag to your app if it was called from the main thread.
            Thread thread = new Thread(new Runnable() {
                @Override
                public void run() {
                    ArrayList<String> skuList = new ArrayList<>();
                    skuList.add(item1);
                    skuList.add(item2);
                    skuList.add(item3);
                    skuList.add(item4);
                    Bundle querySkus = new Bundle();
                    querySkus.putStringArrayList("ITEM_ID_LIST", skuList);

                    try {
                        Bundle skuDetails = inAppBillingService.getSkuDetails(3, getPackageName(),
"inapp", querySkus);
                        int response = skuDetails.getInt("RESPONSE_CODE");

                        if (response == 0) {
                            ArrayList<String> responseList =
skuDetails.getStringArrayList("DETAILS_LIST");

                            for (String thisResponse : responseList) {
                                JSONObject object = new JSONObject(thisResponse);
                                String sku = object.getString("productId");
                                String price = object.getString("price");

                                switch (sku) {
                                    case item1:
                                        buy1.setText(price);
                                        break;
                                    case item2:
                                        buy2.setText(price);
                                        break;
                                    case item3:
                                        buy3.setText(price);
                                        break;
                                    case item4:
                                        buy4.setText(price);
                                        break;
                                }
                            }
                        }
                    } catch (RemoteException | JSONException e) {
                        e.printStackTrace();
                    }
                }
            });
            thread.start();
        }
    }

    // Launch the PurchaseFlow passing the productID of the item the user wants to buy as a
parameter.
    private void buyItem(String productID) {
```

```
        if (inAppBillingService != null) {
            try {
                Bundle buyIntentBundle = inAppBillingService.getBuyIntent(3, getPackageName(),
productID, "inapp", "bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ");
                PendingIntent pendingIntent = buyIntentBundle.getParcelable("BUY_INTENT");
                startIntentSenderForResult(pendingIntent.getIntentSender(), 1003, new Intent(), 0,
0, 0);
            } catch (RemoteException | IntentSender.SendIntentException e) {
                e.printStackTrace();
            }
        }
    }

    // Unbind the service in onDestroy(). If you don't unbind, the open
    // service connection could cause your device's performance to degrade.
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (inAppBillingService != null) {
            unbindService(serviceConnection);
        }
    }

    // Check here if the in-app purchase was successful or not. If it was successful,
    // then consume the product, and let the app make the required changes.
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == 1003 && resultCode == RESULT_OK) {

            final String purchaseData = data.getStringExtra("INAPP_PURCHASE_DATA");

            // Attention: You need to create a new thread here because
            // consumePurchase() triggers a network request, which can
            // cause lag to your app if it was called from the main thread.
            Thread thread = new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        JSONObject jo = new JSONObject(purchaseData);
                        // Get the productID of the purchased item.
                        String sku = jo.getString("productId");
                        String productName = null;

                        // increaseCoins() here is a method used as an example in a game to
                        // increase the in-game currency if the purchase was successful.
                        // You should implement your own code here, and let the app apply
                        // the required changes after the purchase was successful.
                        switch (sku) {
                            case item1:
                                productName = "Item 1";
                                increaseCoins(2000);
                                break;
                            case item2:
                                productName = "Item 2";
                                increaseCoins(8000);
                                break;
                            case item3:
                                productName = "Item 3";
                                increaseCoins(18000);
                                break;
```

```
                            case item4:
                                productName = "Item 4";
                                increaseCoins(30000);
                                break;
                        }

                        // Consume the purchase so that the user is able to purchase the same
product again.
                        inAppBillingService.consumePurchase(3, getPackageName(),
jo.getString("purchaseToken"));
                        Toast.makeText(MainActivity.this, productName + " is successfully
purchased. Excellent choice, master!", Toast.LENGTH_LONG).show();
                    } catch (JSONException | RemoteException e) {
                        Toast.makeText(MainActivity.this, "Failed to parse purchase data.",
Toast.LENGTH_LONG).show();
                        e.printStackTrace();
                    }
                }
            });
            thread.start();
        }
    }
}
```

Step 6:

After implementing the code, you can test it by deploying your apk to beta/alpha channel, and let other users test the code for you. However, real in-app purchases can't be made while in testing mode. You have to publish your app/game first to Play Store so that all the products are fully activated.

More info on testing In-app Billing can be found here.

Section 102.2: (Third party) In-App v3 Library

Step 1: First of all follow these two steps to add in app functionality :

1. Add the library using :

```
repositories {
            mavenCentral()
        }
        dependencies {
           compile 'com.anjlab.android.iab.v3:library:1.0.+'
        }
```

2. Add permission in manifest file.

```
<uses-permission android:name="com.android.vending.BILLING" />
```

Step 2: Initialise your billing processor:

```
BillingProcessor bp = new BillingProcessor(this, "YOUR LICENSE KEY FROM GOOGLE PLAY CONSOLE HERE",
this);
```

and implement Billing Handler : BillingProcessor.IBillingHandler which contains 4 methods : a. onBillingInitialized(); b. onProductPurchased(String productId, TransactionDetails details) : This is where you need to handle actions to be performed after successful purchase c. onBillingError(int errorCode, Throwable error) : Handle any error occurred during purchase process d. onPurchaseHistoryRestored() : For restoring in app purchases

Step 3: How to purchase a product.

To purchase a managed product :

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

And to Purchase a subscription :

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

Step 4 : Consuming a product.

To consume a product simply call consumePurchase method.

bp.consumePurchase("YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");

For other methods related to in app visit github

Chapter 103: FloatingActionButton

Parameter	Detail
`android.support.design:elevation`	Elevation value for the FAB. May be a reference to another resource, in the form "@[+][package:]type/name" or a theme attribute in the form "?[package:]type/name".
`android.support.design:fabSize`	Size for the FAB.
`android.support.design:rippleColor`	Ripple color for the FAB.
`android.support.design:useCompatPadding`	Enable compat padding.

Floating action button is used for a special type of promoted action,it animates onto the screen as an expanding piece of material, by default. The icon within it may be animated,also FAB may move differently than other UI elements because of their relative importance. A floating action button represents the primary action in an application which can simply trigger an action or navigate somewhere.

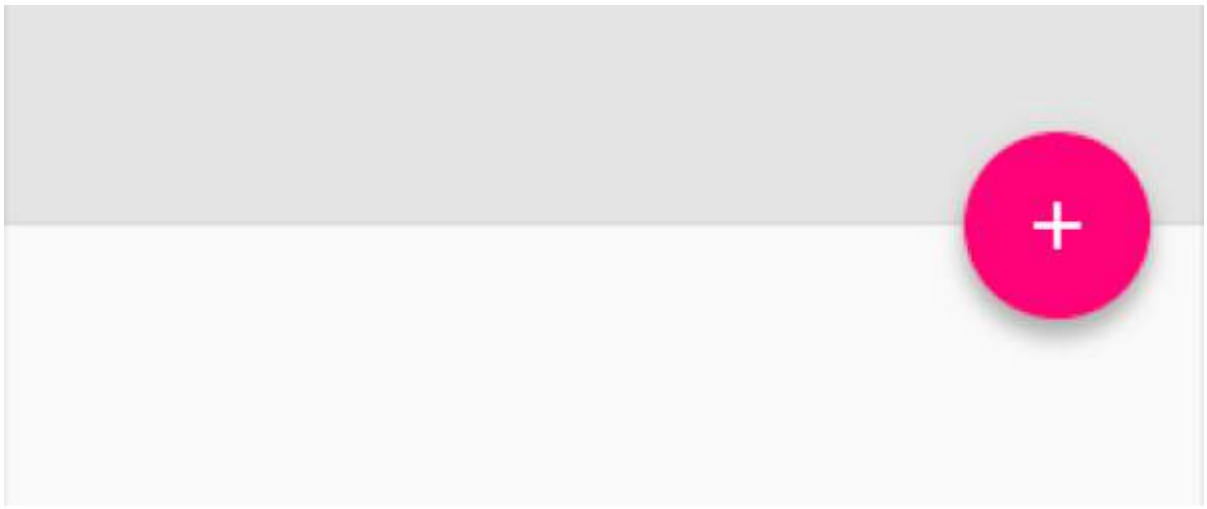
Section 103.1: How to add the FAB to the layout

To use a FloatingActionButton just add the dependency in the `build.gradle` file as described in the remarks section.

Then add to the layout:

```
<android.support.design.widget.FloatingActionButton
      android:id="@+id/fab"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="bottom|end"
      android:layout_margin="@dimen/fab_margin"
      android:src="@drawable/my_icon" />
```

An example:



Color

The background color of this view defaults to the your theme's colorAccent.

In the above image if the `src` only points to + icon (by default 24x24 dp),to get the *background color* of full circle you can use `app:backgroundTint="@color/your_colour"`

If you wish to change the color in code you can use,

```
myFab.setBackgroundTintList(ColorStateList.valueOf(your color in int));
```

If you want to change FAB's color in pressed state use

```
mFab.setRippleColor(your color in int);
```

Positioning

It is recommended to place 16dp minimum from the edge on mobile,and 24dp minimum on tablet/desktop.

Note : Once you set an src excepting to cover the full area of `FloatingActionButton` make sure you have the right size of that image to get the best result.

Default circle size is 56 x 56dp

Mini circle size : 40 x 40dp

If you only want to change only the Interior icon use a 24 x 24dp icon for default size

Section 103.2: Show and Hide FloatingActionButton on Swipe

To show and hide a `FloatingActionButton` with the default animation, just call the methods `show()` and `hide()`. It's good practice to keep a `FloatingActionButton` in the Activity layout instead of putting it in a Fragment, this allows the default animations to work when showing and hiding.

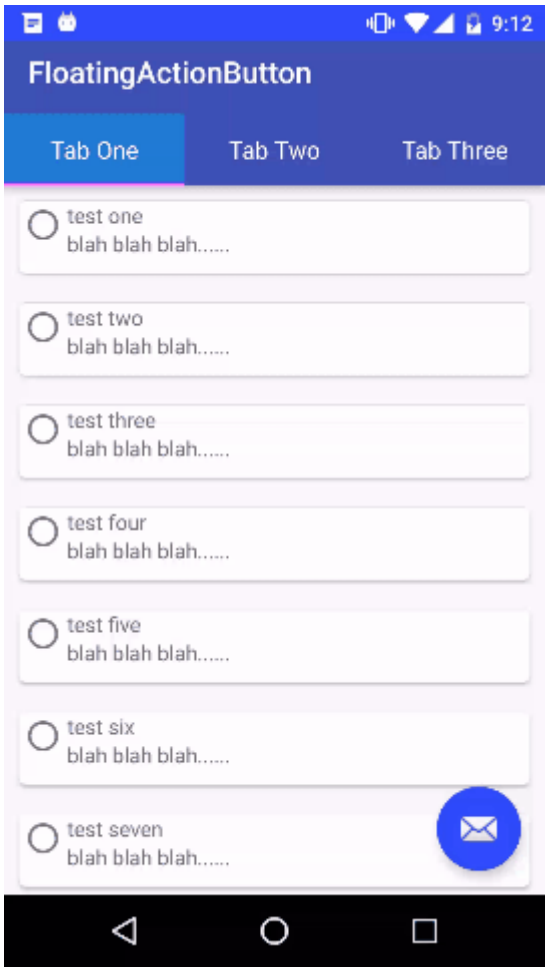
Here is an example with a `ViewPager`:

- Three Tabs
- Show `FloatingActionButton` for the first and third Tab
- Hide the `FloatingActionButton` on the middle Tab

```
public class MainActivity extends AppCompatActivity {

    FloatingActionButton fab;
    ViewPager viewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        fab = (FloatingActionButton) findViewById(R.id.fab);
        viewPager = (ViewPager) findViewById(R.id.viewpager);

        // ..... set up ViewPager ............

        viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {

            @Override
            public void onPageSelected(int position) {
                if (position == 0) {
                    fab.setImageResource(android.R.drawable.ic_dialog_email);
                    fab.show();
                } else if (position == 2) {
```

```
                    fab.setImageResource(android.R.drawable.ic_dialog_map);
                    fab.show();
                } else {
                    fab.hide();
                }
            }

            @Override
            public void onPageScrolled(int position, float positionOffset, int
positionOffsetPixels) {}

            @Override
            public void onPageScrollStateChanged(int state) {}
        });

        // Handle the FloatingActionButton click event:
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int position = viewPager.getCurrentItem();
                if (position == 0) {
                    openSend();
                } else if (position == 2) {
                    openMap();
                }
            }
        });

    }
}
```

Result:



Section 103.3: Show and Hide FloatingActionButton on Scroll

Starting with the Support Library version 22.2.1, it's possible to show and hide a FloatingActionButton from scrolling behavior using a `FloatingActionButton.Behavior` sublclass that takes advantage of the `show()` and `hide()` methods.

Note that this only works with a CoordinatorLayout in conjunction with inner Views that support Nested Scrolling, such as RecyclerView and NestedScrollView.

This `ScrollAwareFABBehavior` class comes from the Android Guides on Codepath (cc-wiki with attribution required)

```
public class ScrollAwareFABBehavior extends FloatingActionButton.Behavior {
    public ScrollAwareFABBehavior(Context context, AttributeSet attrs) {
        super();
    }

    @Override
    public boolean onStartNestedScroll(final CoordinatorLayout coordinatorLayout, final
FloatingActionButton child,
                                       final View directTargetChild, final View target, final int
nestedScrollAxes) {
        // Ensure we react to vertical scrolling
        return nestedScrollAxes == ViewCompat.SCROLL_AXIS_VERTICAL
                || super.onStartNestedScroll(coordinatorLayout, child, directTargetChild, target,
nestedScrollAxes);
    }

    @Override
    public void onNestedScroll(final CoordinatorLayout coordinatorLayout, final
FloatingActionButton child,
                               final View target, final int dxConsumed, final int dyConsumed,
                               final int dxUnconsumed, final int dyUnconsumed) {
        super.onNestedScroll(coordinatorLayout, child, target, dxConsumed, dyConsumed,
dxUnconsumed, dyUnconsumed);
        if (dyConsumed > 0 && child.getVisibility() == View.VISIBLE) {
            // User scrolled down and the FAB is currently visible -> hide the FAB
            child.hide();
        } else if (dyConsumed < 0 && child.getVisibility() != View.VISIBLE) {
            // User scrolled up and the FAB is currently not visible -> show the FAB
            child.show();
        }
    }
}
```

In the FloatingActionButton layout xml, specify the `app:layout_behavior` with the fully-qualified-class-name of `ScrollAwareFABBehavior`:

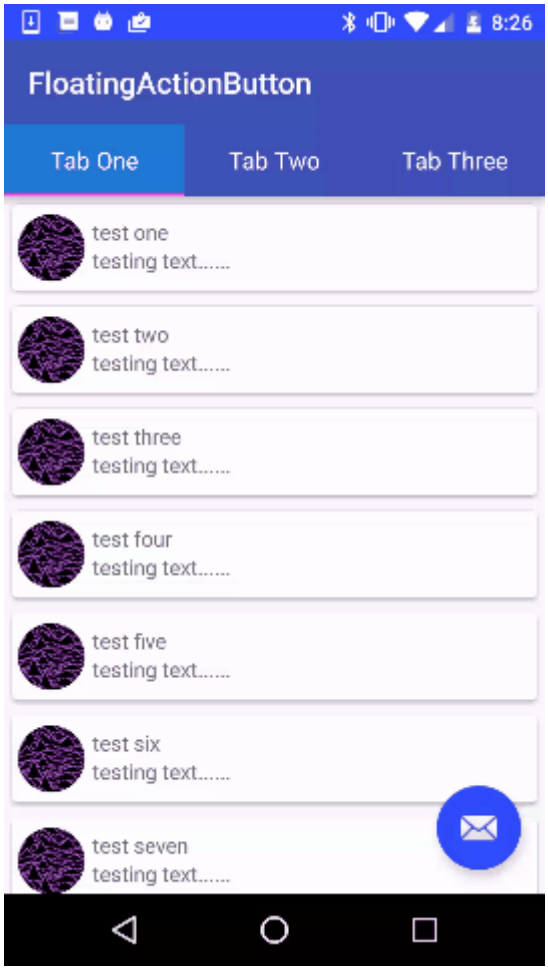
```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

For example with this layout:

```
<android.support.design.widget.CoordinatorLayout
   android:id="@+id/main_layout"
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">
```

```
    <android.support.design.widget.AppBarLayout
       android:id="@+id/appBarLayout"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       app:elevation="6dp">
        <android.support.v7.widget.Toolbar
           android:id="@+id/toolbar"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:layout_alignParentTop="true"
           android:background="?attr/colorPrimary"
           android:minHeight="?attr/actionBarSize"
           android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
           app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
           app:elevation="0dp"
           app:layout_scrollFlags="scroll|enterAlways"
           />

        <android.support.design.widget.TabLayout
           android:id="@+id/tab_layout"
           app:tabMode="fixed"
           android:layout_below="@+id/toolbar"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:background="?attr/colorPrimary"
           app:elevation="0dp"
           app:tabTextColor="#d3d3d3"
           android:minHeight="?attr/actionBarSize"
           />

    </android.support.design.widget.AppBarLayout>

    <android.support.v4.view.ViewPager
       android:id="@+id/viewpager"
       android:layout_below="@+id/tab_layout"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       app:layout_behavior="@string/appbar_scrolling_view_behavior"
       />

    <android.support.design.widget.FloatingActionButton
       android:id="@+id/fab"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="bottom|end"
       app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
       android:layout_margin="@dimen/fab_margin"
       android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

Here is the result:



Section 103.4: Setting behaviour of FloatingActionButton

You can set the behavior of the FAB in XML.

For example:

```
<android.support.design.widget.FloatingActionButton
   app:layout_behavior=".MyBehavior" />
```

Or you can set programmatically using:

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();
p.setBehavior(xxxx);
fab.setLayoutParams(p);
```

Chapter 104: Touch Events

Section 104.1: How to vary between child and parent view group touch events

1. The `onTouchEvents()` for nested view groups can be managed by the **`boolean`** onInterceptTouchEvent.

The default value for the `OnInterceptTouchEvent` is false.

The parent's `onTouchEvent` is received before the child's. If the `OnInterceptTouchEvent` returns false, it sends the motion event down the chain to the child's `OnTouchEvent` handler. If it returns true the parent's will handle the touch event.

However there may be instances when we want some child elements to manage `OnTouchEvents` and some to be managed by the parent view (or possibly the parent of the parent).

This can be managed in more than one way.

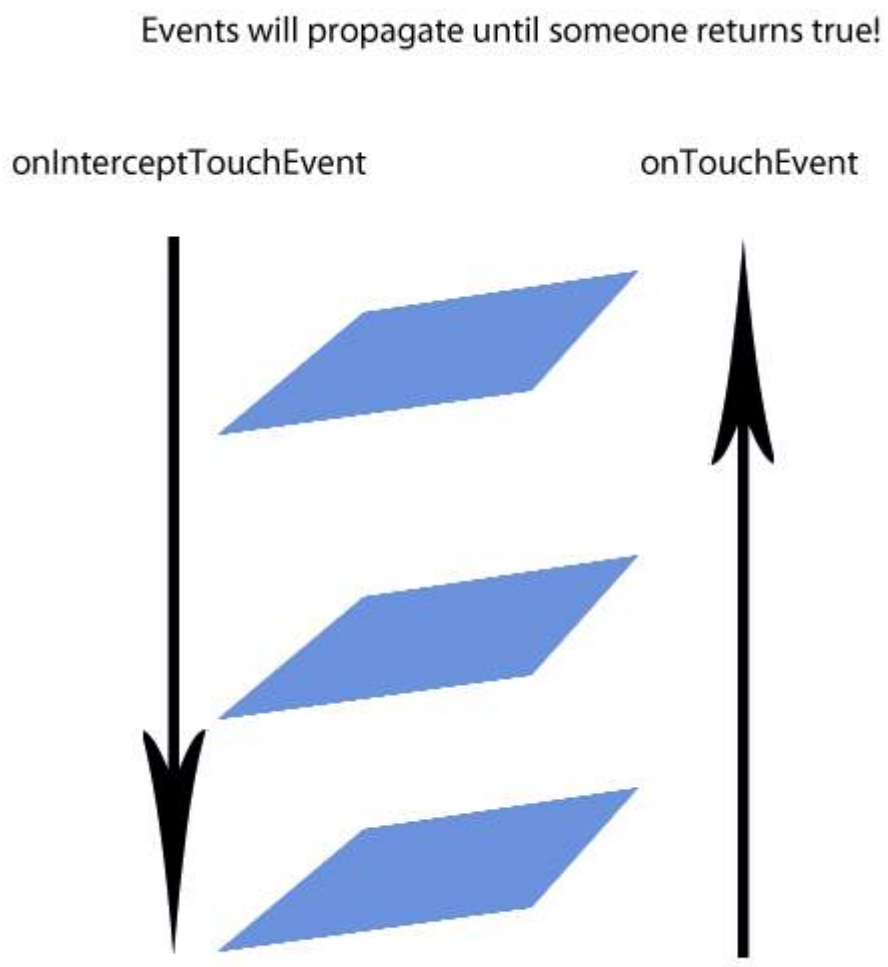
2. One way a child element can be protected from the parent's `OnInterceptTouchEvent` is by implementing the requestDisallowInterceptTouchEvent.

> public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)

This prevents any of the parent views from managing the `OnTouchEvent` for this element, if the element has event handlers enabled.

If the `OnInterceptTouchEvent` is false, the child element's `OnTouchEvent` will be evaluated. If you have a methods within the child elements handling the various touch events, any related event handlers that are disabled will return the OnTouchEvent to the parent.

This answer:
A visualisation of how the propagation of touch events passes through:
`parent -> child|parent -> child|parent -> child views.`



Courtesy from here

4. Another way is returning varying values from the `OnInterceptTouchEvent` for the parent.

This example taken from Managing Touch Events in a ViewGroup and demonstrates how to intercept the child's `OnTouchEvent` when the user is scrolling.

4a.

```
@Override
public boolean onInterceptTouchEvent(MotionEvent ev) {
    /*
     * This method JUST determines whether we want to intercept the motion.
     * If we return true, onTouchEvent will be called and we do the actual
     * scrolling there.
     */


    final int action = MotionEventCompat.getActionMasked(ev);

    // Always handle the case of the touch gesture being complete.
    if (action == MotionEvent.ACTION_CANCEL || action == MotionEvent.ACTION_UP) {
        // Release the scroll.
        mIsScrolling = false;
        return false; // Do not intercept touch event, let the child handle it
```

```
    }

    switch (action) {
        case MotionEvent.ACTION_MOVE: {
            if (mIsScrolling) {
                // We're currently scrolling, so yes, intercept the
                // touch event!
                return true;
            }

            // If the user has dragged her finger horizontally more than
            // the touch slop, start the scroll

            // left as an exercise for the reader
            final int xDiff = calculateDistanceX(ev);

            // Touch slop should be calculated using ViewConfiguration
            // constants.
            if (xDiff > mTouchSlop) {
                // Start scrolling!
                mIsScrolling = true;
                return true;
            }
            break;
        }
        ...
    }

    // In general, we don't want to intercept touch events. They should be
    // handled by the child view.
    return false;
}
```

This is some code from the same link showing how to create the parameters of the rectangle around your element:

4b.

```
// The hit rectangle for the ImageButton
myButton.getHitRect(delegateArea);

// Extend the touch area of the ImageButton beyond its bounds
// on the right and bottom.
delegateArea.right += 100;
delegateArea.bottom += 100;

// Instantiate a TouchDelegate.
// "delegateArea" is the bounds in local coordinates of
// the containing view to be mapped to the delegate view.
// "myButton" is the child view that should receive motion
// events.
TouchDelegate touchDelegate = new TouchDelegate(delegateArea, myButton);

// Sets the TouchDelegate on the parent view, such that touches
// within the touch delegate bounds are routed to the child.
if (View.class.isInstance(myButton.getParent())) {
    ((View) myButton.getParent()).setTouchDelegate(touchDelegate);
}
```

Chapter 105: Handling touch and motion events

Listener	Details
onTouchListener	Handles single touches for buttons, surfaces and more
onTouchEvent	A listener that can be found in surfaces(e.g. SurfaceView). Does not need to be set like other listeners(e,g. onTouchListener)
onLongTouch	Similar to onTouch, but listens for long presses in buttons, surfaces and more.

A summary of some of the basic touch/motion-handling systems in the Android API.

Section 105.1: Buttons

Touch events related to a `Button` can be checked as follows:

```
public class ExampleClass extends Activity implements View.OnClickListener,
View.OnLongClickListener{
    public Button onLong, onClick;

    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        setContentView(R.layout.layout);
        onLong = (Button) findViewById(R.id.onLong);
        onClick = (Button) findViewById(R.id.onClick);
        // The buttons are created. Now we need to tell the system that
        // these buttons have a listener to check for touch events.
        // "this" refers to this class, as it contains the appropriate event listeners.
        onLong.setOnLongClickListener(this);
        onClick.setOnClickListener(this);

        [OR]

        onClick.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v){
                // Take action. This listener is only designed for one button.
                // This means, no other input will come here.
                // This makes a switch statement unnecessary here.
            }
        });

        onLong.setOnLongClickListener(new View.OnLongClickListener(){
            @Override
            public boolean onLongClick(View v){
                // See comment in onClick.setOnClickListener().
            }
        });
    }

    @Override
    public void onClick(View v) {
        // If you have several buttons to handle, use a switch to handle them.
        switch(v.getId()){
            case R.id.onClick:
                // Take action.
                break;
        }
```

```
    }

    @Override
    public boolean onLongClick(View v) {
        // If you have several buttons to handle, use a switch to handle them.
        switch(v.getId()){
            case R.id.onLong:
                // Take action.
                break;
        }
        return false;
    }
}
```

Section 105.2: Surface

Touch event handler for surfaces (e.g. SurfaceView, GLSurfaceView, and others):

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.SurfaceView;
import android.view.View;

public class ExampleClass extends Activity implements View.OnTouchListener{
    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        CustomSurfaceView csv = new CustomSurfaceView(this);
        csv.setOnTouchListener(this);
        setContentView(csv);
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        // Add a switch (see buttons example) if you handle multiple views
        // here you can see (using MotionEvent event) to see what touch event
        // is being taken. Is the pointer touching or lifted? Is it moving?
        return false;
    }
}
```

Or alternatively (in the surface):

```
public class CustomSurfaceView extends SurfaceView {
    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        super.onTouchEvent(ev);
        // Handle touch events here. When doing this, you do not need to call a listener.
        // Please note that this listener only applies to the surface it is placed in
        // (in this case, CustomSurfaceView), which means that anything else which is
        // pressed outside the SurfaceView is handled by the parts of your app that
        // have a listener in that area.
        return true;
    }
}
```

Section 105.3: Handling multitouch in a surface

```
public class CustomSurfaceView extends SurfaceView {
    @Override
    public boolean onTouchEvent(MotionEvent e) {
        super.onTouchEvent(e);
        if(e.getPointerCount() > 2){
            return false; // If we want to limit the amount of pointers, we return false
                // which disallows the pointer. It will not be reacted on either, for
                // any future touch events until it has been lifted and repressed.
        }

        // What can you do here? Check if the amount of pointers are [x] and take action,
        // if a pointer leaves, a new enters, or the [x] pointers are moved.
        // Some examples as to handling etc. touch/motion events.

        switch (MotionEventCompat.getActionMasked(e)) {
            case MotionEvent.ACTION_DOWN:
            case MotionEvent.ACTION_POINTER_DOWN:
                // One or more pointers touch the screen.
                break;
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_POINTER_UP:
                // One or more pointers stop touching the screen.
                break;
            case MotionEvent.ACTION_MOVE:
                // One or more pointers move.
                if(e.getPointerCount() == 2){
                    move();
                }else if(e.getPointerCount() == 1){
                    paint();
                }else{
                    zoom();
                }
                break;
        }
        return true; // Allow repeated action.
    }
}
```

Chapter 106: Detect Shake Event in Android

Section 106.1: Shake Detector in Android Example

```
public class ShakeDetector implements SensorEventListener {


    private static final float SHAKE_THRESHOLD_GRAVITY = 2.7F;
    private static final int SHAKE_SLOP_TIME_MS = 500;
    private static final int SHAKE_COUNT_RESET_TIME_MS = 3000;

    private OnShakeListener mListener;
    private long mShakeTimestamp;
    private int mShakeCount;

    public void setOnShakeListener(OnShakeListener listener) {
        this.mListener = listener;
    }

    public interface OnShakeListener {
        public void onShake(int count);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // ignore
    }

    @Override
    public void onSensorChanged(SensorEvent event) {

        if (mListener != null) {
            float x = event.values[0];
            float y = event.values[1];
            float z = event.values[2];

            float gX = x / SensorManager.GRAVITY_EARTH;
            float gY = y / SensorManager.GRAVITY_EARTH;
            float gZ = z / SensorManager.GRAVITY_EARTH;

            // gForce will be close to 1 when there is no movement.
            float gForce = FloatMath.sqrt(gX * gX + gY * gY + gZ * gZ);

            if (gForce > SHAKE_THRESHOLD_GRAVITY) {
                final long now = System.currentTimeMillis();
                // ignore shake events too close to each other (500ms)
                if (mShakeTimestamp + SHAKE_SLOP_TIME_MS > now) {
                    return;
                }

                // reset the shake count after 3 seconds of no shakes
                if (mShakeTimestamp + SHAKE_COUNT_RESET_TIME_MS < now) {
                    mShakeCount = 0;
                }

                mShakeTimestamp = now;
                mShakeCount++;
```

```
                mListener.onShake(mShakeCount);
            }
        }
    }
}
```

Section 106.2: Using Seismic shake detection

Seismic is an Android device shake detection library by Square. To use it just start listening to the shake events emitted by it.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    sm = (SensorManager) getSystemService(SENSOR_SERVICE);
    sd = new ShakeDetector(() -> { /* react to detected shake */ });
}

@Override
protected void onResume() {
    sd.start(sm);
}

@Override
protected void onPause() {
    sd.stop();
}
```

To define the a different acceleration threshold use `sd.setSensitivity(sensitivity)` with a `sensitivity` of `SENSITIVITY_LIGHT`, `SENSITIVITY_MEDIUM`, `SENSITIVITY_HARD` or any other reasonable integer value. The given default values range from *11* to *15*.

Installation

```
compile 'com.squareup:seismic:1.0.2'
```

Chapter 107: Hardware Button Events/Intents (PTT, LWP, etc.)

Several android devices have custom buttons added by the manufacturer. This opens new possibilities for the developer in handling those buttons especially when making Apps targeted for Hardware Devices.

This topic documents buttons which have intents attached to them which you can listen for via intent-receivers.

Section 107.1: Sonim Devices

Sonim devices have varying by model a lot of different custom buttons:

PTT_KEY

```
com.sonim.intent.action.PTT_KEY_DOWN
com.sonim.intent.action.PTT_KEY_UP
```

YELLOW_KEY

```
com.sonim.intent.action.YELLOW_KEY_DOWN
com.sonim.intent.action.YELLOW_KEY_UP
```

SOS_KEY

```
com.sonim.intent.action.SOS_KEY_DOWN
com.sonim.intent.action.SOS_KEY_UP
```

GREEN_KEY

```
com.sonim.intent.action.GREEN_KEY_DOWN
com.sonim.intent.action.GREEN_KEY_UP
```

Registering the buttons

To receive those intents you will have to assign the buttons to your app in the Phone-Settings. Sonim has a possibilty to auto-register the buttons to the App when it is installed. In order to do that you will have to contact them and get a package-specific key to include in your Manifest like this:

```
<meta-data
   android:name="app_key_green_data"
   android:value="your-key-here" />
```

Section 107.2: RugGear Devices

PTT Button

```
android.intent.action.PTT.down
android.intent.action.PTT.up
```

Confirmed on: RG730, RG740A

Chapter 108: GreenRobot EventBus

Thread Mode	Description
`ThreadMode.POSTING`	Will be called on the same thread that the event was posted on. This is the default mode.
`ThreadMode.MAIN`	Will be called on the main UI thread.
`ThreadMode.BACKGROUND`	Will be called on a background thread. If the posting thread isn't the main thread it will be used. If posted on the main thread `EventBus` has a single background thread that it will use.
`ThreadMode.ASYNC`	Will be called on its own thread.

Section 108.1: Passing a Simple Event

The first thing we need to do it add EventBus to our module's gradle file:

```
dependencies {
    ...
    compile 'org.greenrobot:eventbus:3.0.0'
    ...
}
```

Now we need to create a model for our event. It can contain anything we want to pass along. For now we'll just make an empty class.

```
public class DeviceConnectedEvent
{
}
```

Now we can add the code to our `Activity` that will register with EventBus and subscribe to the event.

```
public class MainActivity extends AppCompatActivity
{
    private EventBus _eventBus;

    @Override
    protected void onCreate (Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        _eventBus = EventBus.getDefault();
    }

    @Override
    protected void onStart ()
    {
        super.onStart();
        _eventBus.register(this);
    }

    @Override
    protected void onStop ()
    {
        _eventBus.unregister(this);
        super.onStop();
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
```

```
    public void onDeviceConnected (final DeviceConnectedEvent event)
    {
        // Process event and update UI
    }
}
```

In this `Activity` we get an instance of `EventBus` in the `onCreate()` method. We register / unregister for events in `onStart()` / `onStop()`. It's important to remember to unregister when your listener loses scope or you could leak your `Activity`.

Finally we define the method that we want called with the event. The `@Subscribe` annotation tells EventBus which methods it can look for to handle events. You have to have at least one methods annotated with `@Subscribe` to register with EventBus or it will throw an exception. In the annotation we define the thread mode. This tells EventBus which thread to call the method on. It is a very handy way of passing information from a background thread to the UI thread! That's exactly what we're doing here. `ThreadMode.MAIN` means that this method will be called on Android's main UI thread so it's safe to do any UI manipulations here that you need. The name of the method doesn't matter. The only think, other that the `@Subscribe` annotation, that EventBus is looking for is the type of the argument. As long as the type matches it will be called when an event is posted.

The last thing we need to do it to post an event. This code will be in our `Service`.

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

That's all there is to it! EventBus will take that DeviceConnectedEvent and look through its registered listeners, look through the methods that they've subscribed and find the ones that take a DeviceConnectedEvent as an argument and call them on the thread that they want to be called on.

Section 108.2: Receiving Events

For receiving events you need to register your class on the `EventBus`.

```
@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}

@Override
public void onStop() {
   EventBus.getDefault().unregister(this);
   super.onStop();
}
```

And then subscribe to the events.

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitraryEvent event) {
    Toast.makeText(getActivity(), "Event type: "+event.getEventType(), Toast.LENGTH_SHORT).show();
}
```

Section 108.3: Sending Events

Sending events is as easy as creating the Event object and then posting it.

```
EventBus.getDefault().post(new ArbitraryEvent(ArbitraryEvent.TYPE_1));
```

Chapter 109: Otto Event Bus

Section 109.1: Passing an event

This example describes passing an event using the Otto Event Bus.

To use the Otto Event Bus in **Android Studio** you have to insert the following statement in your modules gradle file:

```
dependencies {
    compile 'com.squareup:otto:1.3.8'
}
```

The event we'd like to pass is a simple Java object:

```
public class DatabaseContentChangedEvent {
    public String message;

    public DatabaseContentChangedEvent(String message) {
        this.message = message;
    }
}
```

We need a Bus to send events. This is typically a singleton:

```
import com.squareup.otto.Bus;

public final class BusProvider {
    private static final Bus mBus = new Bus();

    public static Bus getInstance() {
        return mBus;
    }

    private BusProvider() {
    }
}
```

To send an event we only need our BusProvider and it's `post` method. Here we send an event if the action of an AsyncTask is completed:

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {

    ...

    @Override
    protected void onPostExecute(Void param) {
        BusProvider.getInstance().post(
            new DatabaseContentChangedEvent("Content changed")
        );
    }
}
```

Section 109.2: Receiving an event

To receive an event it is necessary to implement a method with the event type as parameter and annotate it using `@Subscribe`. Furthermore you have to register/unregister the instance of your object at the `BusProvider` (see

example *Sending an event*):

```
public class MyFragment extends Fragment {
    private final static String TAG = "MyFragment";

    ...

    @Override
    public void onResume() {
        super.onResume();
        BusProvider.getInstance().register(this);
    }

    @Override
    public void onPause() {
        super.onPause();
        BusProvider.getInstance().unregister(this);
    }

    @Subscribe
    public void onDatabaseContentChanged(DatabaseContentChangedEvent event) {
        Log.i(TAG, "onDatabaseContentChanged: "+event.message);
    }
}
```

Important: In order to receive that event an instance of the class has to exist. This is usually not the case when you want to send a result from one activity to another activity. So check your use case for the event bus.

Chapter 110: Vibration

Section 110.1: Getting Started with Vibration

Grant Vibration Permission

before you start implement code, you have to add permission in android manifest :

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

Import Vibration Library

```
import android.os.Vibrator;
```

Get instance of Vibrator from Context

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Check device has vibrator

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

Section 110.2: Vibrate Indefinitely

using the *vibrate(long[] pattern, int repeat)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

// Start time delay
// Vibrate for 500 milliseconds
// Sleep for 1000 milliseconds
long[] pattern = {0, 500, 1000};

// 0 meaning is repeat indefinitely
vibrator.vibrate(pattern, 0);
```

Section 110.3: Vibration Patterns

You can create vibration patterns by passing in an array of longs, each of which represents a duration in milliseconds. The first number is start time delay. Each array entry then alternates between vibrate, sleep, vibrate, sleep, etc.

The following example demonstrates this pattern:

- vibrate 100 milliseconds and sleep 1000 milliseconds
- vibrate 200 milliseconds and sleep 2000 milliseconds

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

To cause the pattern to repeat, pass in the index into the pattern array at which to start the repeat, or `-1` to disable repeating.

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // does not repeat
vibrator.vibrate(pattern,  0); // repeats forever
```

Section 110.4: Stop Vibrate

If you want stop vibrate please call :

```
vibrator.cancel();
```

Section 110.5: Vibrate for one time

using the *vibrate(long milliseconds)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

Chapter 111: ContentProvider

Section 111.1: Implementing a basic content provider class

1) Create a Contract Class

A contract class defines constants that help applications work with the content URIs, column names, intent actions, and other features of a content provider. Contract classes are not included automatically with a provider; the provider's developer has to define them and then make them available to other developers.

A provider usually has a single authority, which serves as its Android-internal name. To avoid conflicts with other providers, use a unique content authority. Because this recommendation is also true for Android package names, you can define your provider authority as an extension of the name of the package containing the provider. For example, if your Android package name is `com.example.appname`, you should give your provider the authority `com.example.appname.provider`.

```
public class MyContract {
    public static final String CONTENT_AUTHORITY = "com.example.myApp";
    public static final String PATH_DATATABLE = "dataTable";
    public static final String TABLE_NAME = "dataTable";
}
```

A content URI is a URI that identifies data in a provider. Content URIs include the symbolic name of the entire provider (its authority) and a name that points to a table or file (a path). The optional id part points to an individual row in a table. Every data access method of ContentProvider has a content URI as an argument; this allows you to determine the table, row, or file to access. Define these in the contract class.

```
public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);
public static final Uri CONTENT_URI =
BASE_CONTENT_URI.buildUpon().appendPath(PATH_DATATABLE).build();

// define all columns of table and common functions required
```

2) Create the Helper Class

A helper class manages database creation and version management.

```
public class DatabaseHelper extends SQLiteOpenHelper {

    // Increment the version when there is a change in the structure of database
    public static final int DATABASE_VERSION = 1;
    // The name of the database in the filesystem, you can choose this to be anything
    public static final String DATABASE_NAME = "weather.db";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Called when the database is created for the first time. This is where the
        // creation of tables and the initial population of the tables should happen.
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

```
        // Called when the database needs to be upgraded. The implementation
        // should use this method to drop tables, add tables, or do anything else it
        // needs to upgrade to the new schema version.
    }
}
```

3) Create a class that extends ContentProvider class

```
public class MyProvider extends ContentProvider {

public DatabaseHelper dbHelper;

public static final UriMatcher matcher = buildUriMatcher();
public static final int DATA_TABLE = 100;
public static final int DATA_TABLE_DATE = 101;
```

A UriMatcher maps an authority and path to an integer value. The method `match()` returns a unique integer value for a URI (it can be any arbitrary number, as long as it's unique). A switch statement chooses between querying the entire table, and querying for a single record. Our UriMatcher returns 100 if the URI is the Content URI of Table and 101 if the URI points to a specific row within that table. You can use the `#` wildcard to match with any number and `*` to match with any string.

```
public static UriMatcher buildUriMatcher() {
    UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    uriMatcher.addURI(CONTENT_AUTHORITY, MyContract.PATH_DATATABLE,        DATA_TABLE);
    uriMatcher.addURI(CONTENT_AUTHORITY, MyContract.PATH_DATATABLE + "/#", DATA_TABLE_DATE);
    return uriMatcher;
}
```

IMPORTANT: the ordering of `addURI()` calls matters! The UriMatcher will look in sequential order from first added to last. Since wildcards like `#` and `*` are greedy, you will need to make sure that you have ordered your URIs correctly. For example:

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);
uriMatcher.addURI(CONTENT_AUTHORITY, "/*",       2);
```

is the proper ordering, since the matcher will look for `/example` first before resorting to the `/*` match. If these method calls were reversed and you called `uriMatcher.match("/example")`, then the UriMatcher will stop looking for matches once it encounters the `/*` path and return the wrong result!

You will then need to override these functions:

onCreate(): Initialize your provider. The Android system calls this method immediately after it creates your provider. Notice that your provider is not created until a ContentResolver object tries to access it.

```
@Override
public boolean onCreate() {
    dbhelper = new DatabaseHelper(getContext());
    return true;
}
```

getType(): Return the MIME type corresponding to a content URI

```
@Override
public String getType(Uri uri) {
    final int match = matcher.match(uri);
```

```
    switch (match) {
        case DATA_TABLE:
            return ContentResolver.CURSOR_DIR_BASE_TYPE + "/" + MyContract.CONTENT_AUTHORITY + "/"
+ MyContract.PATH_DATATABLE;
        case DATA_TABLE_DATE:
            return ContentResolver.ANY_CURSOR_ITEM_TYPE + "/" + MyContract.CONTENT_AUTHORITY + "/"
+ MyContract.PATH_DATATABLE;
        default:
            throw new UnsupportedOperationException("Unknown Uri: " + uri);

    }
}
```

query(): Retrieve data from your provider. Use the arguments to select the table to query, the rows and columns to return, and the sort order of the result. Return the data as a Cursor object.

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String
sortOrder) {
    Cursor retCursor = dbHelper.getReadableDatabase().query(
        MyContract.TABLE_NAME, projection, selection, selectionArgs, null, null, sortOrder);
    retCursor.setNotificationUri(getContext().getContentResolver(), uri);
    return retCursor;
}
```

Insert a new row into your provider. Use the arguments to select the destination table and to get the column values to use. Return a content URI for the newly-inserted row.

```
@Override
public Uri insert(Uri uri, ContentValues values)
{
    final SQLiteDatabase db = dbHelper.getWritableDatabase();
    long id = db.insert(MyContract.TABLE_NAME, null, values);
    return ContentUris.withAppendedId(MyContract.CONTENT_URI, ID);
}
```

delete(): Delete rows from your provider. Use the arguments to select the table and the rows to delete. Return the number of rows deleted.

```
@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int rowsDeleted = db.delete(MyContract.TABLE_NAME, selection, selectionArgs);
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsDeleted;
}
```

update(): Update existing rows in your provider. Use the arguments to select the table and rows to update and to get the new column values. Return the number of rows updated.

```
@Override
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    int rowsUpdated = db.update(MyContract.TABLE_NAME, values, selection, selectionArgs);
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsUpdated;
}
```

4) Update manifest file

```
<provider
       android:authorities="com.example.myApp"
       android:name=".DatabaseProvider"/>
```

Chapter 112: Dagger 2

Section 112.1: Component setup for Application and Activity injection

A basic `AppComponent` that depends on a single `AppModule` to provide application-wide singleton objects.

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {

    void inject(App app);

    Context provideContext();

    Gson provideGson();
}
```

A module to use together with the `AppComponent` which will provide its singleton objects, e.g. an instance of `Gson` to reuse throughout the whole application.

```
@Module
public class AppModule {

    private final Application mApplication;

    public AppModule(Application application) {
        mApplication = application;
    }

    @Singleton
    @Provides
    Gson provideGson() {
        return new Gson();
    }

    @Singleton
    @Provides
    Context provideContext() {
        return mApplication;
    }
}
```

A subclassed application to setup dagger and the singleton component.

```
public class App extends Application {

    @Inject
    AppComponent mAppComponent;

    @Override
    public void onCreate() {
        super.onCreate();

        DaggerAppComponent.builder().appModule(new AppModule(this)).build().inject(this);
    }

    public AppComponent getAppComponent() {
```

```
        return mAppComponent;
    }
}
```

Now an activity scoped component that depends on the `AppComponent` to gain access to the singleton objects.

```
@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {

    void inject(MainActivity activity);
}
```

And a reusable `ActivityModule` that will provide basic dependencies, like a `FragmentManager`

```
@Module
public class ActivityModule {

    private final AppCompatActivity mActivity;

    public ActivityModule(AppCompatActivity activity) {
        mActivity = activity;
    }

    @ActivityScope
    public AppCompatActivity provideActivity() {
        return mActivity;
    }


    @ActivityScope
    public FragmentManager provideFragmentManager(AppCompatActivity activity) {
        return activity.getSupportFragmentManager();
    }
}
```

Putting everything together we're set up and can inject our activity and be sure to use the same `Gson` throughout out app!

```
public class MainActivity extends AppCompatActivity {

    @Inject
    Gson mGson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        DaggerMainActivityComponent.builder()
                .appComponent(((App)getApplication()).getAppComponent())
                .activityModule(new ActivityModule(this))
                .build().inject(this);
    }
}
```

Section 112.2: Custom Scopes

```
@Scope
```

```
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

Scopes are just annotations and you can create your own ones where needed.

Section 112.3: Using @Subcomponent instead of @Component(dependencies={...})

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {
    void inject(App app);

    Context provideContext();
    Gson provideGson();

    MainActivityComponent mainActivityComponent(ActivityModule activityModule);
}

@ActivityScope
@Subcomponent(modules = ActivityModule.class)
public interface MainActivityComponent {
    void inject(MainActivity activity);
}

public class MainActivity extends AppCompatActivity {

    @Inject
    Gson mGson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ((App)getApplication()).getAppComponent()
                .mainActivityComponent(new ActivityModule(this)).inject(this);
    }
}
```

Section 112.4: Creating a component from multiple modules

Dagger 2 supports creating a component from multiple modules. You can create your component this way:

```
@Singleton
@Component(modules = {GeneralPurposeModule.class, SpecificModule.class})
public interface MyMultipleModuleComponent {
    void inject(MyFragment myFragment);
    void inject(MyService myService);
    void inject(MyController myController);
    void inject(MyActivity myActivity);
}
```

The two references modules `GeneralPurposeModule` and `SpecificModule` can then be implemented as follows:

GeneralPurposeModule.java

```
@Module
public class GeneralPurposeModule {
    @Provides
    @Singleton
    public Retrofit getRetrofit(PropertiesReader propertiesReader, RetrofitHeaderInterceptor
headerInterceptor){
        // Logic here...
        return retrofit;
    }

    @Provides
    @Singleton
    public PropertiesReader getPropertiesReader(){
        return new PropertiesReader();
    }

    @Provides
    @Singleton
    public RetrofitHeaderInterceptor getRetrofitHeaderInterceptor(){
         return new RetrofitHeaderInterceptor();
    }
}
```

SpecificModule.java

```
@Singleton
@Module
public class SpecificModule {
    @Provides @Singleton
    public RetrofitController getRetrofitController(Retrofit retrofit){
        RetrofitController retrofitController = new RetrofitController();
        retrofitController.setRetrofit(retrofit);
        return retrofitController;
    }

    @Provides @Singleton
    public MyService getMyService(RetrofitController retrofitController){
        MyService myService = new MyService();
        myService.setRetrofitController(retrofitController);
        return myService;
    }
}
```

During the dependency injection phase, the component will take objects from both modules according to the needs.

This approach is very useful in terms of *modularity*. In the example, there is a general purpose module used to instantiate components such as the `Retrofit` object (used to handle the network communication) and a `PropertiesReader` (in charge of handling configuration files). There is also a specific module that handles the instantiation of specific controllers and service classes in relation to that specific application component.

Section 112.5: How to add Dagger 2 in build.gradle

Since the release of Gradle 2.2, the use of the android-apt plugin is no longer used. The following method of setting up Dagger 2 should be used. For older version of Gradle, use the previous method shown below.

For Gradle >= 2.2

```
dependencies {
    // apt command comes from the android-apt plugin
    annotationProcessor 'com.google.dagger:dagger-compiler:2.8'
    compile 'com.google.dagger:dagger:2.8'
    provided 'javax.annotation:jsr250-api:1.0'
}
```

For Gradle < 2.2

To use Dagger 2 it's necessary to add `android-apt` plugin, add this to the root build.gradle:

```
buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

Then the application module's build.gradle should contain:

```
apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'


android {
    ...
}

final DAGGER_VERSION = '2.0.2'
dependencies {
    ...

    compile "com.google.dagger:dagger:${DAGGER_VERSION}"
    apt "com.google.dagger:dagger-compiler:${DAGGER_VERSION}"
}
```

Reference: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Section 112.6: Constructor Injection

Classes without dependencies can easily be created by dagger.

```
public class Engine {

    @Inject // <-- Annotate your constructor.
    public Engine() {
    }
}
```

This class can be provided by *any* component. It has *no dependencies itself* and is *not scoped*. There is no further code necessary.

Dependencies are declared as parameters in the constructor. Dagger will call the constructor and supply the dependencies, as long as those dependencies can be provided.

```
public class Car {
```

```
    private Engine engine;

    @Inject
    public Car(Engine engine) {
        this.engine = engine;
    }
}
```

This class can be provided by every component *iff* this component can also provide all of its dependencies—`Engine` in this case. Since `Engine` can also be constructor injected, *any* component can provide a `Car`.

You can use constructor injection whenever all of the dependencies can be provided by the component. A component can provide a dependency, if

- it can create it by using constructor injection
- a module of the component can provide it
- it can be provided by the parent component (if it is a `@Subcomponent`)
- it can use an object exposed by a component it depends on (component dependencies)

Chapter 113: Realm

Realm Mobile Database is an alternative to SQLite. Realm Mobile Database is much faster than an ORM, and often faster than raw SQLite.

Benefits

Offline functionality, Fast queries, Safe threading, Cross-platform apps, Encryption, Reactive architecture.

Section 113.1: Sorted queries

In order to sort a query, instead of using `findAll()`, you should use `findAllSorted()`.

```
RealmResults<SomeObject> results = realm.where(SomeObject.class)
                                        .findAllSorted("sortField", Sort.ASCENDING);
```

Note:

`sort()` returns a completely new RealmResults that is sorted, but an update to this RealmResults will reset it. If you use `sort()`, you should always re-sort it in your `RealmChangeListener`, remove the `RealmChangeListener` from the previous `RealmResults` and add it to the returned new `RealmResults`. Using `sort()` on a `RealmResults` returned by an async query that is not yet loaded will fail.

`findAllSorted()` will always return the results sorted by the field, even if it gets updated. It is recommended to use `findAllSorted()`.

Section 113.2: Using Realm with RxJava

For queries, Realm provides the `realmResults.asObservable()` method. Observing results is only possible on looper threads (typically the UI thread).

For this to work, your configuration must contain the following

```
realmConfiguration = new RealmConfiguration.Builder(context)       //
                         .rxFactory(new RealmObservableFactory()) //
                           //...
                         .build();
```

Afterwards, you can use your results as an observable.

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

For asynchronous queries, you should filter the results by `isLoaded()`, so that you receive an event only when the query has been executed. This `filter()` is not needed for synchronous queries (`isLoaded()` always returns **`true`** on sync queries).

```
    Subscription subscription = RxTextView.textChanges(editText).switchMap(charSequence ->
        realm.where(SomeObject.class)
             .contains("searchField", charSequence.toString(), Case.INSENSITIVE)
             .findAllAsync()
             .asObservable())
    .filter(RealmResults::isLoaded) //
    .subscribe(objects -> adapter.updateData(objects));
```

For writes, you should either use the `executeTransactionAsync()` method, or open a Realm instance on the background thread, execute the transaction synchronously, then close the Realm instance.

```
public Subscription loadObjectsFromNetwork() {
    return objectApi.getObjects()
        .subscribeOn(Schedulers.io())
        .subscribe(response -> {
            try(Realm realmInstance = Realm.getDefaultInstance()) {
                realmInstance.executeTransaction(realm -> realm.insertOrUpdate(response.objects));
            }
        });
}
```

Section 113.3: Basic Usage

Setting up an instance

To use Realm you first need to obtain an instance of it. Each Realm instance maps to a file on disk. The most basic way to get an instance is as follows:

```
// Create configuration
RealmConfiguration realmConfiguration = new RealmConfiguration.Builder(context).build();

// Obtain realm instance
Realm realm = Realm.getInstance(realmConfiguration);
// or
Realm.setDefaultConfiguration(realmConfiguration);
Realm realm = Realm.getDefaultInstance();
```

The method `Realm.getInstance()` creates the database file if it has not been created, otherwise opens the file. The `RealmConfiguration` object controls all aspects of how a Realm is created - whether it's an `inMemory()` database, name of the Realm file, if the Realm should be cleared if a migration is needed, initial data, etc.

Please note that calls to `Realm.getInstance()` are reference counted (each call increments a counter), and the counter is decremented when `realm.close()` is called.

Closing an instance

On background threads, it's *very important* to ***close*** the Realm instance(s) once it's no longer used (for example, transaction is complete and the thread execution ends). Failure to close all Realm instances on background thread results in version pinning, and can cause a large growth in file size.

```
Runnable runnable = new Runnable() {
    Realm realm = null;
    try {
        realm = Realm.getDefaultInstance();
        // ...
    } finally {
        if(realm != null) {
            realm.close();
        }
    }
};

new Thread(runnable).start(); // background thread, like `doInBackground()` of AsyncTask
```

It's worth noting that above API Level 19, you can replace this code with just this:

```
try(Realm realm = Realm.getDefaultInstance()) {
    // ...
}
```

Models

Next step would be creating your models. Here a question might be asked, "what is a model?". A model is a structure which defines properties of an object being stored in the database. For example, in the following we model a book.

```
public class Book extends RealmObject {

    // Primary key of this entity
    @PrimaryKey
    private long id;

    private String title;

    @Index // faster queries
    private String author;

    // Standard getters & setter
    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }
}
```

Note that your models should extend RealmObject class. Primary key is also specified by `@PrimaryKey` annotation. Primary keys can be null, but only one element can have **`null`** as a primary key. Also you can use the `@Ignore` annotation for the fields that should not be persisted to the disk:

```
@Ignore
private String isbn;
```

Inserting or updating data

In order to store a book object to your Realm database instance, you can first create an instance of your model and then store it to the database via `copyToRealm` method. For creating or updating you can use `copyToRealmOrUpdate`. (A faster alternative is the newly added `insertOrUpdate()`).

```
// Creating an instance of the model
Book book = new Book();
book.setId(1);
book.setTitle("Walking on air");
book.setAuthor("Taylor Swift")

// Store to the database
realm.executeTransaction(new Realm.Transaction() {
    @Override
    public void execute(Realm realm) {
        realm.insertOrUpdate(book);
    }
});
```

Note that all changes to data must happen in a transaction. Another way to create an object is using the following pattern:

```
Book book = realm.createObject(Book.class, primaryKey);
...
```

Querying the database

- All books:

```
RealmResults<Book> results = realm.where(Book.class).findAll();
```

- All books having id greater than 10:

```
RealmResults<Book> results = realm.where(Book.class)
                                  .greaterThan("id", 10)
                                  .findAll();
```

- Books by `'Taylor Swift'` or `'%Peter%'`:

```
RealmResults<Book> results = realm.where(Book.class)
                                  .beginGroup()
                                      .equalTo("author", "Taylor Swift")
                                      .or()
                                      .contains("author", "Peter")
                                  .endGroup().findAll();
```

Deleting an object

For example, we want to delete all books by Taylor Swift:

```
// Start of transaction
realm.executeTransaction(new Realm.Transaction() {
    @Override
    public void execute(Realm realm) {
        // First Step: Query all Taylor Swift books
        RealmResults<Book> results = ...

        // Second Step: Delete elements in Realm
        results.deleteAllFromRealm();
    }
});
```

Section 113.4: List of primitives (RealmList<Integer/String/...>)

Realm currently does not support storing a list of primitives. It is on their todo list (GitHub issue #575), but for the meantime, here is a workaround.

Create a new class for your primitive type, this uses Integer, but change it for whatever you want to store.

```
public class RealmInteger extends RealmObject {
    private int val;

    public RealmInteger() {
    }

    public RealmInteger(int val) {
        this.val = val;
    }

    // Getters and setters
}
```

You can now use this in your `RealmObject`.

```
public class MainObject extends RealmObject {

    private String name;
    private RealmList<RealmInteger> ints;

    // Getters and setters
}
```

If you are using `GSON` to populate your `RealmObject`, you will need to add a custom type adapter.

```
Type token = new TypeToken<RealmList<RealmInteger>>(){}.getType();
Gson gson = new GsonBuilder()
        .setExclusionStrategies(new ExclusionStrategy() {
            @Override
            public boolean shouldSkipField(FieldAttributes f) {
                return f.getDeclaringClass().equals(RealmObject.class);
            }

            @Override
            public boolean shouldSkipClass(Class<?> clazz) {
                return false;
            }
        })
        .registerTypeAdapter(token, new TypeAdapter<RealmList<RealmInteger>>() {

            @Override
            public void write(JsonWriter out, RealmList<RealmInteger> value) throws IOException {
                // Empty
            }

            @Override
            public RealmList<RealmInteger> read(JsonReader in) throws IOException {
                RealmList<RealmInteger> list = new RealmList<RealmInteger>();
                in.beginArray();
                while (in.hasNext()) {
                    list.add(new RealmInteger(in.nextInt()));
                }
                in.endArray();
```

```
                return list;
            }
        })
        .create();
```

Section 113.5: Async queries

Every synchronous query method (such as `findAll()` or `findAllSorted()`) has an asynchronous counterpart (`findAllAsync()` / `findAllSortedAsync()`).

Asynchronous queries offload the evaluation of the `RealmResults` to another thread. In order to receive these results on the current thread, the current thread must be a looper thread (read: async queries typically only work on the UI thread).

```
RealmChangeListener<RealmResults<SomeObject>> realmChangeListener; // field variable

realmChangeListener = new RealmChangeListener<RealmResults<SomeObject>>() {
    @Override
    public void onChange(RealmResults<SomeObject> element) {
        // asyncResults are now loaded
        adapter.updateData(element);
    }
};

RealmResults<SomeObject> asyncResults = realm.where(SomeObject.class).findAllAsync();
asyncResults.addChangeListener(realmChangeListener);
```

Section 113.6: Adding Realm to your project

Add the following dependency to your **project** level `build.gradle` file.

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

Add the following right at the top of your **app** level `build.gradle` file.

```
apply plugin: 'realm-android'
```

Complete a gradle sync and you now have Realm added as a dependency to your project!

Realm requires an initial call since 2.0.0 before using it. You can do this in your `Application` class or in your first Activity's `onCreate` method.

```
Realm.init(this); // added in Realm 2.0.0
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

Section 113.7: Realm Models

Realm models must extend the `RealmObject` base class, they define the schema of the underlying database.

Supported field types are **`boolean`**, **`byte`**, **`short`**, **`int`**, **`long`**, **`float`**, **`double`**, `String`, `Date`, **`byte`**`[]`, links to other `RealmObjects`, and `RealmList<T extends RealmModel>`.

```
public class Person extends RealmObject {
```

```
    @PrimaryKey //primary key is also implicitly an @Index
                //it is required for `copyToRealmOrUpdate()` to update the object.
    private long id;

    @Index //index makes queries faster on this field
    @Required //prevents `null` value from being inserted
    private String name;

    private RealmList<Dog> dogs; //->many relationship to Dog

    private Person spouse; //->one relationship to Person

    @Ignore
    private Calendar birthday; //calendars are not supported but can be ignored

    // getters, setters
}
```

If you add (or remove) a new field to your RealmObject (or you add a new RealmObject class or delete an existing one), a **migration** will be needed. You can either set `deleteIfMigrationNeeded()` in your `RealmConfiguration.Builder`, or define the necessary migration. Migration is also required when adding (or removing) `@Required`, or `@Index`, or `@PrimaryKey` annotation.

Relationships must be set manually, they are NOT automatic based on primary keys.

Since 0.88.0, it is also possible to use public fields instead of private fields/getters/setters in RealmObject classes.

It is also possible to implement `RealmModel` instead of extending `RealmObject`, if the class is also annotated with `@RealmClass`.

```
@RealmClass
public class Person implements RealmModel {
    // ...
}
```

In that case, methods like `person.deleteFromRealm()` or `person.addChangeListener()` are replaced with `RealmObject.deleteFromRealm(person)` and `RealmObject.addChangeListener(person)`.

Limitations are that by a `RealmObject`, only `RealmObject` can be extended, and there is no support for **`final`**, **`volatile`** and **`transient`** fields.

It is important that a *managed* RealmObject class can only be modified in a transaction. A *managed* RealmObject cannot be passed between threads.

Section 113.8: try-with-resources

```
try (Realm realm = Realm.getDefaultInstance()) {
    realm.executeTransaction(new Realm.Transaction() {
            @Override
            public void execute(Realm realm) {
                //whatever Transaction that has to be done
            }
    });
    //No need to close realm in try-with-resources
}
```

The Try with resources can be used only from KITKAT (minSDK 19)

Chapter 114: Android Versions

Section 114.1: Checking the Android Version on device at runtime

`Build.VERSION_CODES` is an enumeration of the currently known SDK version codes.

In order to conditionally run code based on the device's Android version, use the `TargetApi` annotation to avoid Lint errors, and check the build version before running the code specific to the API level.

Here is an example of how to use a class that was introduced in API-23, in a project that supports API levels lower than 23:

```
@Override
@TargetApi(23)
public void onResume() {
    super.onResume();
    if (android.os.Build.VERSION.SDK_INT <= Build.VERSION_CODES.M) {
        //run Marshmallow code
        FingerprintManager fingerprintManager = this.getSystemService(FingerprintManager.class);
        //.....................
    }
}
```

Chapter 115: Wi-Fi Connections

Section 115.1: Connect with WEP encryption

This example connects to a Wi-Fi access point with WEP encryption, given an SSID and the password.

```
public boolean ConnectToNetworkWEP(String networkSSID, String password)
{
    try {
        WifiConfiguration conf = new WifiConfiguration();
        conf.SSID = "\"" + networkSSID + "\"";   // Please note the quotes. String should contain
SSID in quotes
        conf.wepKeys[0] = "\"" + password + "\""; //Try it with quotes first

        conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
        conf.allowedGroupCiphers.set(WifiConfiguration.AuthAlgorithm.OPEN);
        conf.allowedGroupCiphers.set(WifiConfiguration.AuthAlgorithm.SHARED);

        WifiManager wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        int networkId = wifiManager.addNetwork(conf);

        if (networkId == -1){
            //Try it again with no quotes in case of hex password
            conf.wepKeys[0] = password;
            networkId = wifiManager.addNetwork(conf);
        }

        List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
        for( WifiConfiguration i : list ) {
            if(i.SSID != null && i.SSID.equals("\"" + networkSSID + "\"")) {
                wifiManager.disconnect();
                wifiManager.enableNetwork(i.networkId, true);
                wifiManager.reconnect();
                break;
            }
        }

        //WiFi Connection success, return true
        return true;
    } catch (Exception ex) {
        System.out.println(Arrays.toString(ex.getStackTrace()));
        return false;
    }
}
```

Section 115.2: Connect with WPA2 encryption

This example connects to a Wi-Fi access point with WPA2 encryption.

```
public boolean ConnectToNetworkWPA(String networkSSID, String password) {
  try {
    WifiConfiguration conf = new WifiConfiguration();
    conf.SSID = "\"" + networkSSID + "\"";   // Please note the quotes. String should contain SSID
in quotes

    conf.preSharedKey = "\"" + password + "\"";

    conf.status = WifiConfiguration.Status.ENABLED;
```

```
    conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
    conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
    conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
    conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP);
    conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMP);

    Log.d("connecting", conf.SSID + " " + conf.preSharedKey);

    WifiManager wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
    wifiManager.addNetwork(conf);

    Log.d("after connecting", conf.SSID + " " + conf.preSharedKey);

    List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
    for( WifiConfiguration i : list ) {
      if(i.SSID != null && i.SSID.equals("\"" + networkSSID + "\"")) {
        wifiManager.disconnect();
        wifiManager.enableNetwork(i.networkId, true);
        wifiManager.reconnect();
        Log.d("re connecting", i.SSID + " " + conf.preSharedKey);

        break;
      }
    }

    //WiFi Connection success, return true
    return true;
  } catch (Exception ex) {
    System.out.println(Arrays.toString(ex.getStackTrace()));
    return false;
  }
}
```

Section 115.3: Scan for access points

This example scans for available access points and ad hoc networks. `btnScan` activates a scan initiated by the `WifiManager.startScan()` method. After the scan, `WifiManager` calls the `SCAN_RESULTS_AVAILABLE_ACTION` intent and the `WifiScanReceiver` class processes the scan result. The results are displayed in a `TextView`.

```
public class MainActivity extends AppCompatActivity {

    private final static String TAG = "MainActivity";

    TextView txtWifiInfo;
    WifiManager wifi;
    WifiScanReceiver wifiReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        wifi=(WifiManager)getSystemService(Context.WIFI_SERVICE);
        wifiReceiver = new WifiScanReceiver();

        txtWifiInfo = (TextView)findViewById(R.id.txtWifiInfo);
        Button btnScan = (Button)findViewById(R.id.btnScan);
        btnScan.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                Log.i(TAG, "Start scan...");
                wifi.startScan();
            }
        });
    }

    protected void onPause() {
        unregisterReceiver(wifiReceiver);
        super.onPause();
    }

    protected void onResume() {
        registerReceiver(
            wifiReceiver,
            new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)
        );
        super.onResume();
    }

    private class WifiScanReceiver extends BroadcastReceiver {
        public void onReceive(Context c, Intent intent) {
            List<ScanResult> wifiScanList = wifi.getScanResults();
            txtWifiInfo.setText("");
            for(int i = 0; i < wifiScanList.size(); i++){
                String info = ((wifiScanList.get(i)).toString());
                txtWifiInfo.append(info+"\n\n");
            }
        }
    }
}
```

Permissions

The following permissions need to be defined in *AndroidManifest.xml*:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

`android.permission.ACCESS_WIFI_STATE` is necessary for calling `WifiManager.getScanResults()`. Without `android.permission.CHANGE_WIFI_STATE` you cannot initiate a scan with `WifiManager.startScan()`.

When compiling the project for api level 23 or greater (Android 6.0 and up), either `android.permission.ACCESS_FINE_LOCATION` or `android.permission.ACCESS_COARSE_LOCATION` must be inserted. Furthermore that permission needs to be requested, e.g. in the `onCreate` method of your main activity:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
            Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}
```

Chapter 116: SensorManager

Section 116.1: Decide if your device is static or not, using the accelerometer

Add the following code to the `onCreate()`/`onResume()` method:

```
SensorManager sensorManager;
Sensor mAccelerometer;
final float movementThreshold = 0.5f;  // You may have to change this value.
boolean isMoving = false;
float[] prevValues = {1.0f, 1.0f, 1.0f};
float[] currValues = new float[3];

sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
mAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
```

You may have to adjust the sensitivity by adapting the `movementThreshold` by trial and error. Then, override the `onSensorChanged()` method as follows:

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor == mAccelerometer) {
        System.arraycopy(event.values, 0, currValues, 0, event.values.length);
        if ((Math.abs(currValues[0] - prevValues[0]) > movementThreshold) ||
                (Math.abs(currValues[1] - prevValues[1]) > movementThreshold) ||
                (Math.abs(currValues[2] - prevValues[2]) > movementThreshold)) {
            isMoving = true;
        } else {
            isMoving = false;
        }
        System.arraycopy(currValues, 0, prevValues, 0, currValues.length);
    }
}
```

If you want to prevent your app from being installed on devices that do not have an accelerometer, you have to add the following line to your manifest:

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

Section 116.2: Retrieving sensor events

Retrieving sensor information from the onboard sensors:

```
public class MainActivity extends Activity implements SensorEventListener {

    private SensorManager mSensorManager;
    private Sensor accelerometer;
    private Sensor gyroscope;

    float[] accelerometerData = new float[3];
    float[] gyroscopeData = new float[3];

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

        accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        gyroscope = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);

    }

    @Override
    public void onResume() {
        //Register listeners for your sensors of interest
        mSensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_FASTEST);
        mSensorManager.registerListener(this, gyroscope, SensorManager.SENSOR_DELAY_FASTEST);
        super.onResume();
    }

    @Override
    protected void onPause() {
        //Unregister any previously registered listeners
        mSensorManager.unregisterListener(this);
        super.onPause();
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        //Check the type of sensor data being polled and store into corresponding float array
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            accelerometerData = event.values;
        } else if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) {
            gyroscopeData = event.values;
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // TODO Auto-generated method stub
    }
}
```

Section 116.3: Sensor transformation to world coordinate system

The sensor values returned by Android are with respective to the phone's coordinate system (e.g. +Y points towards the top of the phone). We can transform these sensor values into a world coordinate system (e.g. +Y points towards magnetic North, tangential to the ground) using the sensor managers rotation matrix

First, you would need to declare and initialize the matrices/arrays where data will be stored (you can do this in the `onCreate` method, for example):

```
float[] accelerometerData = new float[3];
float[] accelerometerWorldData = new float[3];
float[] gravityData = new float[3];
float[] magneticData = new float[3];
float[] rotationMatrix = new float[9];
```

Next, we need to detect changes in sensor values, store them into the corresponding arrays (if we want to use them later/elsewhere), then calculate the rotation matrix and resulting transformation into world coordinates:

```
public void onSensorChanged(SensorEvent event) {
    sensor = event.sensor;
    int i = sensor.getType();

    if (i == Sensor.TYPE_ACCELEROMETER) {
        accelerometerData = event.values;
    } else if (i == Sensor.TYPE_GRAVITY) {
        gravityData = event.values;
    } else if (i == Sensor.TYPE_MAGNETIC) {
        magneticData = event.values;
    }

    //Calculate rotation matrix from gravity and magnetic sensor data
    SensorManager.getRotationMatrix(rotationMatrix, null, gravityData, magneticData);

    //World coordinate system transformation for acceleration
    accelerometerWorldData[0] = rotationMatrix[0] * accelerometerData[0] + rotationMatrix[1] *
accelerometerData[1] + rotationMatrix[2] * accelerometerData[2];
    accelerometerWorldData[1] = rotationMatrix[3] * accelerometerData[0] + rotationMatrix[4] *
accelerometerData[1] + rotationMatrix[5] * accelerometerData[2];
    accelerometerWorldData[2] = rotationMatrix[6] * accelerometerData[0] + rotationMatrix[7] *
accelerometerData[1] + rotationMatrix[8] * accelerometerData[2];

}
```

Chapter 117: ProgressBar

Section 117.1: Material Linear ProgressBar

According to Material Documentation:

> A linear progress indicator should always fill from 0% to 100% and never decrease in value.
> It should be represented by bars on the edge of a header or sheet that appear and disappear.

To use a material Linear ProgressBar just use in your xml:

```
<ProgressBar
   android:id="@+id/my_progressBar"
   style="@style/Widget.AppCompat.ProgressBar.Horizontal"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
```

Indeterminate

To create **indeterminate** ProgressBar set the `android:indeterminate` attribute to **`true`**.

```
<ProgressBar
```

```
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="true"/>
```

Determinate

To create **determinate** ProgressBar set the `android:indeterminate` attribute to **`false`** and use the `android:max` and the `android:progress` attributes:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

Just use this code to update the value:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

Buffer

To create a **buffer** effect with the ProgressBar set the `android:indeterminate` attribute to **`false`** and use the `android:max`, the `android:progress` and the `android:secondaryProgress` attributes:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"
    android:secondaryProgress="25"/>
```

The buffer value is defined by `android:secondaryProgress` attribute.
Just use this code to update the values:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

Indeterminate and Determinate

To obtain this kind of ProgressBar just use an indeterminate ProgressBar using the `android:indeterminate` attribute to true.

```
<ProgressBar
    android:id="@+id/progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="true"/>
```

Then when you need to switch from indeterminate to determinate progress use `setIndeterminate()` method .

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

Section 117.2: Tinting ProgressBar

Using an AppCompat theme, the `ProgressBar`'s color will be the `colorAccent` you have defined.

`Version ≥ 5.0`

To change the `ProgressBar` color without changing the accent color you can use the`android:theme` attribute overriding the accent color:

```
<ProgressBar
   android:theme="@style/MyProgress"
   style="@style/Widget.AppCompat.ProgressBar" />

<!-- res/values/styles.xml -->
<style name="MyProgress" parent="Theme.AppCompat.Light">
    <item name="colorAccent">@color/myColor</item>
</style>
```

To tint the `ProgressBar` you can use in the xml file the attributes `android:indeterminateTintMode` and `android:indeterminateTint`

```
<ProgressBar
   android:indeterminateTintMode="src_in"
   android:indeterminateTint="@color/my_color"
/>
```

Section 117.3: Customized progressbar

CustomProgressBarActivity.java:

```
public class CustomProgressBarActivity extends AppCompatActivity {

    private TextView txtProgress;
    private ProgressBar progressBar;
    private int pStatus = 0;
    private Handler handler = new Handler();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom_progressbar);

        txtProgress = (TextView) findViewById(R.id.txtProgress);
        progressBar = (ProgressBar) findViewById(R.id.progressBar);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (pStatus <= 100) {
                    handler.post(new Runnable() {
                        @Override
                        public void run() {
                            progressBar.setProgress(pStatus);
                            txtProgress.setText(pStatus + " %");
                        }
                    });
                    try {
                        Thread.sleep(100);
                    } catch (InterruptedException e) {
```

```
                        e.printStackTrace();
                    }
                    pStatus++;
                }
            }
        }).start();

    }
}
```

activity_custom_progressbar.xml:

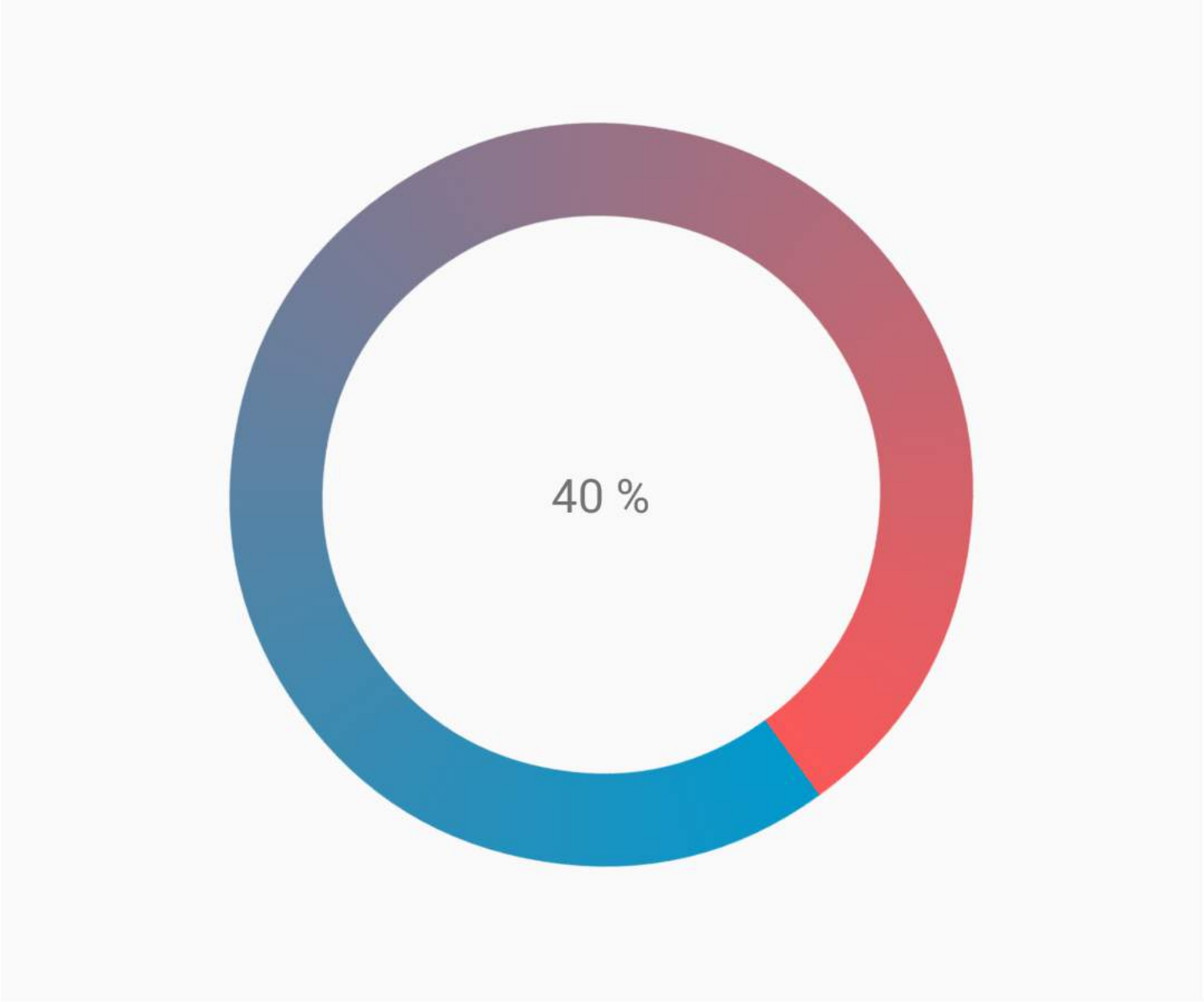
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.skholingua.android.custom_progressbar_circular.MainActivity" >


    <RelativeLayout
       android:layout_width="wrap_content"
       android:layout_centerInParent="true"
       android:layout_height="wrap_content">

        <ProgressBar
           android:id="@+id/progressBar"
           style="?android:attr/progressBarStyleHorizontal"
           android:layout_width="250dp"
           android:layout_height="250dp"
           android:layout_centerInParent="true"
           android:indeterminate="false"
           android:max="100"
           android:progress="0"
           android:progressDrawable="@drawable/custom_progressbar_drawable"
           android:secondaryProgress="0" />


        <TextView
           android:id="@+id/txtProgress"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_alignBottom="@+id/progressBar"
           android:layout_centerInParent="true"
           android:textAppearance="?android:attr/textAppearanceSmall" />
    </RelativeLayout>



</RelativeLayout>
```

custom_progressbar_drawable.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="-90"
    android:pivotX="50%"
    android:pivotY="50%"
```

```
    android:toDegrees="270" >

    <shape
        android:shape="ring"
        android:useLevel="false" >
        <gradient
            android:centerY="0.5"
            android:endColor="#FA5858"
            android:startColor="#0099CC"
            android:type="sweep"
            android:useLevel="false" />
    </shape>

</rotate>
```

Reference screenshot:



Section 117.4: Creating Custom Progress Dialog

By Creating Custom Progress Dialog class, the dialog can be used to show in UI instance, without recreating the dialog.

First Create a Custom Progress Dialog Class.

> CustomProgress.java

```
public class CustomProgress {

   public static CustomProgress customProgress = null;
   private Dialog mDialog;

   public static CustomProgress getInstance() {
       if (customProgress == null) {
           customProgress = new CustomProgress();
       }
       return customProgress;
   }

   public void showProgress(Context context, String message, boolean cancelable) {
       mDialog = new Dialog(context);
    // no tile for the dialog
       mDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
       mDialog.setContentView(R.layout.prograss_bar_dialog);
       mProgressBar = (ProgressBar) mDialog.findViewById(R.id.progress_bar);
    //  mProgressBar.getIndeterminateDrawable().setColorFilter(context.getResources()
    // .getColor(R.color.material_blue_gray_500), PorterDuff.Mode.SRC_IN);
       TextView progressText = (TextView) mDialog.findViewById(R.id.progress_text);
       progressText.setText("" + message);
       progressText.setVisibility(View.VISIBLE);
       mProgressBar.setVisibility(View.VISIBLE);
    // you can change or add this line according to your need
       mProgressBar.setIndeterminate(true);
       mDialog.setCancelable(cancelable);
       mDialog.setCanceledOnTouchOutside(cancelable);
       mDialog.show();
   }

   public void hideProgress() {
       if (mDialog != null) {
           mDialog.dismiss();
           mDialog = null;
       }
   }
}
```

Now creating the custom progress layout

> prograss_bar_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="65dp"
   android:background="@android:color/background_dark"
   android:orientation="vertical">

    <TextView
       android:id="@+id/progress_text"
       android:layout_width="wrap_content"
       android:layout_height="40dp"
       android:layout_above="@+id/progress_bar"
       android:layout_marginLeft="10dp"
```

```
        android:layout_marginStart="10dp"
        android:background="@android:color/transparent"
        android:gravity="center_vertical"
        android:text=""
        android:textColor="@android:color/white"
        android:textSize="16sp"
        android:visibility="gone" />

    <-- Where the style can be changed to any kind of ProgressBar -->

    <ProgressBar
        android:id="@+id/progress_bar"
        style="@android:style/Widget.DeviceDefault.ProgressBar.Horizontal"
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_gravity="center"
        android:background="@color/cardview_dark_background"
        android:maxHeight="20dp"
        android:minHeight="20dp" />

</RelativeLayout>
```

This is it. Now for calling the Dialog in Code

```
CustomProgress customProgress = CustomProgress.getInstance();

// now you have the instance of CustomProgres
// for showing the ProgressBar

customProgress.showProgress(#Context, getString(#StringId), #boolean);

// for hiding the ProgressBar

customProgress.hideProgress();
```

Section 117.5: Indeterminate ProgressBar

An indeterminate ProgressBar shows a cyclic animation without an indication of progress.

Basic indeterminate ProgressBar (spinning wheel)

```
<ProgressBar
   android:id="@+id/progressBar"
   android:indeterminate="true"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
```

Horizontal indeterminate ProgressBar (flat bar)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Other built-in ProgressBar styles

```
style="@android:style/Widget.ProgressBar.Small"
style="@android:style/Widget.ProgressBar.Large"
style="@android:style/Widget.ProgressBar.Inverse"
style="@android:style/Widget.ProgressBar.Small.Inverse"
style="@android:style/Widget.ProgressBar.Large.Inverse"
```

To use the indeterminate ProgressBar in an Activity

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

Section 117.6: Determinate ProgressBar

A determinate ProgressBar shows the current progress towards a specific maximum value.

Horizontal determinate ProgressBar

```
<ProgressBar
   android:id="@+id/progressBar"
   android:indeterminate="false"
   android:layout_width="match_parent"
   android:layout_height="10dp"
   style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Vertical determinate ProgressBar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="10dp"
    android:layout_height="match_parent"
    android:progressDrawable="@drawable/progress_vertical"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res/drawable/progress_vertical.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/background">
        <shape>
            <corners android:radius="3dp"/>
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_light"/>
            </shape>
        </clip>
    </item>
    <item android:id="@android:id/progress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
```

```
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_dark"/>
            </shape>
        </clip>
    </item>
</layer-list>
```

Ring determinate ProgressBar

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:progressDrawable="@drawable/progress_ring"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res/drawable/progress_ring.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/secondaryProgress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#0000FF"/>
        </shape>
    </item>

    <item android:id="@android:id/progress">
        <shape
            android:shape="ring"
            android:useLevel="true"
            android:thicknessRatio="24"
            android:innerRadiusRatio="2.2">
            <corners android:radius="3dp"/>
            <solid android:color="#FFFFFF"/>
        </shape>
    </item>
</layer-list>
```

To use the determinate ProgressBar in an Activity.

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);
```

Chapter 118: Custom Fonts

Section 118.1: Custom font in canvas text

Drawing text in canvas with your font from assets.

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

Section 118.2: Working with fonts in Android O

Android O changes the way to work with fonts.

Android O introduces a new feature, called *Fonts in XML*, which allows you to use fonts as resources. This means, that there is no need to bundle fonts as assets. Fonts are now compiled in an *R* file and are automatically available in the system as a resource.

In order to add a new **font**, you have to do the following:

- Create a new resource directory: `res/font`.
- Add your font files into this font folder. For example, by adding `myfont.ttf`, you will be able to use this font via `R.font.myfont`.

You can also create your own **font family** by adding the following XML file into the `res/font` directory:

```
<?xml version="1.0" encoding="utf-8"?>
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
    <font
       android:fontStyle="normal"
       android:fontWeight="400"
       android:font="@font/lobster_regular" />
    <font
       android:fontStyle="italic"
       android:fontWeight="400"
       android:font="@font/lobster_italic" />
</font-family>
```

You can use both the **font** file and the **font family** file in the same way:

- **In an XML file,** by using the `android:fontFamily` attribute, for example like this:

  ```
  <TextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:fontFamily="@font/myfont"/>
  ```

  Or like this:

  ```
  <style name="customfontstyle" parent="@android:style/TextAppearance.Small">
      <item name="android:fontFamily">@font/myfont</item>
  </style>
  ```

- **In your code**, by using the following lines of code:

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

Section 118.3: Custom font to whole activity

```
public class ReplaceFont {

public static void changeDefaultFont(Context context, String oldFont, String assetsFont) {
    Typeface typeface = Typeface.createFromAsset(context.getAssets(), assetsFont);
    replaceFont(oldFont, typeface);
}

private static void replaceFont(String oldFont, Typeface typeface) {
    try {
        Field myField = Typeface.class.getDeclaredField(oldFont);
        myField.setAccessible(true);
        myField.set(null, typeface);
    } catch (NoSuchFieldException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }
}
```

Then in your activity, in `onCreate()` method:

```
// Put your font to assets folder...

ReplaceFont.changeDefaultFont(getApplication(), "DEFAULT", "LinLibertine.ttf");
```

Section 118.4: Putting a custom font in your app

1. Go to the (project folder)
2. Then app -> src -> main.
3. Create folder 'assets -> fonts' into the main folder.
4. Put your 'fontfile.ttf' into the fonts folder.

Section 118.5: Initializing a font

```
private Typeface myFont;

// A good practice might be to call this in onCreate() of a custom
// Application class and pass 'this' as Context. Your font will be ready to use
// as long as your app lives
public void initFont(Context context) {
    myFont = Typeface.createFromAsset(context.getAssets(), "fonts/Roboto-Light.ttf");
}
```

Section 118.6: Using a custom font in a TextView

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

Section 118.7: Apply font on TextView by xml (Not required Java code)

TextViewPlus.java:

```
public class TextViewPlus extends TextView {
    private static final String TAG = "TextView";

    public TextViewPlus(Context context) {
        super(context);
    }

    public TextViewPlus(Context context, AttributeSet attrs) {
        super(context, attrs);
        setCustomFont(context, attrs);
    }

    public TextViewPlus(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setCustomFont(context, attrs);
    }

    private void setCustomFont(Context ctx, AttributeSet attrs) {
        TypedArray a = ctx.obtainStyledAttributes(attrs, R.styleable.TextViewPlus);
        String customFont = a.getString(R.styleable.TextViewPlus_customFont);
        setCustomFont(ctx, customFont);
        a.recycle();
    }

    public boolean setCustomFont(Context ctx, String asset) {
        Typeface typeface = null;
        try {
            typeface = Typeface.createFromAsset(ctx.getAssets(), asset);
        } catch (Exception e) {
            Log.e(TAG, "Unable to load typeface: "+e.getMessage());
            return false;
        }

        setTypeface(typeface);
        return true;
    }
}
```

attrs.xml: (Where to place **res/values**)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

How to use:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:foo="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
```

```
    <com.mypackage.TextViewPlus
        android:id="@+id/textViewPlus1"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:text="@string/showingOffTheNewTypeface"
        foo:customFont="my_font_name_regular.otf">
    </com.mypackage.TextViewPlus>
</LinearLayout>
```

Section 118.8: Efficient Typeface loading

Loading custom fonts can be lead to a bad performance. I highly recommend to use this little helper which saves/loads your already used fonts into a Hashtable.

```
public class TypefaceUtils {

private static final Hashtable<String, Typeface> sTypeFaces = new Hashtable<>();

/**
 * Get typeface by filename from assets main directory
 *
 * @param context
 * @param fileName the name of the font file in the asset main directory
 * @return
 */
public static Typeface getTypeFace(final Context context, final String fileName) {
    Typeface tempTypeface = sTypeFaces.get(fileName);

    if (tempTypeface == null) {
        tempTypeface = Typeface.createFromAsset(context.getAssets(), fileName);
        sTypeFaces.put(fileName, tempTypeface);
    }

    return tempTypeface;
}
```

}

Usage:

```
Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);
```

Chapter 119: Getting system font names and using the fonts

The following examples show how to retrieve the default names of the system fonts that are store in the */system/fonts/* directory and how to use a system font to set the typeface of a `TextView` element.

Section 119.1: Getting system font names

```
ArrayList<String> fontNames = new ArrayList<String>();
File temp = new File("/system/fonts/");
String fontSuffix = ".ttf";

for(File font : temp.listFiles()) {
    String fontName = font.getName();
    if(fontName.endsWith(fontSuffix)) {
        fontNames.add(fontName.subSequence(0,fontName.lastIndexOf(fontSuffix)).toString());
    }
}
```

Section 119.2: Applying a system font to a TextView

In the following code you need to replace `fontsname` by the name of the font you would like to use:

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

Chapter 120: Text to Speech(TTS)

Section 120.1: Text to Speech Base

layout_text_to_speech.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp">

    <EditText
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:hint="Enter text here!"
       android:id="@+id/textToSpeak"/>
    <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_centerHorizontal="true"
       android:layout_below="@id/textToSpeak"
       android:id="@+id/btnSpeak"/>

</RelativeLayout>
```

AndroidTextToSpeechActivity.java

```
public class AndroidTextToSpeechActivity extends Activity implements
        TextToSpeech.OnInitListener {

    EditText textToSpeak = null;
    Button btnSpeak = null;
    TextToSpeech tts;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        textToSpeak = findViewById(R.id.textToSpeak);
        btnSpeak = findViewById(R.id.btnSpeak);
        btnSpeak.setEnabled(false);
        tts = new TextToSpeech(this, this);
        btnSpeak.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    speakOut();
                }
            });
    }

    @Override
    public void onDestroy() {
        // Don't forget to shutdown tts!
        if (tts != null) {
            tts.stop();
            tts.shutdown();
        }
        super.onDestroy();
    }
```

```
    @Override
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            int result = tts.setLanguage(Locale.US);

            if (result == TextToSpeech.LANG_MISSING_DATA
                    || result == TextToSpeech.LANG_NOT_SUPPORTED) {
                Log.e("TTS", "This Language is not supported");
            } else {
                btnSpeak.setEnabled(true);
                speakOut();
            }
        } else {
            Log.e("TTS", "Initilization Failed!");
        }
    }

    private void speakOut() {
        String text = textToSpeak.getText().toString();
        if(text == null || text.isEmpty())
            return;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            String utteranceId=this.hashCode() + "";
            tts.speak(text, TextToSpeech.QUEUE_FLUSH, null, utteranceId);
        } else {
            tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
        }
    }
}
```

The language to be spoken can be set by providing a `Locale` to the `setLanguage()` method:

```
tts.setLanguage(Locale.CHINESE); // Chinese language
```

The number of supported languages varies between Android levels. The method `isLanguageAvailable()` can be used to check if a certain language is supported:

```
tts.isLanguageAvailable(Locale.CHINESE);
```

The speech pitch level can be set by using the `setPitch()` method. By default, the pitch value is 1.0. Use values less than 1.0 to decrease the pitch level or values greater than 1.0 to increase the pitch level:

```
tts.setPitch(0.6);
```

The speech rate can be set using `setSpeechRate()`. The default speech rate is 1.0. The speech rate can be doubled by setting it to 2.0 or made half by setting it to 0.5:

```
tts.setSpeechRate(2.0);
```

Section 120.2: TextToSpeech implementation across the APIs

Cold observable implementation, emits true when TTS engine finishes speaking, starts speaking when subscribed. Notice that API level 21 introduces different way to perform speaking:

```
public class RxTextToSpeech {
```

```
@Nullable RxTTSObservableOnSubscribe audio;

WeakReference<Context> contextRef;

public RxTextToSpeech(Context context) {
    this.contextRef = new WeakReference<>(context);
}

public void requestTTS(FragmentActivity activity, int requestCode) {
    Intent checkTTSIntent = new Intent();
    checkTTSIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
    activity.startActivityForResult(checkTTSIntent, requestCode);
}

public void cancelCurrent() {
    if (audio != null) {
        audio.dispose();
        audio = null;
    }
}

public Observable<Boolean> speak(String textToRead) {
    audio = new RxTTSObservableOnSubscribe(contextRef.get(), textToRead, Locale.GERMANY);
    return Observable.create(audio);
}


public static class RxTTSObservableOnSubscribe extends UtteranceProgressListener
        implements ObservableOnSubscribe<Boolean>,
        Disposable, Cancellable, TextToSpeech.OnInitListener {

    volatile boolean disposed;
    ObservableEmitter<Boolean> emitter;
    TextToSpeech textToSpeech;
    String text = "";
    Locale selectedLocale;
    Context context;

    public RxTTSObservableOnSubscribe(Context context, String text, Locale locale) {
        this.selectedLocale = locale;
        this.context = context;
        this.text = text;
    }

    @Override public void subscribe(ObservableEmitter<Boolean> e) throws Exception {
        this.emitter = e;
        if (context == null) {
            this.emitter.onError(new Throwable("nullable context, cannot execute " + text));
        } else {
            this.textToSpeech = new TextToSpeech(context, this);
        }
    }

    @Override @DebugLog public void dispose() {
        if (textToSpeech != null) {
            textToSpeech.setOnUtteranceProgressListener(null);
            textToSpeech.stop();
            textToSpeech.shutdown();
            textToSpeech = null;
        }
        disposed = true;
    }
```

```
    @Override public boolean isDisposed() {
        return disposed;
    }

    @Override public void cancel() throws Exception {
        dispose();
    }

    @Override public void onInit(int status) {

        int languageCode = textToSpeech.setLanguage(selectedLocale);

        if (languageCode == android.speech.tts.TextToSpeech.LANG_COUNTRY_AVAILABLE) {
            textToSpeech.setPitch(1);
            textToSpeech.setSpeechRate(1.0f);
            textToSpeech.setOnUtteranceProgressListener(this);
            performSpeak();
        } else {
            emitter.onError(new Throwable("language " + selectedLocale.getCountry() + " is not
supported"));
        }
    }

    @Override public void onStart(String utteranceId) {
        //no-op
    }

    @Override public void onDone(String utteranceId) {
        this.emitter.onNext(true);
        this.emitter.onComplete();
    }

    @Override public void onError(String utteranceId) {
        this.emitter.onError(new Throwable("error TTS " + utteranceId));
    }

    void performSpeak() {

        if (isAtLeastApiLevel(21)) {
            speakWithNewApi();
        } else {
            speakWithOldApi();
        }
    }

    @RequiresApi(api = 21) void speakWithNewApi() {
        Bundle params = new Bundle();
        params.putString(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, "");
        textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, params, uniqueId());
    }

    void speakWithOldApi() {
        HashMap<String, String> map = new HashMap<>();
        map.put(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, uniqueId());
        textToSpeech.speak(text, TextToSpeech.QUEUE_ADD, map);
    }

    private String uniqueId() {
        return UUID.randomUUID().toString();
    }
}
```

```
public static boolean isAtLeastApiLevel(int apiLevel) {
    return Build.VERSION.SDK_INT >= apiLevel;
}
```

}

Chapter 121: Spinner

Section 121.1: Basic Spinner Example

Spinner It is a type of dropdown input. Firstly in layout

```
  <Spinner
   android:id="@+id/spinner"     <!-- id to refer this spinner from JAVA-->
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</Spinner>
```

Now Secondly populate values in spinner There are mainly two ways to populate values in `spinner`.

1. From XML itself create a **array.xml** in **values** directory under **res**. Create this `array`

```
  <string-array name="defaultValue">
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
</string-array>
```

Now add this line in sppiner XML

```
            android:entries="@array/defaultValue"
```

2. You can also add values via JAVA

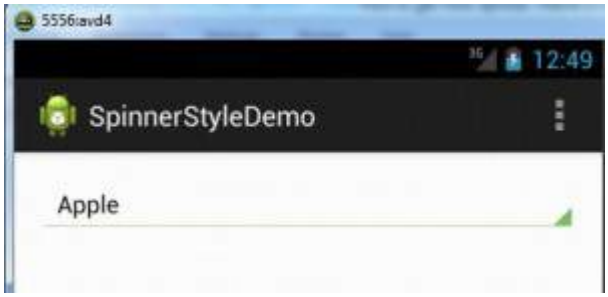
if you are using in `activity` cityArea = (Spinner) findViewById(R.id.cityArea); else if you are using in `fragment`

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

Now create a `arrayList` of `Strings`

```
ArrayList<String> area = new ArrayList<>();
//add values in area arrayList
cityArea.setAdapter(new ArrayAdapter<String>(context
                           , android.R.layout.simple_list_item_1, area));
```

This will look like



According to the device Android version it will render style

Following are some of the default themes

If an app does not explicitly request a theme in its manifest, Android System will determine the default theme based on the app's targetSdkVersion to maintain the app's original expectations:

Android SDK Version	Default Theme
Version < 11	@android:style/Theme
Version between 11 and 13	@android:style/Theme.Holo
14 and higher	@android:style/Theme.DeviceDefault

`Spinner` can be easily customized with the help of xml eg

```
android:background="@drawable/spinner_background"

android:layout_margin="16dp"

android:padding="16dp"
```

Create a custom background in XML and use it.

easily get the position and other details of the selected item in spinner

```
cityArea.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            areaNo = position;
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    });
```

Change the text color of the selected item in spinner

This can be done in two ways in XML

```
<item android:state_activated="true" android:color="@color/red"/>
```

This will change the selected item color in the popup.

and from JAVA do this (in the setOnItemSelectedListener(...))

```
 @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                 ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);
// similarly change `background color` etc.
            }
```

Section 121.2: Adding a spinner to your activity

In /res/values/strings.xml:

```
<string-array name="spinner_options">
    <item>Option 1</item>
```

```
    <item>Option 2</item>
    <item>Option 3</item>
</string-array>
```

In layout XML:

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

In Activity:

```
Spinner spinnerName = (Spinner) findViewById(R.id.spinnerName);
spinnerName.setOnItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
         String chosenOption = (String) parent.getItemAtPosition(position);
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
});
```

Chapter 122: Data Encryption/Decryption

This topic discusses how encryption and decryption works in Android.

Section 122.1: AES encryption of data using password in a secure way

The following example encrypts a given data block using AES. The encryption key is derived in a secure way (random salt, 1000 rounds of SHA-256). The encryption uses AES in CBC mode with random IV.

Note that the data stored in the class `EncryptedData` (`salt`, `iv`, and `encryptedData`) can be concatenated to a single byte array. You can then save the data or transmit it to the recipient.

```
private static final int SALT_BYTES = 8;
private static final int PBK_ITERATIONS = 1000;
private static final String ENCRYPTION_ALGORITHM = "AES/CBC/PKCS5Padding";
private static final String PBE_ALGORITHM = "PBEwithSHA256and128BITAES-CBC-BC";

private EncryptedData encrypt(String password, byte[] data) throws NoSuchPaddingException,
NoSuchAlgorithmException, InvalidKeySpecException, InvalidKeyException, BadPaddingException,
IllegalBlockSizeException, InvalidAlgorithmParameterException {
    EncryptedData encData = new EncryptedData();
    SecureRandom rnd = new SecureRandom();
    encData.salt = new byte[SALT_BYTES];
    encData.iv = new byte[16]; // AES block size
    rnd.nextBytes(encData.salt);
    rnd.nextBytes(encData.iv);

    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), encData.salt, PBK_ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance(ENCRYPTION_ALGORITHM);
    IvParameterSpec ivSpec = new IvParameterSpec(encData.iv);
    cipher.init(Cipher.ENCRYPT_MODE, key, ivSpec);
    encData.encryptedData = cipher.doFinal(data);
    return encData;
}

private byte[] decrypt(String password, byte[] salt, byte[] iv, byte[] encryptedData) throws
NoSuchAlgorithmException, InvalidKeySpecException, NoSuchPaddingException, InvalidKeyException,
BadPaddingException, IllegalBlockSizeException, InvalidAlgorithmParameterException {
    PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt, PBK_ITERATIONS);
    SecretKeyFactory secretKeyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
    Key key = secretKeyFactory.generateSecret(keySpec);
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, key, ivSpec);
    return cipher.doFinal(encryptedData);
}

private static class EncryptedData {
    public byte[] salt;
    public byte[] iv;
    public byte[] encryptedData;
}
```

The following example code shows how to test encryption and decryption:

```
try {
    String password = "test12345";
    byte[] data = "plaintext11223344556677889900".getBytes("UTF-8");
    EncryptedData encData = encrypt(password, data);
    byte[] decryptedData = decrypt(password, encData.salt, encData.iv, encData.encryptedData);
    String decDataAsString = new String(decryptedData, "UTF-8");
    Toast.makeText(this, decDataAsString, Toast.LENGTH_LONG).show();
} catch (Exception e) {
    e.printStackTrace();
}
```

Chapter 123: OkHttp

Section 123.1: Basic usage example

I like to wrap my `OkHttp` into a class called `HttpClient` for example, and in this class I have methods for each of the major HTTP verbs, `post`, `get`, `put` and `delete`, most commonly. (I usually include an interface, in order to keep for it to implement, in order to be able to easily change to a different implementation, if need be):

```
public class HttpClient implements HttpClientInterface{

private static final String TAG = OkHttpClient.class.getSimpleName();
public static final MediaType JSON
        = MediaType.parse("application/json; charset=utf-8");

OkHttpClient httpClient = new OkHttpClient();

@Override
public String post(String url, String json) throws IOException {
    Log.i(TAG, "Sending a post request with body:\n" + json + "\n to URL: " + url);

    RequestBody body = RequestBody.create(JSON, json);
    Request request = new Request.Builder()
            .url(url)
            .post(body)
            .build();
    Response response = httpClient.newCall(request).execute();
    return response.body().string();
}
```

The syntax is the same for `put`, `get` and `delete` except for 1 word (`.put(body)`) so it might be obnoxious to post that code as well. Usage is pretty simple, just call the appropriate method on some `url` with some `json` payload and the method will return a string as a result that you can later use and parse. Let's assume that the response will be a `json`, we can create a `JSONObject` easily from it:

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);
JSONObject json = new JSONObject(response);
// continue to parse the response according to it's structure
```

Section 123.2: Setting up OkHttp

Grab via Maven:

```
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>3.6.0</version>
</dependency>
```

or Gradle:

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

Section 123.3: Logging interceptor

`Interceptors` are used to intercept `OkHttp` calls. The reason to intercept could be to monitor, rewrite and retry

calls. It can be used for outgoing request or incoming response both.

```
class LoggingInterceptor implements Interceptor {
  @Override public Response intercept(Interceptor.Chain chain) throws IOException {
    Request request = chain.request();

    long t1 = System.nanoTime();
    logger.info(String.format("Sending request %s on %s%n%s",
        request.url(), chain.connection(), request.headers()));

    Response response = chain.proceed(request);

    long t2 = System.nanoTime();
    logger.info(String.format("Received response for %s in %.1fms%n%s",
        response.request().url(), (t2 - t1) / 1e6d, response.headers()));

    return response;
  }
}
```

Section 123.4: Synchronous Get Call

```
private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

    Headers responseHeaders = response.headers();

    System.out.println(response.body().string());
  }
```

Section 123.5: Asynchronous Get Call

```
private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    client.newCall(request).enqueue(new Callback() {
      @Override public void onFailure(Call call, IOException e) {
        e.printStackTrace();
      }

      @Override
      public void onResponse(Call call, Response response) throws IOException {
        if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

        Headers responseHeaders = response.headers();

        System.out.println(response.body().string());
      }
```

```
    });
  }
```

Section 123.6: Posting form parameters

```
private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    RequestBody formBody = new FormBody.Builder()
        .add("search", "Jurassic Park")
        .build();
    Request request = new Request.Builder()
        .url("https://en.wikipedia.org/w/index.php")
        .post(formBody)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

    System.out.println(response.body().string());
  }
```

Section 123.7: Posting a multipart request

```
 private static final String IMGUR_CLIENT_ID = "...";
  private static final MediaType MEDIA_TYPE_PNG = MediaType.parse("image/png");

  private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    // Use the imgur image upload API as documented at https://api.imgur.com/endpoints/image
    RequestBody requestBody = new MultipartBody.Builder()
        .setType(MultipartBody.FORM)
        .addFormDataPart("title", "Square Logo")
        .addFormDataPart("image", "logo-square.png",
            RequestBody.create(MEDIA_TYPE_PNG, new File("website/static/logo-square.png")))
        .build();

    Request request = new Request.Builder()
        .header("Authorization", "Client-ID " + IMGUR_CLIENT_ID)
        .url("https://api.imgur.com/3/image")
        .post(requestBody)
        .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

    System.out.println(response.body().string());
  }
```

Section 123.8: Rewriting Responses

```
private static final Interceptor REWRITE_CACHE_CONTROL_INTERCEPTOR = new Interceptor() {
  @Override public Response intercept(Interceptor.Chain chain) throws IOException {
    Response originalResponse = chain.proceed(chain.request());
    return originalResponse.newBuilder()
        .header("Cache-Control", "max-age=60")
        .build();
  }
```

```
};
```

Chapter 124: Handling Deep Links

`<data>` Attribute	Details
scheme	The *scheme* part of a URI (case-sensitive). Examples: `http`, `https`, `ftp`
host	The *host* part of a URI (case-sensitive). Examples: `google.com`, `example.org`
port	The *port* part of a URI. Examples: `80`, `443`
path	The *path* part of a URI. Must begin with `/`. Examples: `/`, `/about`
pathPrefix	A prefix for the *path* part of a URI. Examples: `/item`, `/article`
pathPattern	A pattern to match for the *path* part of a URI. Examples: `/item/.*`, `/article/[0-9]*`
mimeType	A mime type to match. Examples: `image/jpeg`, `audio/*`

Deep links are URLs that take users directly to specific content in your app. You can set up deep links by adding intent filters and extracting data from incoming intents to drive users to the right screen in your app.

Section 124.1: Retrieving query parameters

```
public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Intent intent = getIntent();
        Uri data = intent.getData();

        if (data != null) {
            String param1 = data.getQueryParameter("param1");
            String param2 = data.getQueryParameter("param2");
        }
    }

}
```

If the user clicks on a linkto `http://www.example.com/map?param1=FOO&param2=BAR`, then `param1` here will have a value of `"FOO"` and `param2` will have a value of `"BAR"`.

Section 124.2: Simple deep link

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

    </intent-filter>

</activity>
```

This will accept any link starting with `http://www.example.com` as a deep link to start your `MainActivity`.

Section 124.3: Multiple paths on a single domain

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:path="/" />
        <data android:path="/about" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

This will launch your `MainActivity` when the user clicks any of these links:

- `http://www.example.com/`
- `http://www.example.com/about`
- `http://www.example.com/map`

Section 124.4: Multiple domains and multiple paths

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

        <data android:scheme="http"
              android:host="www.example2.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

This will launch your MainActivity when the user clicks any of these links:

- `http://www.example.com/`
- `http://www.example2.com/`

- `http://www.example.com/map`
- `http://www.example2.com/map`

Section 124.5: Both http and https for the same domain

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http" />
        <data android:scheme="https" />

        <data android:host="www.example.com" />

        <data android:path="/" />
        <data android:path="/map" />

    </intent-filter>

</activity>
```

This will launch your MainActivity when the user clicks any of these links:

- `http://www.example.com/`
- `https://www.example.com/`
- `http://www.example.com/map`
- `https://www.example.com/map`

Section 124.6: Using pathPrefix

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com"
              android:path="/item" />

    </intent-filter>

</activity>
```

This will launch your MainActivity when the user clicks any link starting with `http://www.example.com/item`, such as:

- `https://www.example.com/item`
- `http://www.example.com/item/1234`
- `https://www.example.com/item/xyz/details`

Chapter 125: Crash Reporting Tools

Section 125.1: Fabric - Crashlytics

Fabric is a modular mobile platform that provides useful kits you can mix to build your application. **Crashlytics** is a crash and issue reporting tool provided by Fabric that allows you to track and monitor your applications in detail.

How to Configure Fabric-Crashlytics

Step 1: Change your `build.gradle`:

Add the plugin repo and the gradle plugin:

```
buildscript {
  repositories {
    maven { url 'https://maven.fabric.io/public' }
  }

  dependencies {
    // The Fabric Gradle plugin uses an open ended version to react
    // quickly to Android tooling updates
    classpath 'io.fabric.tools:gradle:1.+'
  }
}
```

Apply the plugin:

```
apply plugin: 'com.android.application'
//Put Fabric plugin after Android plugin
apply plugin: 'io.fabric'
```

Add the Fabric repo:

```
repositories {
  maven { url 'https://maven.fabric.io/public' }
}
```

Add the Crashlyrics Kit:

```
dependencies {

  compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {
    transitive = true;
  }
}
```

Step 2: Add Your **API Key** and the **INTERNET** permission in `AndroidManifest.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
  <application
    ... >

      <meta-data
         android:name="io.fabric.ApiKey"
```

```
        android:value="25eeca3bb31cd41577e097cabd1ab9eee9da151d"
        />

  </application>

  <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

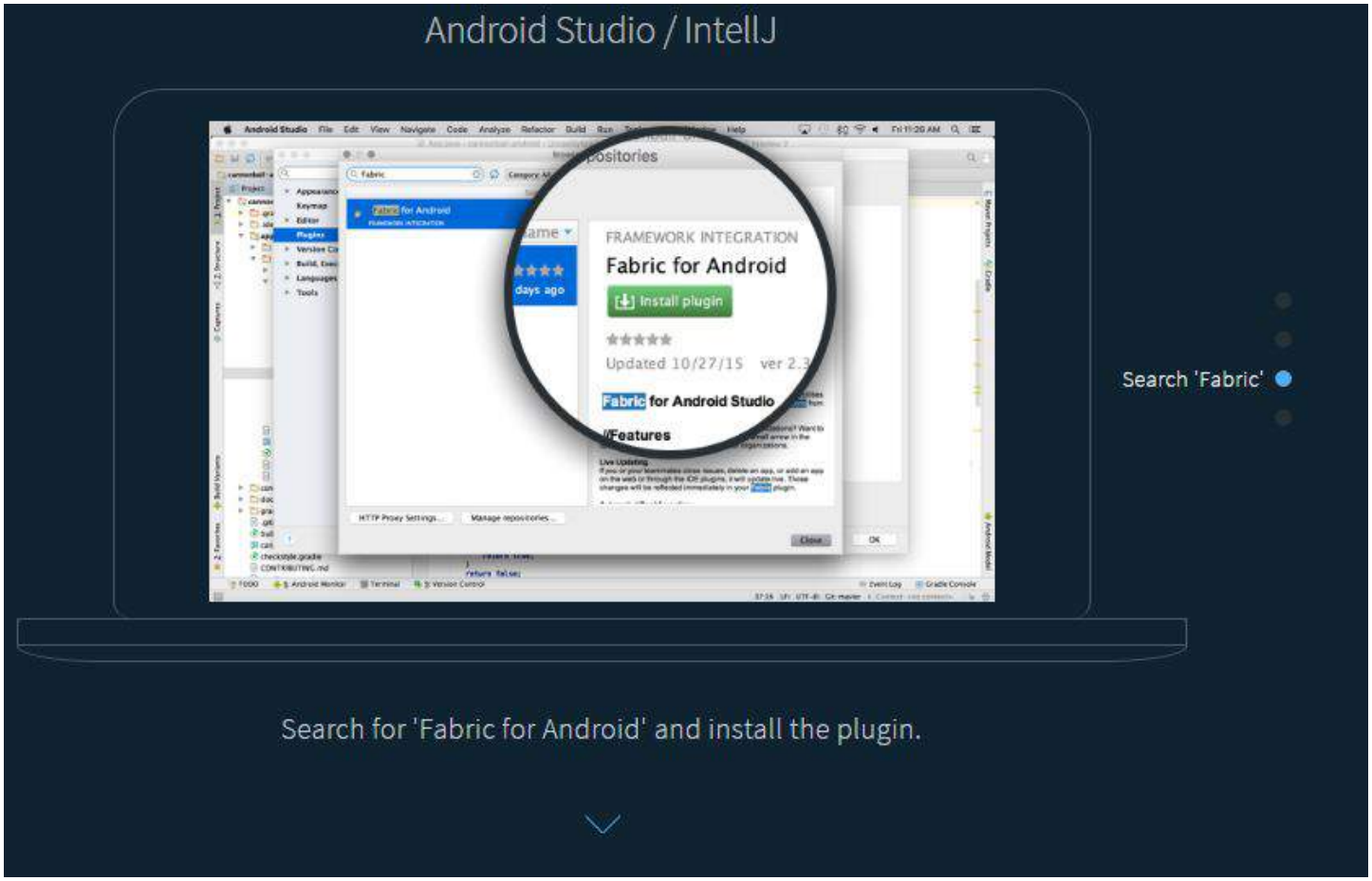
Step 3: Init the Kit at runtime in you code, for example:

```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      //Init the KIT
      Fabric.with(this, new Crashlytics());

      setContentView(R.layout.activity_main);
    }
}
```

Step 4: Build project. To build and run:

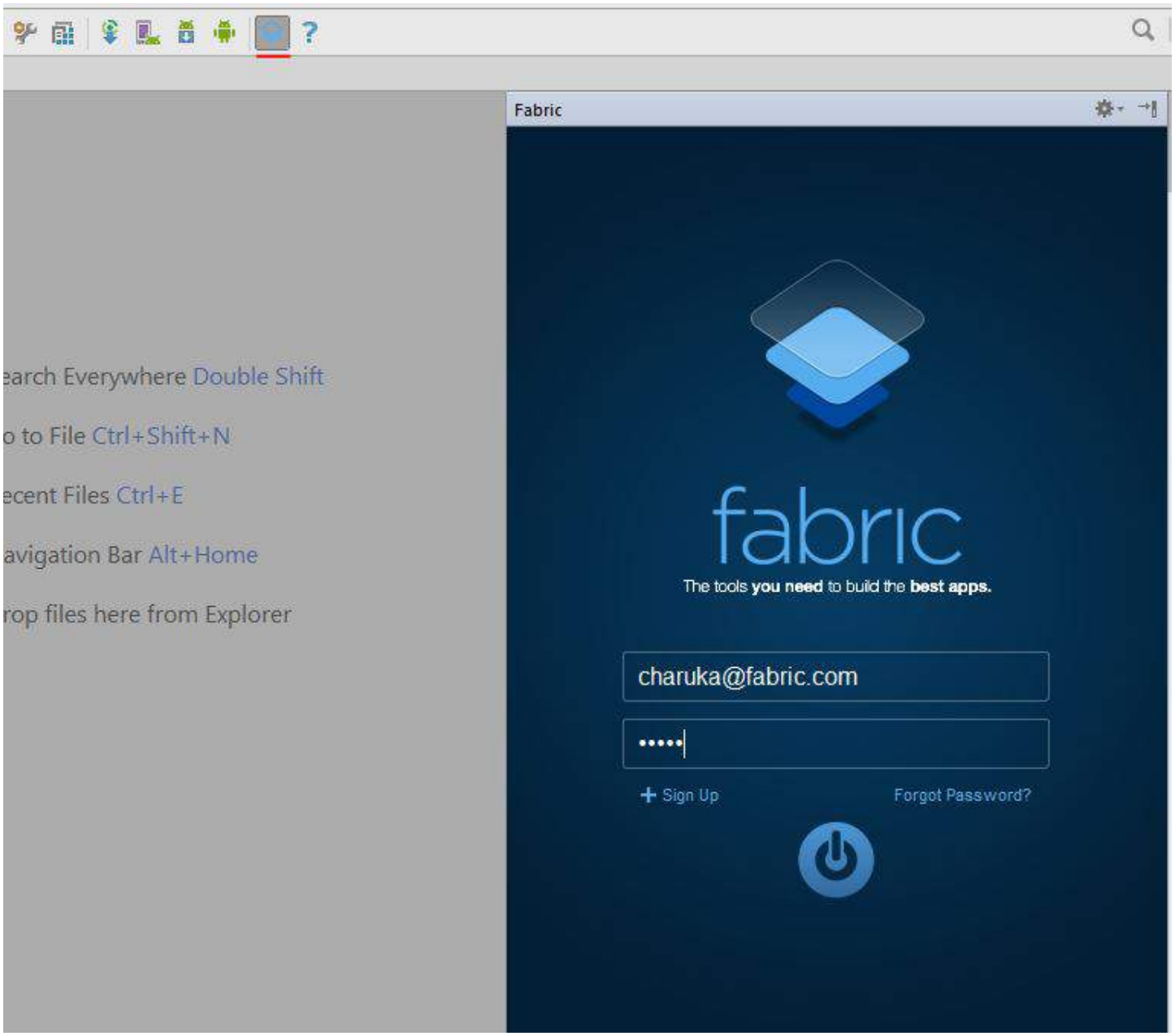
Using the Fabric IDE plugin

Kits can be installed using the Fabric IDE plugin for Android Studio or IntelliJ following this link.



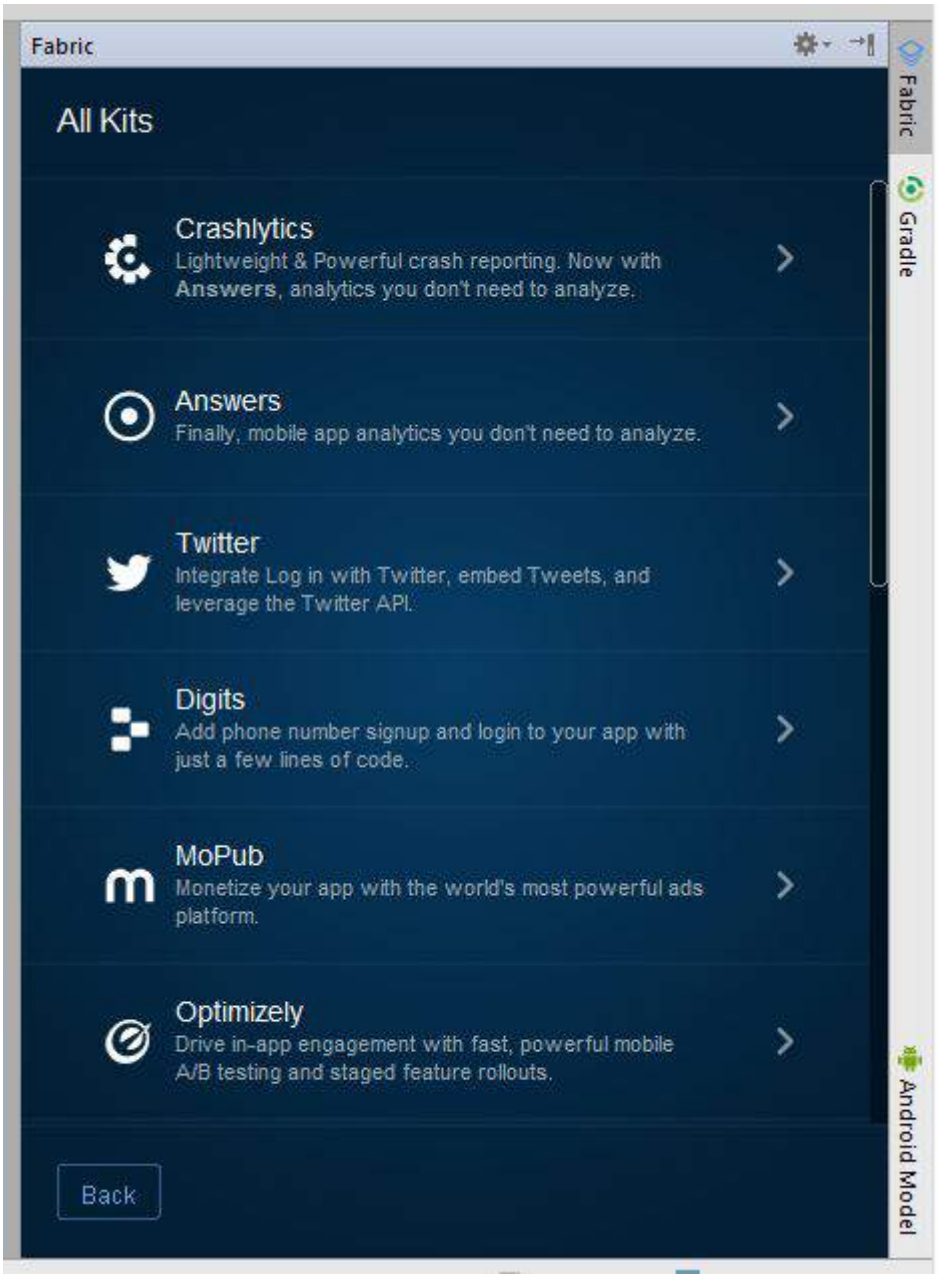
After installing the plugin, **restart** Android Studio and **login** with your account using **Android Studio**.

(short key > `CTRL + L`)

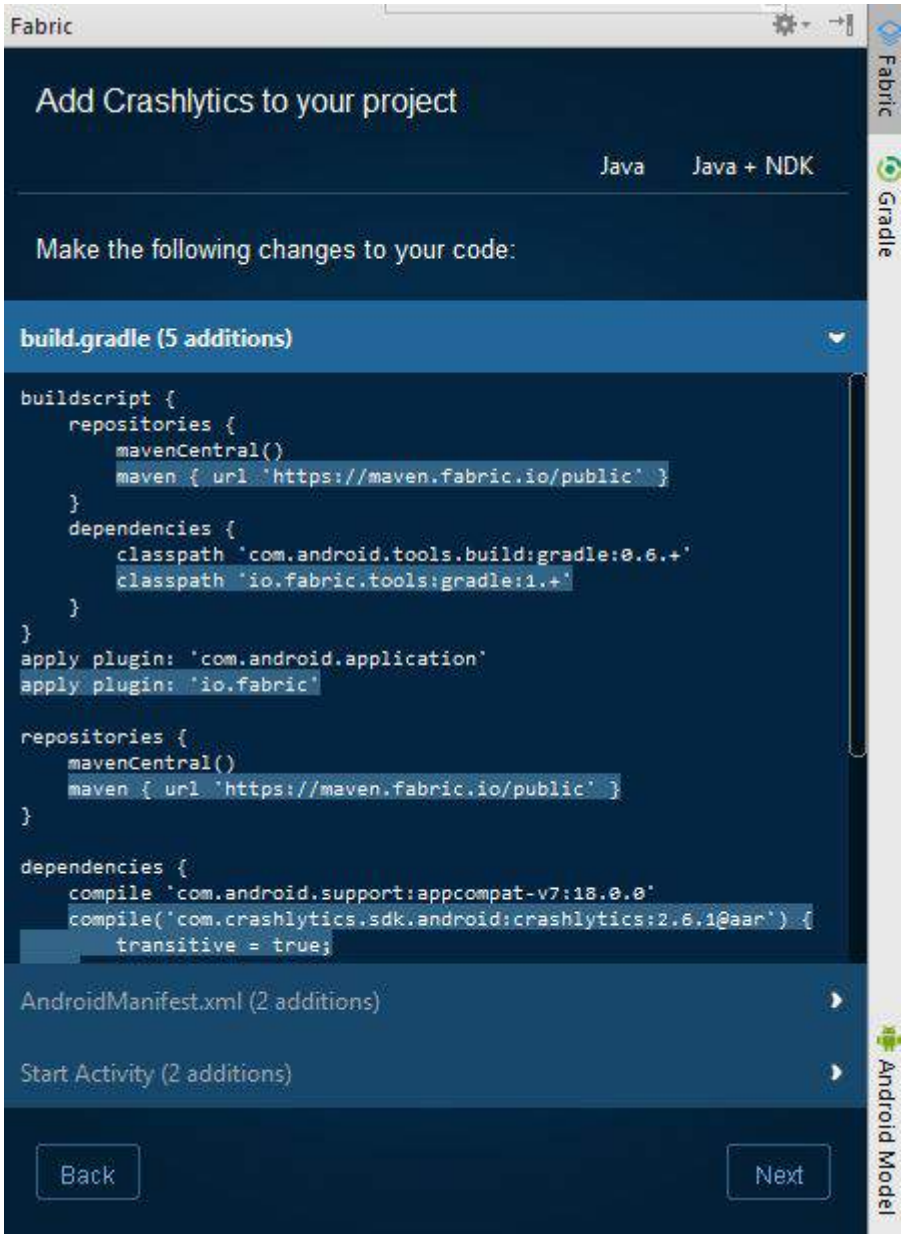


Then it will show the projects that you have / the project you opened, select the one you need and click next .. next.

Select the kit you would like to add, for his example it is **Crashlytics** :



Then hit `Install`. You don't need to add it manually this time like above **gradle plugin**, instead it will build for you.



Done!

Section 125.2: Capture crashes using Sherlock

Sherlock captures all your crashes and reports them as a notification. When you tap on the notification, it opens up an activity with all the crash details along with Device and Application info

How to integrate Sherlock with your application?

You just need to add Sherlock as a gradle dependency in your project.

```
dependencies {
    compile('com.github.ajitsing:sherlock:1.0.1@aar') {
        transitive = true
    }
}
```

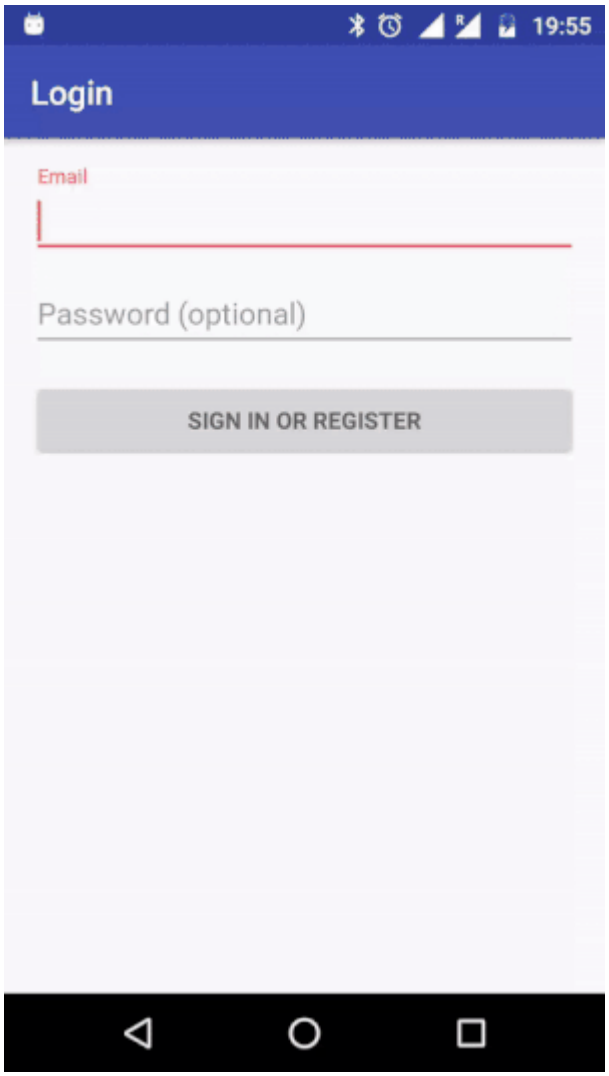
After syncing your android studio, initialize Sherlock in your Application class.

```
package com.singhajit.login;
```

```
import android.app.Application;

import com.singhajit.sherlock.core.Sherlock;

public class SampleApp extends Application {
  @Override
  public void onCreate() {
    super.onCreate();
    Sherlock.init(this);
  }
}
```

Thats all you need to do. Also Sherlock does much more than just reporting a crash. To checkout all its features take a look at this article.

Demo



Section 125.3: Force a Test Crash With Fabric

Add a button you can tap to trigger a crash. Paste this code into your layout where you'd like the button to appear.

```
<Button
   android:layout_height="wrap_content"
   android:layout_width="wrap_content"
   android:text="Force Crash!"
   android:onClick="forceCrash"
   android:layout_centerVertical="true"
```

```
    android:layout_centerHorizontal="true" />
```

Throw a RuntimeException

```
public void forceCrash(View view) {
    throw new RuntimeException("This is a crash");
}
```

Run your app and tap the new button to cause a crash. In a minute or two you should be able to see the crash on your Crashlytics dashboard as well as you will get a mail.

Section 125.4: Crash Reporting with ACRA

Step 1: Add the dependency of latest ACRA AAR to your application gradle(build.gradle).

Step 2: In your application class(the class which extends Application; if not create it) Add a `@ReportsCrashes` annotation and override the `attachBaseContext()` method.

Step 3: Initialize the ACRA class in your application class

```
@ReportsCrashes(
    formUri = "Your choice of backend",
    reportType = REPORT_TYPES(JSON/FORM),
    httpMethod = HTTP_METHOD(POST/PUT),
    formUriBasicAuthLogin = "AUTH_USERNAME",
    formUriBasicAuthPassword = "AUTH_PASSWORD,
    customReportContent = {
            ReportField.USER_APP_START_DATE,
            ReportField.USER_CRASH_DATE,
            ReportField.APP_VERSION_CODE,
            ReportField.APP_VERSION_NAME,
            ReportField.ANDROID_VERSION,
            ReportField.DEVICE_ID,
            ReportField.BUILD,
            ReportField.BRAND,
            ReportField.DEVICE_FEATURES,
            ReportField.PACKAGE_NAME,
            ReportField.REPORT_ID,
            ReportField.STACK_TRACE,
    },
    mode = NOTIFICATION_TYPE(TOAST,DIALOG,NOTIFICATION)
    resToastText = R.string.crash_text_toast)

    public class MyApplication extends Application {
     @Override
            protected void attachBaseContext(Context base) {
                super.attachBaseContext(base);
                // Initialization of ACRA
                ACRA.init(this);
            }
    }
```

Where AUTH_USERNAME and AUTH_PASSWORD are the credentials of your desired backends.

Step 4: Define the Application class in AndroidManifest.xml

```
<application
      android:name=".MyApplication">
       <service></service>
```

```
        <activity></activity>
        <receiver></receiver>
</application>
```

Step 5: Make sure you have `internet` permission to receive the report from crashed application

```
<uses-permission android:name="android.permission.INTERNET" />
```

In case if you want to send the silent report to the backend then just use the below method to achieve it.

```
ACRA.getErrorReporter().handleSilentException(e);
```

Chapter 126: Check Internet Connectivity

Parameter	Detail
Context	A reference of Activity context

This method is used to check weather WI-Fi is connected or not.

Section 126.1: Check if device has internet connectivity

Add the required network permissions to the application manifest file:

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
```

```
/**
 * If network connectivity is available, will return true
 *
 * @param context the current context
 * @return boolean true if a network connection is available
 */
public static boolean isNetworkAvailable(Context context) {
    ConnectivityManager connectivity = (ConnectivityManager) context
            .getSystemService(Context.CONNECTIVITY_SERVICE);
    if (connectivity == null) {
        Log.d("NetworkCheck", "isNetworkAvailable: No");
        return false;
    }

    // get network info for all of the data interfaces (e.g. WiFi, 3G, LTE, etc.)
    NetworkInfo[] info = connectivity.getAllNetworkInfo();

    // make sure that there is at least one interface to test against
    if (info != null) {
        // iterate through the interfaces
        for (int i = 0; i < info.length; i++) {
            // check this interface for a connected state
            if (info[i].getState() == NetworkInfo.State.CONNECTED) {
                Log.d("NetworkCheck", "isNetworkAvailable: Yes");
                return true;
            }
        }
    }
    return false;
}
```

Section 126.2: How to check network strength in android?

```
        ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo Info = cm.getActiveNetworkInfo();
        if (Info == null || !Info.isConnectedOrConnecting()) {
            Log.i(TAG, "No connection");
        } else {
            int netType = Info.getType();
            int netSubtype = Info.getSubtype();

            if (netType == ConnectivityManager.TYPE_WIFI) {
                Log.i(TAG, "Wifi connection");
```

```
                WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
                List<ScanResult> scanResult = wifiManager.getScanResults();
                for (int i = 0; i < scanResult.size(); i++) {
                    Log.d("scanResult", "Speed of wifi"+scanResult.get(i).level);//The db level of
signal
                }


                // Need to get wifi strength
            } else if (netType == ConnectivityManager.TYPE_MOBILE) {
                Log.i(TAG, "GPRS/3G connection");
                // Need to get differentiate between 3G/GPRS
            }
        }
```

Section 126.3: How to check network strength

To check exact strength in decibels use this-

```
ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo Info = cm.getActiveNetworkInfo();
    if (Info == null || !Info.isConnectedOrConnecting()) {
        Log.i(TAG, "No connection");
    } else {
        int netType = Info.getType();
        int netSubtype = Info.getSubtype();

        if (netType == ConnectivityManager.TYPE_WIFI) {
            Log.i(TAG, "Wifi connection");
            WifiManager wifiManager = (WifiManager)
getApplication().getSystemService(Context.WIFI_SERVICE);
            List<ScanResult> scanResult = wifiManager.getScanResults();
            for (int i = 0; i < scanResult.size(); i++) {
                Log.d("scanResult", "Speed of wifi"+scanResult.get(i).level);//The db level of
signal
            }


            // Need to get wifi strength
        } else if (netType == ConnectivityManager.TYPE_MOBILE) {
            Log.i(TAG, "GPRS/3G connection");
            // Need to get differentiate between 3G/GPRS
        }
    }
```

To check Network type use this Class-

```
 public class Connectivity {
        /*
         * These constants aren't yet available in my API level (7), but I need to
         * handle these cases if they come up, on newer versions
         */
        public static final int NETWORK_TYPE_EHRPD = 14; // Level 11
        public static final int NETWORK_TYPE_EVDO_B = 12; // Level 9
        public static final int NETWORK_TYPE_HSPAP = 15; // Level 13
        public static final int NETWORK_TYPE_IDEN = 11; // Level 8
        public static final int NETWORK_TYPE_LTE = 13; // Level 11

    /**
```

```
     * Check if there is any connectivity
     *
     * @param context
     * @return
     */
    public static boolean isConnected(Context context) {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo info = cm.getActiveNetworkInfo();
        return (info != null && info.isConnected());
    }

    /**
     * Check if there is fast connectivity
     *
     * @param context
     * @return
     */
    public static String isConnectedFast(Context context) {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo info = cm.getActiveNetworkInfo();

        if ((info != null && info.isConnected())) {
            return Connectivity.isConnectionFast(info.getType(),
                    info.getSubtype());
        } else
            return "No NetWork Access";

    }

    /**
     * Check if the connection is fast
     *
     * @param type
     * @param subType
     * @return
     */
    public static String isConnectionFast(int type, int subType) {
        if (type == ConnectivityManager.TYPE_WIFI) {
            System.out.println("CONNECTED VIA WIFI");
            return "CONNECTED VIA WIFI";
        } else if (type == ConnectivityManager.TYPE_MOBILE) {
            switch (subType) {
            case TelephonyManager.NETWORK_TYPE_1xRTT:
                return "NETWORK TYPE 1xRTT"; // ~ 50-100 kbps
            case TelephonyManager.NETWORK_TYPE_CDMA:
                return "NETWORK TYPE CDMA (3G) Speed: 2 Mbps"; // ~ 14-64 kbps
            case TelephonyManager.NETWORK_TYPE_EDGE:

                return "NETWORK TYPE EDGE (2.75G) Speed: 100-120 Kbps"; // ~
                                                                        // 50-100
                                                                        // kbps
            case TelephonyManager.NETWORK_TYPE_EVDO_0:
                return "NETWORK TYPE EVDO_0"; // ~ 400-1000 kbps
            case TelephonyManager.NETWORK_TYPE_EVDO_A:
                return "NETWORK TYPE EVDO_A"; // ~ 600-1400 kbps
            case TelephonyManager.NETWORK_TYPE_GPRS:
                return "NETWORK TYPE GPRS (2.5G) Speed: 40-50 Kbps"; // ~ 100
                                                                     // kbps
            case TelephonyManager.NETWORK_TYPE_HSDPA:
                return "NETWORK TYPE HSDPA (4G) Speed: 2-14 Mbps"; // ~ 2-14
```

```
                                                                    // Mbps
            case TelephonyManager.NETWORK_TYPE_HSPA:
                return "NETWORK TYPE HSPA (4G) Speed: 0.7-1.7 Mbps"; // ~
                                                                        // 700-1700
                                                                        // kbps
            case TelephonyManager.NETWORK_TYPE_HSUPA:
                return "NETWORK TYPE HSUPA (3G) Speed: 1-23 Mbps"; // ~ 1-23
                                                                    // Mbps
            case TelephonyManager.NETWORK_TYPE_UMTS:
                return "NETWORK TYPE UMTS (3G) Speed: 0.4-7 Mbps"; // ~ 400-7000
                                                                    // kbps
                // NOT AVAILABLE YET IN API LEVEL 7
            case Connectivity.NETWORK_TYPE_EHRPD:
                return "NETWORK TYPE EHRPD"; // ~ 1-2 Mbps
            case Connectivity.NETWORK_TYPE_EVDO_B:
                return "NETWORK_TYPE_EVDO_B"; // ~ 5 Mbps
            case Connectivity.NETWORK_TYPE_HSPAP:
                return "NETWORK TYPE HSPA+ (4G) Speed: 10-20 Mbps"; // ~ 10-20
                                                                    // Mbps
            case Connectivity.NETWORK_TYPE_IDEN:
                return "NETWORK TYPE IDEN"; // ~25 kbps
            case Connectivity.NETWORK_TYPE_LTE:
                return "NETWORK TYPE LTE (4G) Speed: 10+ Mbps"; // ~ 10+ Mbps
                // Unknown
            case TelephonyManager.NETWORK_TYPE_UNKNOWN:
                return "NETWORK TYPE UNKNOWN";
            default:
                return "";
            }
        } else {
            return "";
        }
    }

}
```

Chapter 127: Creating your own libraries for Android applications

Section 127.1: Create a library available on Jitpack.io

Perform the following steps to create the library:

1. Create a GitHub account.
2. Create a Git repository containing your library project.
3. Modify your library project's `build.gradle` file by adding the following code:

```
apply plugin: 'com.github.dcendents.android-maven'

...

// Build a jar with source files.
task sourcesJar(type: Jar) {
    from android.sourceSets.main.java.srcDirs
    classifier = 'sources'
}

task javadoc(type: Javadoc) {
    failOnError  false
    source = android.sourceSets.main.java.sourceFiles
    classpath += project.files(android.getBootClasspath().join(File.pathSeparator))
    classpath += configurations.compile
}

// Build a jar with javadoc.
task javadocJar(type: Jar, dependsOn: javadoc) {
    classifier = 'javadoc'
    from javadoc.destinationDir
}

artifacts {
    archives sourcesJar
    archives javadocJar
}
```

   Make sure that you commit/push the above changes to GitHub.

4. Create a release from the current code on Github.
5. Run `gradlew install` on your code.
6. Your library is now available by the following dependency:

```
compile 'com.github.[YourUser]:[github repository name]:[release tag]'
```

Section 127.2: Creating library project

To create a libary , you should use `File -> New -> New Module -> Android Library`. This will create a basic library project.

When that's done, you must have a project that is set up the following manner:

```
[project root directory]
    [library root directory]
    [gradle]
    build.gradle //project level
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle //this is important!
```

Your `settings.gradle` file must contain the following:

```
include ':[library root directory]'
```

Your `[library root directory]` must contain the following:

```
[libs]
[src]
   [main]
      [java]
         [library package]
   [test]
      [java]
         [library package]
build.gradle //"app"-level
proguard-rules.pro
```

Your "app"-level `build.gradle` file must contain the following:

```
apply plugin: 'com.android.library'

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        minSdkVersion 14
        targetSdkVersion 23
    }
}
```

With that, your project should be working fine!

Section 127.3: Using library in project as a module

To use the library, you must include it as a dependency with the following line:

```
compile project(':[library root directory]')
```

Chapter 128: Device Display Metrics

Section 128.1: Get the screens pixel dimensions

To retrieve the screens width and height in pixels, we can make use of the WindowManagers display metrics.

```
// Get display metrics
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

These DisplayMetrics hold a series of information about the devices screen, like its density or size:

```
// Get width and height in pixel
Integer heightPixels = metrics.heightPixels;
Integer widthPixels = metrics.widthPixels;
```

Section 128.2: Get screen density

To get the screens density, we also can make use of the Windowmanagers DisplayMetrics. This is a quick example:

```
// Get density in dpi
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
int densityInDpi =  metrics.densityDpi;
```

Section 128.3: Formula px to dp, dp to px conversation

DP to Pixel:

```
private int dpToPx(int dp)
{
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);
}
```

Pixel to DP:

```
private int pxToDp(int px)
{
    return (int) (px / Resources.getSystem().getDisplayMetrics().density);
}
```

Chapter 129: Building Backwards Compatible Apps

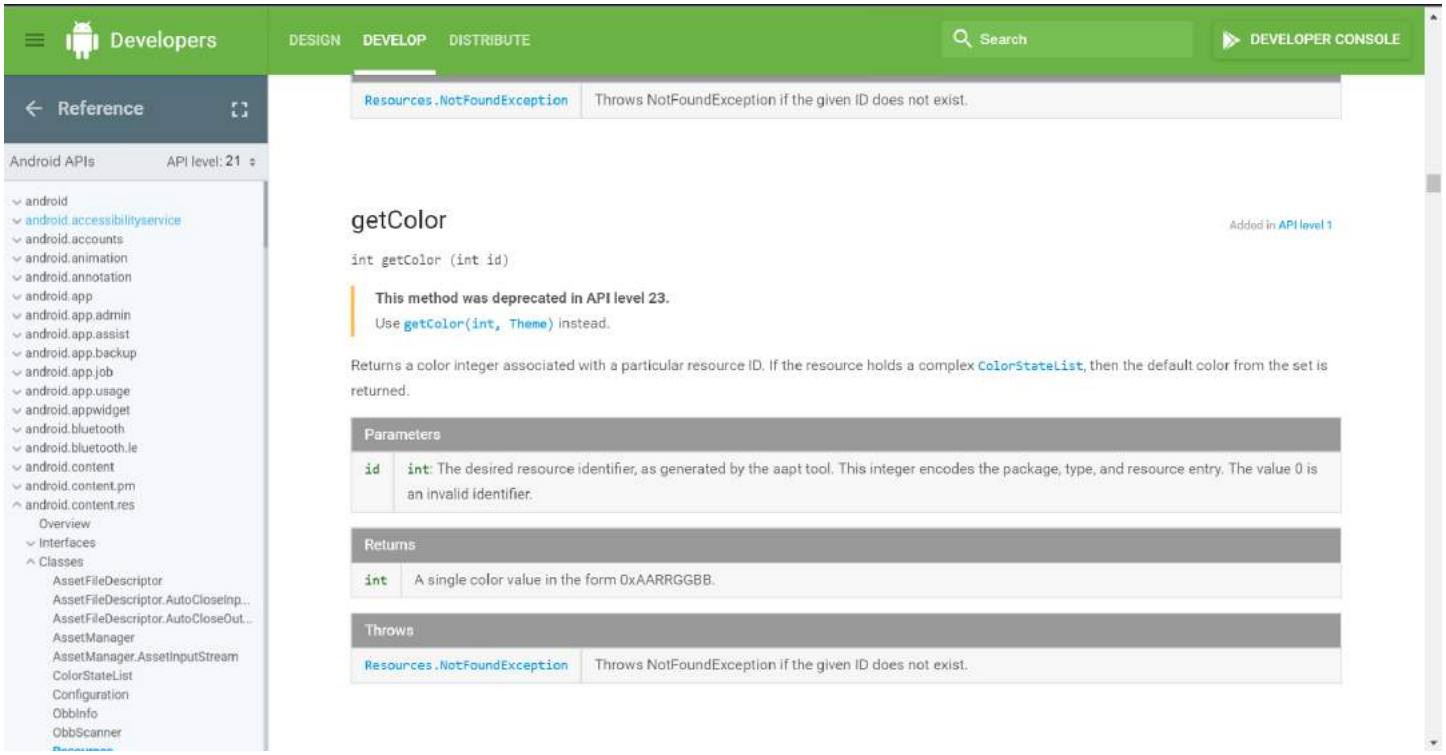
Section 129.1: How to handle deprecated API

It is unlikely for a developer to not come across a deprecated API during a development process. A deprecated program element is one that programmers are discouraged from using, typically because it is dangerous, or because a better alternative exists. Compilers and analyzers (like LINT) warn when a deprecated program element is used or overridden in non-deprecated code.

A deprecated API is usually identified in Android Studio using a strikeout. In the example below, the method `.getColor(int id)` is deprecated:

```
getResources().getColor(R.color.colorAccent));
```

If possible, developers are encouraged to use alternative APIs and elements. It is possible to check backwards compatibility of a library by visiting the Android documentation for the library and checking the "Added in API level x" section:



In the case that the API you need to use is not compatible with the Android version that your users are using, you should check for the API level of the user before using that library. For example:

```
//Checks the API level of the running device
if (Build.VERSION.SDK_INT < 23) {
    //use for backwards compatibility with API levels below 23
    int color = getResources().getColor(R.color.colorPrimary);
} else {
    int color = getResources().getColor(R.color.colorPrimary, getActivity().getTheme());
}
```

Using this method ensures that your app will remain compatible with new Android versions as well as existing versions.

Easier alternative: Use the Support Library

If the Support Libraries are used, often there are static helper methods to accomplish the same task with less client code. Instead of the if/else block above, just use:

```
final int color = android.support.v4.content.ContextCompat
    .getColor(context, R.color.colorPrimary);
```

Most deprecated methods that have newer methods with a different signature and many new features that may not have been able to be used on older versions have compatibility helper methods like this. To find others, browse through the support library for classes like `ContextCompat`, `ViewCompat`, etc.

Chapter 130: Loader

Class	Description
LoaderManager	An abstract class associated with an Activity or Fragment for managing one or more Loader instances.
LoaderManager.LoaderCallbacks	A callback interface for a client to interact with the LoaderManager.
Loader	An abstract class that performs asynchronous loading of data.
AsyncTaskLoader	Abstract loader that provides an AsyncTask to do the work.
CursorLoader	A subclass of AsyncTaskLoader that queries the ContentResolver and returns a Cursor.

Loader is good choice for prevent memory leak if you want to load data in background when oncreate method is called. For example when we execute Asynctask in oncreate method and we rotate the screen so the activity will recreate which will execute another AsyncTask again, so probably two Asyntask running in parallel together rather than like loader which will continue the background process we executed before.

Section 130.1: Basic AsyncTaskLoader

`AsyncTaskLoader` is an abstract `Loader` that provides an `AsyncTask` to do the work.

Here some basic implementation:

```
final class BasicLoader extends AsyncTaskLoader<String> {

    public BasicLoader(Context context) {
        super(context);
    }

    @Override
    public String loadInBackground() {
        // Some work, e.g. load something from internet
        return "OK";
    }

    @Override
    public void deliverResult(String data) {
        if (isStarted()) {
            // Deliver result if loader is currently started
            super.deliverResult(data);
        }
    }

    @Override
    protected void onStartLoading() {
        // Start loading
        forceLoad();
    }

    @Override
    protected void onStopLoading() {
        cancelLoad();
    }

    @Override
    protected void onReset() {
        super.onReset();
```

```
        // Ensure the loader is stopped
        onStopLoading();
    }
}
```

Typically `Loader` is initialized within the activity's `onCreate()` method, or within the fragment's `onActivityCreated()`. Also usually activity or fragment implements `LoaderManager.LoaderCallbacks` interface:

```
public class MainActivity extends Activity implements LoaderManager.LoaderCallbacks<String> {

    // Unique id for loader
    private static final int LDR_BASIC_ID = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Initialize loader; Some data can be passed as second param instead of Bundle.Empty
        getLoaderManager().initLoader(LDR_BASIC_ID, Bundle.EMPTY, this);
    }

    @Override
    public Loader<String> onCreateLoader(int id, Bundle args) {
        return new BasicLoader(this);
    }

    @Override
    public void onLoadFinished(Loader<String> loader, String data) {
        Toast.makeText(this, data, Toast.LENGTH_LONG).show();
    }

    @Override
    public void onLoaderReset(Loader<String> loader) {
    }
}
```

In this example, when loader completed, toast with result will be shown.

Section 130.2: AsyncTaskLoader with cache

It's a good practice to cache loaded result to avoid multiple loading of same data.
To invalidate cache `onContentChanged()` should be called. If loader has been already started, `forceLoad()` will be called, otherwise (if loader in stopped state) loader will be able to understand content change with `takeContentChanged()` check.

Remark: `onContentChanged()` must be called from the process's main thread.

Javadocs says about takeContentChanged():

> Take the current flag indicating whether the loader's content had changed while it was stopped. If it had, true is returned and the flag is cleared.

```
public abstract class BaseLoader<T> extends AsyncTaskLoader<T> {

    // Cached result saved here
    private final AtomicReference<T> cache = new AtomicReference<>();
```

```
    public BaseLoader(@NonNull final Context context) {
        super(context);
    }

    @Override
    public final void deliverResult(final T data) {
        if (!isReset()) {
            // Save loaded result
            cache.set(data);
            if (isStarted()) {
                super.deliverResult(data);
            }
        }
    }

    @Override
    protected final void onStartLoading() {
        // Register observers
        registerObserver();

        final T cached = cache.get();
        // Start new loading if content changed in background
        // or if we never loaded any data
        if (takeContentChanged() || cached == null) {
            forceLoad();
        } else {
            deliverResult(cached);
        }
    }

    @Override
    public final void onStopLoading() {
        cancelLoad();
    }

    @Override
    protected final void onReset() {
        super.onReset();
        onStopLoading();
        // Clear cache and remove observers
        cache.set(null);
        unregisterObserver();
    }

    /* virtual */
    protected void registerObserver() {
        // Register observers here, call onContentChanged() to invalidate cache
    }

    /* virtual */
    protected void unregisterObserver() {
        // Remove observers
    }
}
```

Section 130.3: Reloading

To invalidate your old data and restart existing loader you can use `restartLoader()` method:

```
private void reload() {
    getLoaderManager().reastartLoader(LOADER_ID, Bundle.EMPTY, this);
```

```
}
```

Section 130.4: Pass parameters using a Bundle

You can pass parameters by Bundle:

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

Get the value in onCreateLoader:

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

Chapter 131: ProGuard - Obfuscating and Shrinking your code

Section 131.1: Rules for some of the widely used Libraries

Currently it contains rules for following libraries:

1. ButterKnife
2. RxJava
3. Android Support Library
4. Android Design Support Library
5. Retrofit
6. Gson and Jackson
7. Otto
8. Crashlitycs
9. Picasso
10. Volley
11. OkHttp3
12. Parcelable

```
#Butterknife
-keep class butterknife.** { *; }
-keepnames class * { @butterknife.Bind *;}

-dontwarn butterknife.internal.**
-keep class **$$ViewBinder { *; }

-keepclasseswithmembernames class * {
    @butterknife.* <fields>;
}

-keepclasseswithmembernames class * {
    @butterknife.* <methods>;
}

# rxjava
-keep class rx.schedulers.Schedulers {
    public static <methods>;
}
-keep class rx.schedulers.ImmediateScheduler {
    public <methods>;
}
-keep class rx.schedulers.TestScheduler {
    public <methods>;
}
-keep class rx.schedulers.Schedulers {
    public static ** test();
}
-keepclassmembers class rx.internal.util.unsafe.*ArrayQueue*Field* {
    long producerIndex;
    long consumerIndex;
}
-keepclassmembers class rx.internal.util.unsafe.BaseLinkedQueueProducerNodeRef {
    long producerNode;
    long consumerNode;
}
```

```
# Support library
-dontwarn android.support.**
-dontwarn android.support.v4.**
-keep class android.support.v4.** { *; }
-keep interface android.support.v4.** { *; }
-dontwarn android.support.v7.**
-keep class android.support.v7.** { *; }
-keep interface android.support.v7.** { *; }

# support design
-dontwarn android.support.design.**
-keep class android.support.design.** { *; }
-keep interface android.support.design.** { *; }
-keep public class android.support.design.R$* { *; }

# retrofit
-dontwarn okio.**
-keepattributes Signature
-keepattributes *Annotation*
-keep class com.squareup.okhttp.** { *; }
-keep interface com.squareup.okhttp.** { *; }
-dontwarn com.squareup.okhttp.**

-dontwarn rx.**
-dontwarn retrofit.**
-keep class retrofit.** { *; }
-keepclasseswithmembers class * {
    @retrofit.http.* <methods>;
}

-keep class sun.misc.Unsafe { *; }
#your package path where your gson models are stored
-keep class com.abc.model.** { *; }

# Keep these for GSON and Jackson
-keepattributes Signature
-keepattributes *Annotation*
-keepattributes EnclosingMethod
-keep class sun.misc.Unsafe { *; }
-keep class com.google.gson.** { *; }

#keep otto
-keepattributes *Annotation*
-keepclassmembers class ** {
    @com.squareup.otto.Subscribe public *;
    @com.squareup.otto.Produce public *;
}

# Crashlitycs 2.+
-keep class com.crashlytics.** { *; }
-keep class com.crashlytics.android.**
-keepattributes SourceFile, LineNumberTable, *Annotation*
# If you are using custom exceptions, add this line so that custom exception types are skipped
during obfuscation:
-keep public class * extends java.lang.Exception
# For Fabric to properly de-obfuscate your crash reports, you need to remove this line from your
ProGuard config:
# -printmapping mapping.txt

# Picasso
-dontwarn com.squareup.okhttp.**
```

```
# Volley
-keep class com.android.volley.toolbox.ImageLoader { *; }

# OkHttp3
-keep class okhttp3.** { *; }
-keep interface okhttp3.** { *; }
-dontwarn okhttp3.**

# Needed for Parcelable/SafeParcelable Creators to not get stripped
-keepnames class * implements android.os.Parcelable {
    public static final ** CREATOR;
}
```

Section 131.2: Remove trace logging (and other) statements at build time

If you want to remove calls to certain methods, assuming they return void and have no side affects (as in, calling them doesn't change any system values, reference arguments, statics, etc.) then you can have ProGuard remove them from the output after the build is complete.

For example, I find this useful in removing debug/verbose logging statements useful in debugging, but generating the strings for them is unnecessary in production.

```
# Remove the debug and verbose level Logging statements.
# That means the code to generate the arguments to these methods will also not be called.
# ONLY WORKS IF -dontoptimize IS _NOT_ USED in any ProGuard configs
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    public static *** v(...);
}
```

Note: If `-dontoptimize` is used in any ProGuard config so that it is not minifying/removing unused code, then this will not strip out the statements. (But who would not want to remove unused code, right?)

Note2: this call will remove the call to log, but will not protect you code. The Strings will actually remain in the generated apk. Read more in this post.

Section 131.3: Protecting your code from hackers

Obfuscation is often considered as a magic solution for code protection, by making your code harder to understand if it ever gets de-compiled by hackers.

But if you're thinking that removing the `Log.x(..)` actually removes the information the hackers need, you'll have a nasty surprise.

Removing all your log calls with:

```
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    ...etc
}
```

will indeed remove the Log call itself, but usually *not* the Strings you put into them.

If for example inside your log call you type a common log message such as: `Log.d(MyTag,"Score="+score);`, the compiler converts the `+` to a 'new StringBuilder()' outside the Log call. ProGuard doesn't change this new object.

Your de-compiled code will still have a hanging `StringBuilder` for `"Score="`, appended with the obfuscated version for `score` variable (let's say it was converted to `b`).
Now the hacker knows what is `b`, and make sense of your code.

A good practice to actually remove these residuals from your code is either not put them there in the first place (Use String formatter instead, with proguard rules to remove them), or to wrap your `Log` calls with:

```
    if (BuildConfig.DEBUG) {
        Log.d(TAG,".."+var);
    }
```

Tip:

Test how well protected your obfuscated code is by de-compiling it yourself!

1. dex2jar - converts the apk to jar
2. jd - decompiles the jar and opens it in a gui editor

Section 131.4: Enable ProGuard for your build

For enabling `ProGuard` configurations for your application you need to enable it in your module level gradle file. you need to set the value of `minifyEnabled true`.

You can also enable `shrinkResources true` which will remove resources that `ProGuard` flaggs as unused.

```
buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
```

The above code will apply your ProGuard configurations contained in `proguard-rules.pro` ("proguard-project.txt" in Eclipse) to your released apk.

To enable you to later determine the line on which an exception occurred in a stack trace, "proguard-rules.pro" should contain following lines:

```
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
```

To enable Proguard in Eclipse add `proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt` to "project.properties"

Section 131.5: Enabling ProGuard with a custom obfuscation configuration file

ProGuard allows the developer to obfuscate, shrink and optimize his code.

#1 The first step of the procedure is to enable proguard on the build.

This can be done by **setting the 'minifyEnabled' command to true** on your desired build

#2 The second step is to specify which proguard files are we using for the given build

This can be done by **setting the 'proguardFiles' line with the proper filenames**

```
buildTypes {
    debug {
        minifyEnabled false
    }
    testRelease {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules-tests.pro'
    }
    productionRelease {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules-tests.pro', 'proguard-rules-release.pro'
    }
}
```

#3 The developer can then edit his proguard file with the rules he desires.

That can be done by editting the file (for example 'proguard-rules-tests.pro') and adding the desired constraints.
The following file serves as an example proguard file

```
// default & basic optimization configurations
-optimizationpasses 5
-dontpreverify
-repackageclasses ''
-allowaccessmodification
-optimizations !code/simplification/arithmetic
-keepattributes *Annotation*

-verbose

-dump obfuscation/class_files.txt
-printseeds obfuscation/seeds.txt
-printusage obfuscation/unused.txt // unused classes that are stripped out in the process
-printmapping obfuscation/mapping.txt // mapping file that shows the obfuscated names of the classes after proguad is applied

// the developer can specify keywords for the obfuscation (I myself use fruits for obfuscation names once in a while :-) )
-obfuscationdictionary obfuscation/keywords.txt
-classobfuscationdictionary obfuscation/keywords.txt
-packageobfuscationdictionary obfuscation/keywords.txt
```

Finally, whenever the developer runs and/or generates his new .APK file, the custom proguard configurations will be applied thus fulfilling the requirements.

Chapter 132: Typedef Annotations: @IntDef, @StringDef

Section 132.1: IntDef Annotations

This annotation ensures that only the valid integer constants that you expect are used.
The following example illustrates the steps to create an annotation:

```
import android.support.annotation.IntDef;

public abstract class Car {

    //Define the list of accepted constants
    @IntDef({MICROCAR, CONVERTIBLE, SUPERCAR, MINIVAN, SUV})

    //Tell the compiler not to store annotation data in the .class file
    @Retention(RetentionPolicy.SOURCE)
    //Declare the CarType annotation
    public @interface CarType {}

    //Declare the constants
    public static final int MICROCAR = 0;
    public static final int CONVERTIBLE = 1;
    public static final int SUPERCAR = 2;
    public static final int MINIVAN = 3;
    public static final int SUV = 4;

    @CarType
    private int mType;

    @CarType
    public int getCarType(){
        return mType;
    };

    public void setCarType(@CarType int type){
        mType = type;
    }
}
```

They also enable code completion to automatically offer the allowed constants.
When you build this code, a warning is generated if the type parameter does not reference one of the defined constants.

Section 132.2: Combining constants with flags

Using the `IntDef#flag()` attribute set to **`true`**, multiple constants can be combined.

Using the same example in this topic:

```
public abstract class Car {

    //Define the list of accepted constants
    @IntDef(flag=true, value={MICROCAR, CONVERTIBLE, SUPERCAR, MINIVAN, SUV})

    //Tell the compiler not to store annotation data in the .class file
    @Retention(RetentionPolicy.SOURCE)
```

```
    .....

}
```

Users can combine the allowed constants with a flag (such as `|`, `&`, `^`).

Chapter 133: Capturing Screenshots

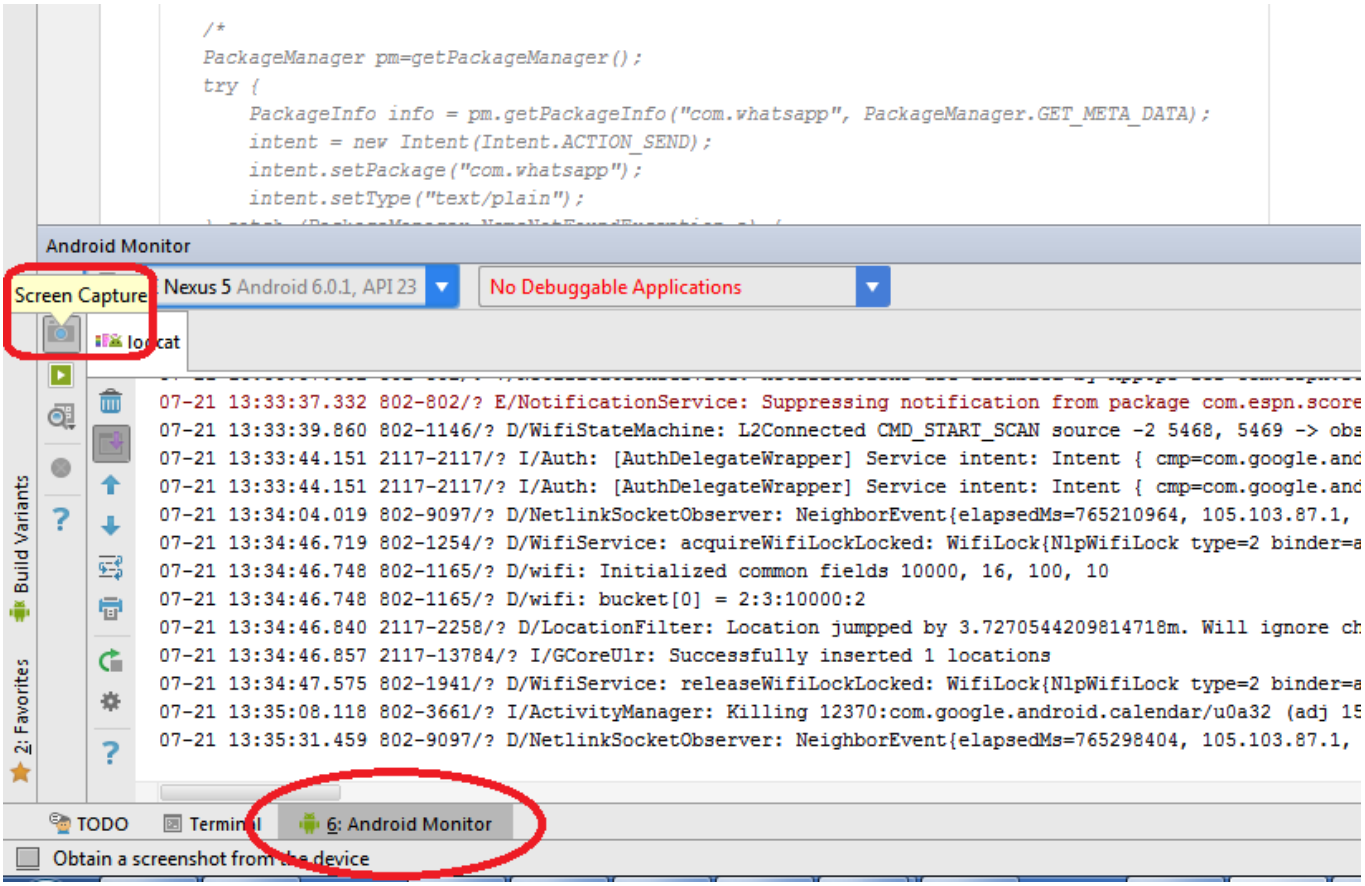
Section 133.1: Taking a screenshot of a particular view

If you want to take a screenshot of a particular View `v`, then you can use the following code:

```
Bitmap viewBitmap = Bitmap.createBitmap(v.getWidth(), v.getHeight(), Bitmap.Config.RGB_565);
Canvas viewCanvas = new Canvas(viewBitmap);
Drawable backgroundDrawable = v.getBackground();

if(backgroundDrawable != null){
    // Draw the background onto the canvas.
    backgroundDrawable.draw(viewCanvas);
}
else{
    viewCanvas.drawColor(Color.GREEN);
    // Draw the view onto the canvas.
    v.draw(viewCanvas)
}

// Write the bitmap generated above into a file.
String fileStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
OutputStream outputStream = null;
try{
    imgFile = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), fileStamp +
".png");
    outputStream = new FileOutputStream(imgFile);
    viewBitmap.compress(Bitmap.CompressFormat.PNG, 40, outputStream);
    outputStream.close();
}
catch(Exception e){
    e.printStackTrace();
}
```

Section 133.2: Capturing Screenshot via Android Studio

1. Open Android Monitor Tab
2. Click on Screen Capture Button



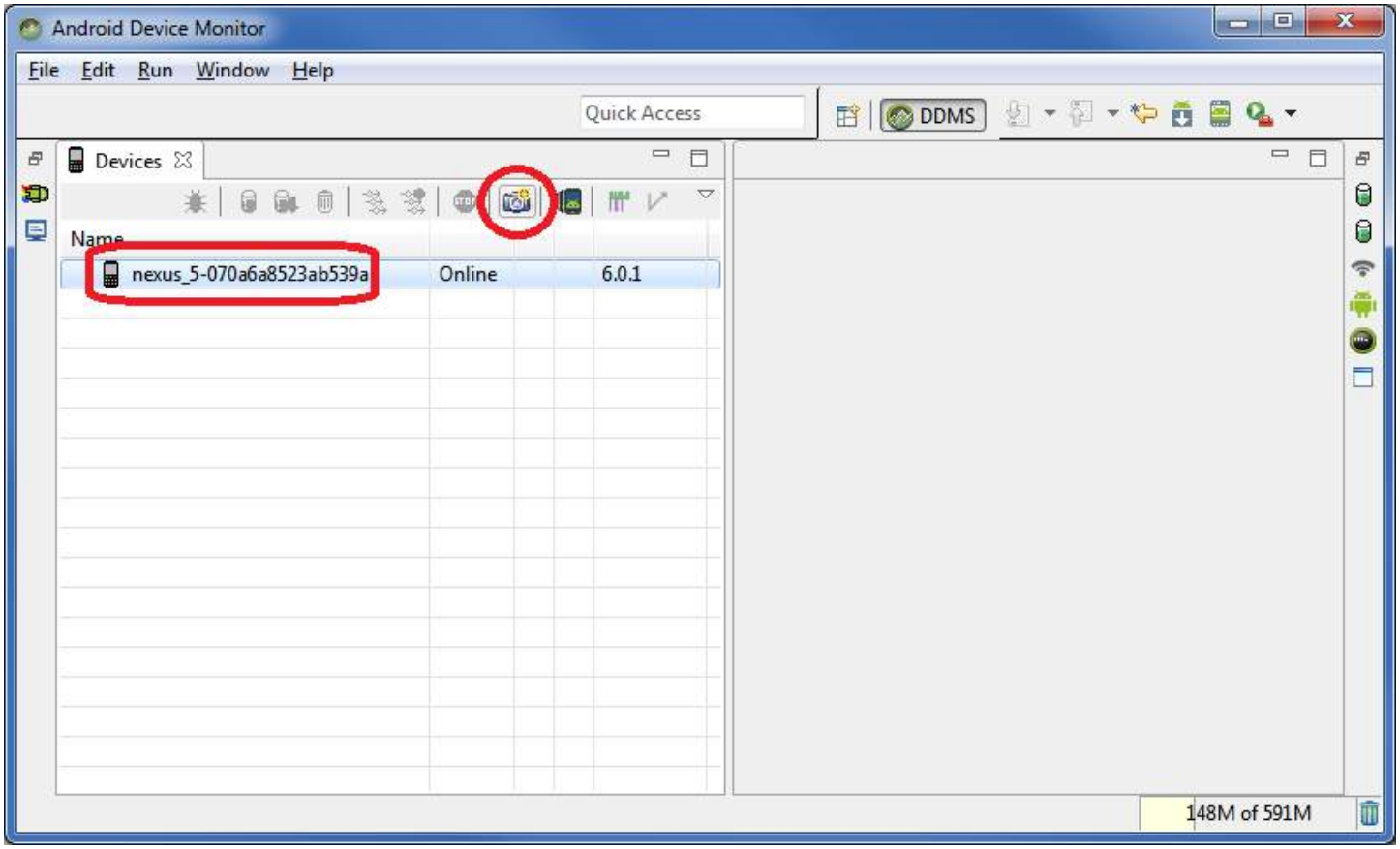
Section 133.3: Capturing Screenshot via ADB and saving directly in your PC

If you use Linux (or Windows with Cygwin), you can run:

```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```

Section 133.4: Capturing Screenshot via Android Device Monitor

1. Open Android Device Monitor (*ie C:<ANDROID_SDK_LOCATION>\tools\monitor.bat*)
2. Select your device
3. Click on Screen Capture Button



Section 133.5: Capturing Screenshot via ADB

Example below saves a screenshot on Devices's Internal Storage.

```
adb shell screencap /sdcard/screen.png
```

Chapter 134: MVP Architecture

This topic will provide Model-View-Presenter (MVP) architecture of Android with various examples.

Section 134.1: Login example in the Model View Presenter (MVP) pattern

Let's see MVP in action using a simple Login Screen. There are two `Button`s—one for login action and another for a registration screen; two `EditTexts`—one for the email and the other for the password.

LoginFragment (The View)

```
public class LoginFragment extends Fragment implements LoginContract.PresenterToView,
View.OnClickListener {

    private View view;
    private EditText email, password;
    private Button login, register;

    private LoginContract.ToPresenter presenter;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.fragment_login, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        email = (EditText) view.findViewById(R.id.email_et);
        password = (EditText) view.findViewById(R.id.password_et);
        login = (Button) view.findViewById(R.id.login_btn);
        login.setOnClickListener(this);
        register = (Button) view.findViewById(R.id.register_btn);
        register.setOnClickListener(this);

        presenter = new LoginPresenter(this);

        presenter.isLoggedIn();

    }

    @Override
    public void onLoginResponse(boolean isLoginSuccess) {
        if (isLoginSuccess) {
            startActivity(new Intent(getActivity(), MapActivity.class));
            getActivity().finish();
        }
    }

    @Override
    public void onError(String message) {
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void isLoggedIn(boolean isLoggedIn) {
        if (isLoggedIn) {
```

```
            startActivity(new Intent(getActivity(), MapActivity.class));
            getActivity().finish();
        }
    }

    @Override
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.login_btn:
                LoginItem loginItem = new LoginItem();
                loginItem.setPassword(password.getText().toString().trim());
                loginItem.setEmail(email.getText().toString().trim());
                presenter.login(loginItem);
                break;
            case R.id.register_btn:
                startActivity(new Intent(getActivity(), RegisterActivity.class));
                getActivity().finish();
                break;
        }
    }
}
```

LoginPresenter (The Presenter)

```
public class LoginPresenter implements LoginContract.ToPresenter {

    private LoginContract.PresenterToModel model;
    private LoginContract.PresenterToView view;

    public LoginPresenter(LoginContract.PresenterToView view) {
        this.view = view;
        model = new LoginModel(this);
    }

    @Override
    public void login(LoginItem userCredentials) {
        model.login(userCredentials);
    }

    @Override
    public void isLoggedIn() {
        model.isLoggedIn();
    }

    @Override
    public void onLoginResponse(boolean isLoginSuccess) {
        view.onLoginResponse(isLoginSuccess);
    }

    @Override
    public void onError(String message) {
        view.onError(message);
    }

    @Override
    public void isloggedIn(boolean isLoggedin) {
        view.isLoggedIn(isLoggedin);
    }
}
```

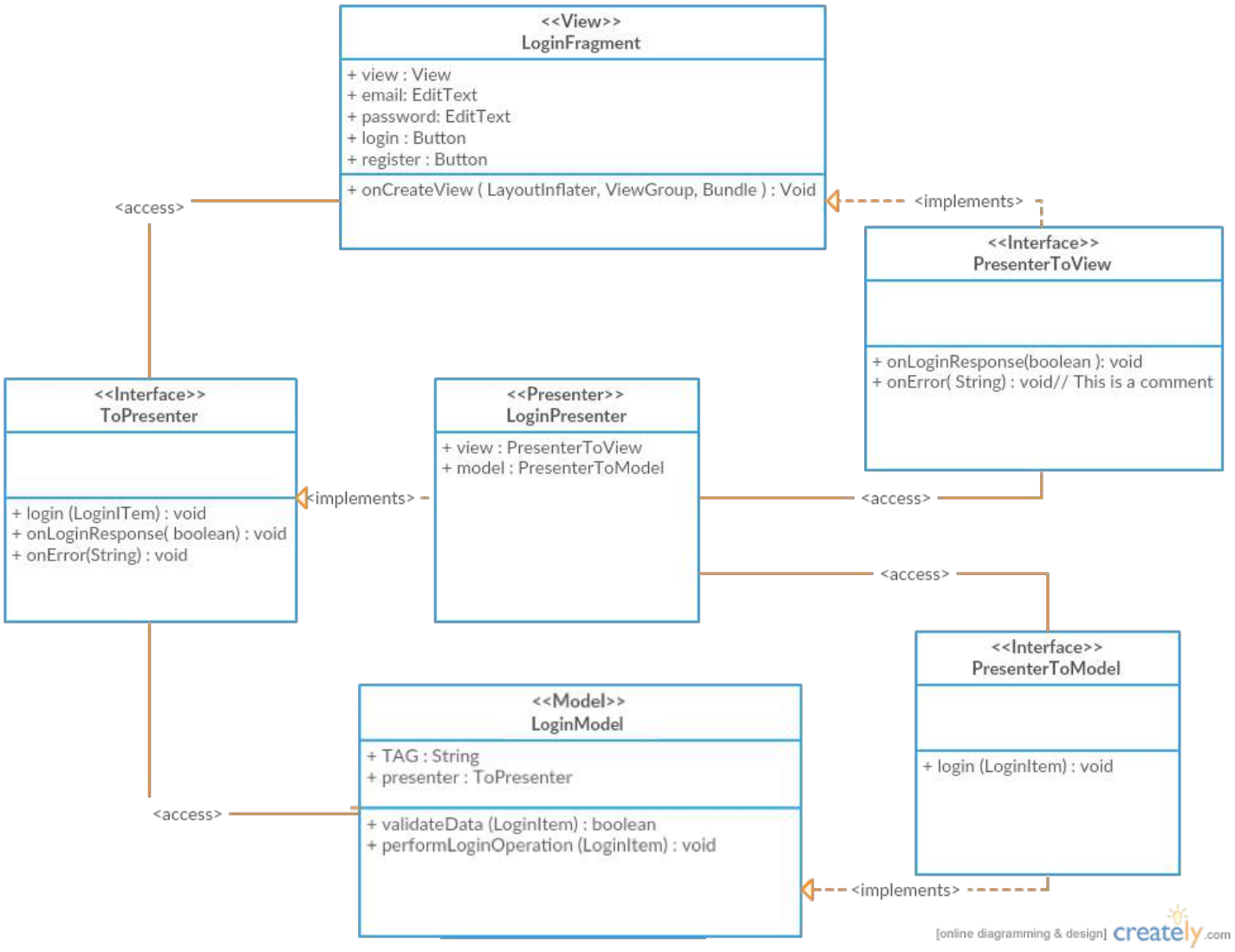
LoginModel (The Model)

```
public class LoginModel implements LoginContract.PresenterToModel,
ResponseErrorListener.ErrorListener {

    private static final String TAG = LoginModel.class.getSimpleName();
    private LoginContract.ToPresenter presenter;

    public LoginModel(LoginContract.ToPresenter presenter) {
        this.presenter = presenter;
    }

    @Override
    public void login(LoginItem userCredentials) {
        if (validateData(userCredentials)) {
            try {
                performLoginOperation(userCredentials);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        } else {

presenter.onError(BaseContext.getContext().getString(R.string.error_login_field_validation));
        }
    }

    @Override
    public void isLoggedIn() {
        DatabaseHelper database = new DatabaseHelper(BaseContext.getContext());
        presenter.isloggedIn(database.isLoggedIn());
    }

    private boolean validateData(LoginItem userCredentials) {
        return Patterns.EMAIL_ADDRESS.matcher(userCredentials.getEmail()).matches()
                && !userCredentials.getPassword().trim().equals("");
    }

    private void performLoginOperation(final LoginItem userCredentials) throws JSONException {

        JSONObject postData = new JSONObject();
        postData.put(Constants.EMAIL, userCredentials.getEmail());
        postData.put(Constants.PASSWORD, userCredentials.getPassword());

        JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST, Url.AUTH, postData,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        try {
                            String token = response.getString(Constants.ACCESS_TOKEN);
                            DatabaseHelper databaseHelper = new
DatabaseHelper(BaseContext.getContext());
                            databaseHelper.login(token);
                            Log.d(TAG, "onResponse: " + token);
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                        presenter.onLoginResponse(true);
                    }
                }, new ErrorResponse(this));

        RequestQueue queue = Volley.newRequestQueue(BaseContext.getContext());
        queue.add(request);
    }
```

```
    @Override
    public void onError(String message) {
        presenter.onError(message);
    }
}
```

Class Diagram

Let's see the action in the form of class diagram.

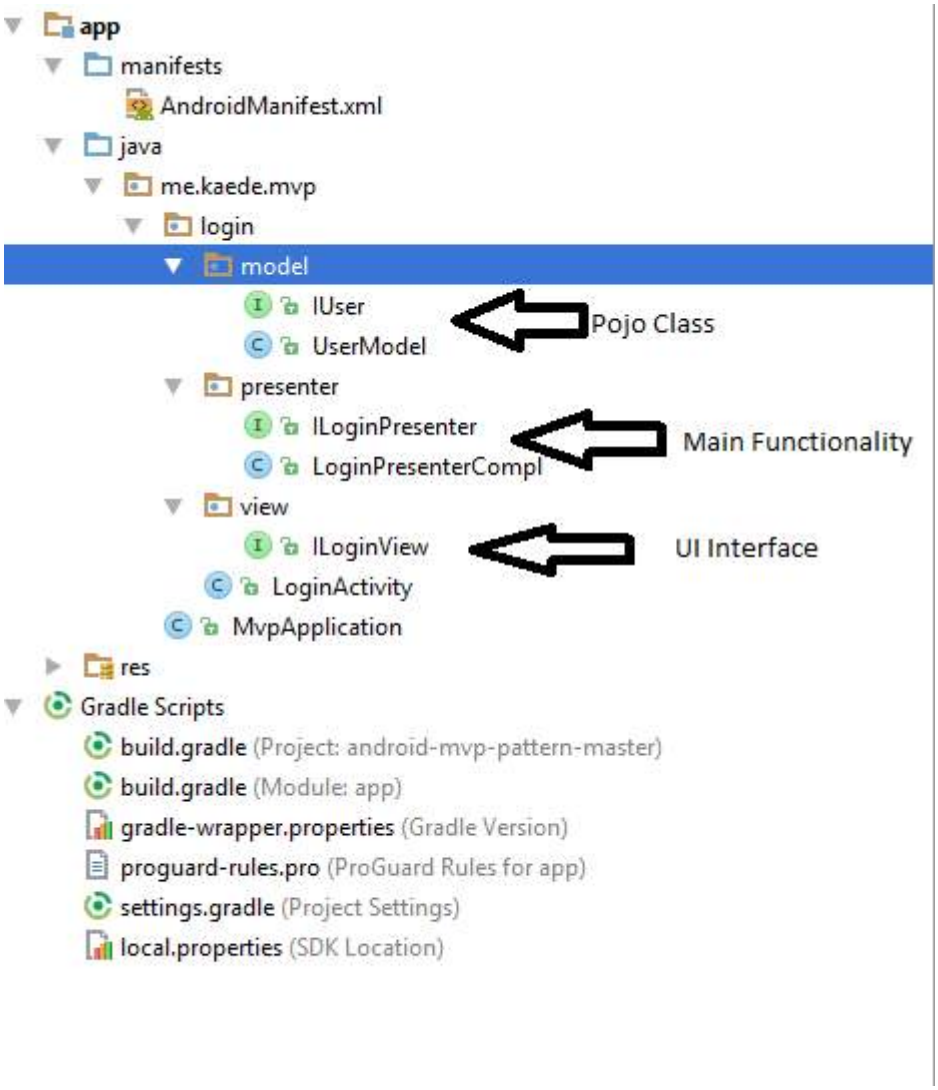


Notes:

- This example uses Volley for network communication, but this library is not required for MVP
- `UrlUtils` is a class which contains all the links for my API Endpoints
- `ResponseErrorListener.ErrorListener` is an **`interface`** that listens for error in `ErrorResponse` that **`implements`** Volley's `Response.ErrorListener`; these classes are not included here as they are not directly part of this example

Section 134.2: Simple Login Example in MVP

Required package structure



XML activity_login

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:gravity="center_vertical"
   android:orientation="vertical"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin">

    <EditText
       android:id="@+id/et_login_username"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:hint="USERNAME" />

    <EditText
       android:id="@+id/et_login_password"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:hint="PASSWORD" />

    <LinearLayout
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:orientation="horizontal">
```

```
        <Button
            android:id="@+id/btn_login_login"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="4dp"
            android:layout_weight="1"
            android:text="Login" />

        <Button
            android:id="@+id/btn_login_clear"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            android:layout_weight="1"
            android:text="Clear" />
    </LinearLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="3dp"
        android:text="correct user: mvp, mvp" />

    <ProgressBar
        android:id="@+id/progress_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp" />

</LinearLayout>
```

Activity Class LoginActivity.class

```
public class LoginActivity extends AppCompatActivity implements ILoginView, View.OnClickListener {
    private EditText editUser;
    private EditText editPass;
    private Button   btnLogin;
    private Button   btnClear;
    private ILoginPresenter loginPresenter;
    private ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        //find view
        editUser = (EditText) this.findViewById(R.id.et_login_username);
        editPass = (EditText) this.findViewById(R.id.et_login_password);
        btnLogin = (Button) this.findViewById(R.id.btn_login_login);
        btnClear = (Button) this.findViewById(R.id.btn_login_clear);
        progressBar = (ProgressBar) this.findViewById(R.id.progress_login);

        //set listener
        btnLogin.setOnClickListener(this);
        btnClear.setOnClickListener(this);

        //init
        loginPresenter = new LoginPresenterCompl(this);
        loginPresenter.setProgressBarVisiblity(View.INVISIBLE);
    }

    @Override
```

```
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.btn_login_clear:
                loginPresenter.clear();
                break;
            case R.id.btn_login_login:
                loginPresenter.setProgressBarVisiblity(View.VISIBLE);
                btnLogin.setEnabled(false);
                btnClear.setEnabled(false);
                loginPresenter.doLogin(editUser.getText().toString(),
editPass.getText().toString());
                break;
        }
    }

    @Override
    public void onClearText() {
        editUser.setText("");
        editPass.setText("");
    }

    @Override
    public void onLoginResult(Boolean result, int code) {
        loginPresenter.setProgressBarVisiblity(View.INVISIBLE);
        btnLogin.setEnabled(true);
        btnClear.setEnabled(true);
        if (result){
            Toast.makeText(this,"Login Success",Toast.LENGTH_SHORT).show();
        }
        else
            Toast.makeText(this,"Login Fail, code = " + code,Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    @Override
    public void onSetProgressBarVisibility(int visibility) {
        progressBar.setVisibility(visibility);
    }
}
```

Creating an ILoginView Interface

Create an `ILoginView` interface for update info from Presenter under view folder as follows:

```
public interface ILoginView {
    public void onClearText();
    public void onLoginResult(Boolean result, int code);
    public void onSetProgressBarVisibility(int visibility);
}
```

Creating an ILoginPresenter Interface

Create an `ILoginPresenter` interface in order to communicate with `LoginActivity` (Views) and create the `LoginPresenterCompl` class for handling login functionality and reporting back to the Activity. The `LoginPresenterCompl` class implements the `ILoginPresenter` interface:

ILoginPresenter.class

```
public interface ILoginPresenter {
```

```
    void clear();
    void doLogin(String name, String passwd);
    void setProgressBarVisiblity(int visiblity);
}
```

LoginPresenterCompl.class

```
public class LoginPresenterCompl implements ILoginPresenter {
    ILoginView iLoginView;
    IUser user;
    Handler handler;

    public LoginPresenterCompl(ILoginView iLoginView) {
        this.iLoginView = iLoginView;
        initUser();
        handler = new Handler(Looper.getMainLooper());
    }

    @Override
    public void clear() {
        iLoginView.onClearText();
    }

    @Override
    public void doLogin(String name, String passwd) {
        Boolean isLoginSuccess = true;
        final int code = user.checkUserValidity(name,passwd);
        if (code!=0) isLoginSuccess = false;
        final Boolean result = isLoginSuccess;
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                    iLoginView.onLoginResult(result, code);
            }
        }, 5000);
    }

    @Override
    public void setProgressBarVisiblity(int visiblity){
        iLoginView.onSetProgressBarVisibility(visiblity);
    }

    private void initUser(){
        user = new UserModel("mvp","mvp");
    }
}
```

Creating a UserModel

Create a `UserModel` which is like a Pojo class for `LoginActivity`. Create an `IUser` interface for Pojo validations:

UserModel.class

```
public class UserModel implements IUser {
String name;
String passwd;

public UserModel(String name, String passwd) {
    this.name = name;
    this.passwd = passwd;
}

@Override
```

```
public String getName() {
    return name;
}

@Override
public String getPasswd() {
    return passwd;
}

@Override
public int checkUserValidity(String name, String passwd){
    if (name==null||passwd==null||!name.equals(getName())||!passwd.equals(getPasswd())){
        return -1;
    }
    return 0;
}
```

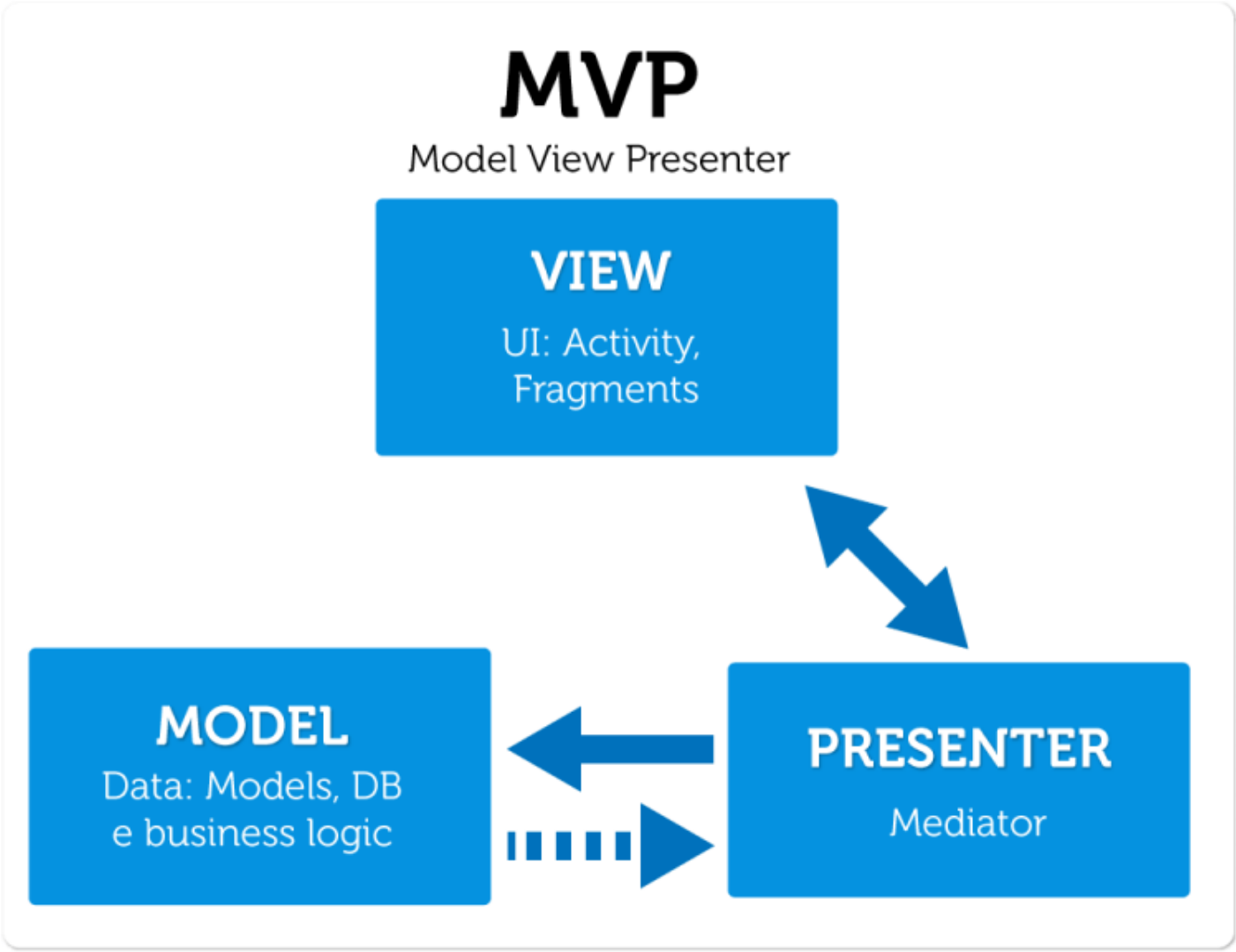
IUser.class

```
public interface IUser {
    String getName();

    String getPasswd();

    int checkUserValidity(String name, String passwd);
}
```

MVP

A Model-view-presenter (MVP) is a derivation of the model–view–controller (MVC) architectural pattern. It is used mostly for building user interfaces and offers the following benefits:

- Views are more separated from Models. The Presenter is the mediator between Model and View.
- It is easier to create unit tests.
- Generally, there is a one-to-one mapping between View and Presenter, with the possibility to use multiple Presenters for complex Views.



Chapter 135: Orientation Changes

Section 135.1: Saving and Restoring Activity State

As your activity begins to stop, the system calls `onSaveInstanceState()` so your activity can save state information with a collection of key-value pairs. The default implementation of this method automatically saves information about the state of the activity's view hierarchy, such as the text in an `EditText` widget or the scroll position of a `ListView`.

To save additional state information for your activity, you must implement `onSaveInstanceState()` and add key-value pairs to the Bundle object. For example:

```
public class MainActivity extends Activity {
    static final String SOME_VALUE = "int_value";
    static final String SOME_OTHER_VALUE = "string_value";

    @Override
    protected void onSaveInstanceState(Bundle savedInstanceState) {
        // Save custom values into the bundle
        savedInstanceState.putInt(SOME_VALUE, someIntValue);
        savedInstanceState.putString(SOME_OTHER_VALUE, someStringValue);
        // Always call the superclass so it can save the view hierarchy state
        super.onSaveInstanceState(savedInstanceState);
    }
}
```

The system will call that method before an Activity is destroyed. Then later the system will call `onRestoreInstanceState` where we can restore state from the bundle:

```
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    // Always call the superclass so it can restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);
    // Restore state members from saved instance
    someIntValue = savedInstanceState.getInt(SOME_VALUE);
    someStringValue = savedInstanceState.getString(SOME_OTHER_VALUE);
}
```

Instance state can also be restored in the standard Activity#onCreate method but it is convenient to do it in `onRestoreInstanceState` which ensures all of the initialization has been done and allows subclasses to decide whether to use the default implementation. Read this stackoverflow post for details.

Note that `onSaveInstanceState` and `onRestoreInstanceState` are not guaranteed to be called together. Android invokes `onSaveInstanceState()` when there's a chance the activity might be destroyed. However, there are cases where `onSaveInstanceState` is called but the activity is not destroyed and as a result `onRestoreInstanceState` is not invoked.

Section 135.2: Retaining Fragments

In many cases, we can avoid problems when an Activity is re-created by simply using fragments. If your views and state are within a fragment, we can easily have the fragment be retained when the activity is re-created:

```
public class RetainedFragment extends Fragment {
    // data object we want to retain
    private MyDataObject data;
```

```
    // this method is only called once for this fragment
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // retain this fragment when activity is re-initialized
        setRetainInstance(true);
    }

    public void setData(MyDataObject data) {
        this.data = data;
    }

    public MyDataObject getData() {
        return data;
    }
}
```

This approach keeps the fragment from being destroyed during the activity lifecycle. They are instead retained inside the Fragment Manager. See the Android official docs for more information.

Now you can check to see if the fragment already exists by tag before creating one and the fragment will retain it's state across configuration changes. See the Handling Runtime Changes guide for more details.

Section 135.3: Manually Managing Configuration Changes

If your application doesn't need to update resources during a specific configuration change and you have a performance limitation that requires you to avoid the activity restart, then you can declare that your activity handles the configuration change itself, which prevents the system from restarting your activity.

However, this technique should be considered a last resort when you must avoid restarts due to a configuration change and is not recommended for most applications. To take this approach, we must add the `android:configChanges` node to the activity within the **AndroidManifest.xml**:

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|screenSize|keyboardHidden"
          android:label="@string/app_name">
```

Now, when one of these configurations change, the activity does not restart but instead receives a call to `onConfigurationChanged()`:

```
// Within the activity which receives these changes
// Checks the current device orientation, and toasts accordingly
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // Checks the orientation of the screen
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "landscape", Toast.LENGTH_SHORT).show();
    } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){
        Toast.makeText(this, "portrait", Toast.LENGTH_SHORT).show();
    }
}
```

See the Handling the Change docs. For more about which configuration changes you can handle in your activity, see the android:configChanges documentation and the Configuration class.

Section 135.4: Handling AsyncTask

Problem:

- If after the `AsyncTask` starts there is a screen rotation the owning activity is destroyed and recreated.
- When the `AsyncTask` finishes it wants to update the UI that may not valid anymore.

Solution:

Using Loaders, one can easily overcome the activity destruction/recreation.

Example:

MainActivity:

```
public class MainActivity extends AppCompatActivity
        implements LoaderManager.LoaderCallbacks<Bitmap> {

    //Unique id for the loader
    private static final int MY_LOADER = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        LoaderManager loaderManager = getSupportLoaderManager();

        if(loaderManager.getLoader(MY_LOADER) == null) {
            loaderManager.initLoader(MY_LOADER, null, this).forceLoad();
        }
    }

    @Override
    public Loader<Bitmap> onCreateLoader(int id, Bundle args) {
        //Create a new instance of your Loader<Bitmap>
        MyLoader loader = new MyLoader(MainActivity.this);
        return loader;
    }

    @Override
    public void onLoadFinished(Loader<Bitmap> loader, Bitmap data) {
        // do something in the parent activity/service
        // i.e. display the downloaded image
        Log.d("MyAsyncTask", "Received result: ");
    }

    @Override
    public void onLoaderReset(Loader<Bitmap> loader) {

    }
}
```

AsyncTaskLoader:

```
public class MyLoader extends AsyncTaskLoader<Bitmap> {
    private WeakReference<Activity> motherActivity;

    public MyLoader(Activity activity) {
        super(activity);
        //We don't use this, but if you want you can use it, but remember, WeakReference
        motherActivity = new WeakReference<>(activity);
```

```
    }

    @Override
    public Bitmap loadInBackground() {
        // Do work. I.e download an image from internet to be displayed in gui.
        // i.e. return the downloaded gui
        return result;
    }
}
```

Note:

It is important to use either the v4 compatibility library or not, but do not use part of one and part of the other, as it will lead to compilation errors. To check you can look at the imports for `android.support.v4.content` and `android.content` (you shouldn't have both).

Section 135.5: Lock Screen's rotation programmatically

It is very common that during development, one may find **very useful to lock/unlock the device screen during specific parts of the code**.

*For instance, while showing a Dialog with information, the developer might want to **lock** the screen's rotation to prevent the dialog from being dismissed and the current activity from being rebuilt to **unlock** it again when the dialog is dismissed.*

Even though we can achieve rotation locking from the manifest by doing :

```
<activity
   android:name=".TheActivity"
   android:screenOrientation="portrait"
   android:label="@string/app_name" >
</activity>
```

One can do it programmatically as well by doing the following :

```
public void lockDeviceRotation(boolean value) {
    if (value) {
        int currentOrientation = getResources().getConfiguration().orientation;
        if (currentOrientation == Configuration.ORIENTATION_LANDSCAPE) {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_LANDSCAPE);
        } else {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT);
        }
    } else {
        getWindow().clearFlags(WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR2) {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_USER);
        } else {
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
        }
    }
}
```

And then calling the following, to respectively lock and unlock the device rotation

```
lockDeviceRotation(true)
```

and

```
lockDeviceRotation(false)
```

Section 135.6: Saving and Restoring Fragment State

Fragments also have a `onSaveInstanceState()` method which is called when their state needs to be saved:

```
public class MySimpleFragment extends Fragment {
    private int someStateValue;
    private final String SOME_VALUE_KEY = "someValueToSave";

    // Fires when a configuration change occurs and fragment needs to save state
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putInt(SOME_VALUE_KEY, someStateValue);
        super.onSaveInstanceState(outState);
    }
}
```

Then we can pull data out of this saved state in `onCreateView`:

```
public class MySimpleFragment extends Fragment {
   // ...

   // Inflate the view for the fragment based on layout XML
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.my_simple_fragment, container, false);
        if (savedInstanceState != null) {
            someStateValue = savedInstanceState.getInt(SOME_VALUE_KEY);
            // Do something with value if needed
        }
        return view;
   }
}
```

For the fragment state to be saved properly, we need to be sure that we aren't unnecessarily recreating the fragment on configuration changes. This means being careful not to reinitialize existing fragments when they already exist. Any fragments being initialized in an Activity need to be looked up by tag after a configuration change:

```
public class ParentActivity extends AppCompatActivity {
    private MySimpleFragment fragmentSimple;
    private final String SIMPLE_FRAGMENT_TAG = "myfragmenttag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        if (savedInstanceState != null) { // saved instance state, fragment may exist
           // look up the instance that already exists by tag
           fragmentSimple = (MySimpleFragment)
              getSupportFragmentManager().findFragmentByTag(SIMPLE_FRAGMENT_TAG);
        } else if (fragmentSimple == null) {
           // only create fragment if they haven't been instantiated already
           fragmentSimple = new MySimpleFragment();
        }
    }
}
```

This requires us to be careful to include a tag for lookup whenever putting a fragment into the activity within a

transaction:

```
public class ParentActivity extends AppCompatActivity {
    private MySimpleFragment fragmentSimple;
    private final String SIMPLE_FRAGMENT_TAG = "myfragmenttag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // ... fragment lookup or instantation from above...
        // Always add a tag to a fragment being inserted into container
        if (!fragmentSimple.isInLayout()) {
            getSupportFragmentManager()
                .beginTransaction()
                .replace(R.id.container, fragmentSimple, SIMPLE_FRAGMENT_TAG)
                .commit();
        }
    }
}
```

With this simple pattern, we can properly re-use fragments and restore their state across configuration changes.

Chapter 136: Xposed

Section 136.1: Creating a Xposed Module

Xposed is a framework that allows you to hook method calls of other apps. When you do a modification by decompiling an APK, you can insert/change commands directly wherever you want. However, you will need to recompile/sign the APK afterwards and you can only distribute the whole package. With Xposed, you can inject your own code before or after methods, or replace whole methods completely. Unfortunately, you can only install Xposed on rooted devices. You should use Xposed whenever you want to manipulate the behavior of other apps or the core Android system and don't want to go through the hassle of decompiling, recompiling and signing APKs.

First, you create a standard app without an Activity in Android Studio.

Then you have to include the following code in your *build.gradle*:

```
repositories {
    jcenter();
}
```

After that you add the following dependencies:

```
provided 'de.robv.android.xposed:api:82'
provided 'de.robv.android.xposed:api:82:sources'
```

Now you have to place these tags inside the *application* tag found in the *AndroidManifest.xml* so Xposed recognizes your module:

```
<meta-data
       android:name="xposedmodule"
       android:value="true" />
<meta-data
       android:name="xposeddescription"
       android:value="YOUR_MODULE_DESCRIPTION" />
<meta-data
       android:name="xposedminversion"
       android:value="82" />
```

NOTE: Always replace **82** with the latest Xposed version.

Section 136.2: Hooking a method

Create a new class implementing `IXposedHookLoadPackage` and implement the `handleLoadPackage` method:

```
public class MultiPatcher implements IXposedHookLoadPackage
{
    @Override
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
    {

    }
}
```

Inside the method, you check `loadPackageParam.packageName` for the package name of the app you want to hook:

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}
```

Now you can hook your method and either manipulate it before it's code is run, or after:

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }

    XposedHelpers.findAndHookMethod(
        "other.package.name",
        loadPackageParam.classLoader,
        "otherMethodName",
        YourFirstParameter.class,
        YourSecondParameter.class,
        new XC_MethodHook()
    {
        @Override
        protected void beforeHookedMethod(MethodHookParam param) throws Throwable
        {
            Object[] args = param.args;

            args[0] = true;
            args[1] = "example string";
            args[2] = 1;

            Object thisObject = param.thisObject;

            // Do something with the instance of the class
        }

        @Override
        protected void afterHookedMethod(MethodHookParam param) throws Throwable
        {
            Object result = param.getResult();

            param.setResult(result + "example string");
        }
    });
}
```

Chapter 137: PackageManager

Section 137.1: Retrieve application version

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
                            getApplicationContext().getPackageName(),
                            0);

    return info.versionName;
}
```

Section 137.2: Version name and version code

To get `versionName` and `versionCode` of current build of your application you should query Android's package manager.

```
try {
    // Reference to Android's package manager
    PackageManager packageManager = this.getPackageManager();

    // Getting package info of this application
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // Version code
    info.versionCode

    // Version name
    info.versionName

} catch (NameNotFoundException e) {
    // Handle the exception
}
```

Section 137.3: Install time and update time

To get the time at which your app was installed or updated, you should query Android's package manager.

```
try {
    // Reference to Android's package manager
    PackageManager packageManager = this.getPackageManager();

    // Getting package info of this application
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // Install time. Units are as per currentTimeMillis().
    info.firstInstallTime

    // Last update time. Units are as per currentTimeMillis().
    info.lastUpdateTime

} catch (NameNotFoundException e) {
    // Handle the exception
}
```

Section 137.4: Utility method using PackageManager

Here we can find some useful method using PackageManager,

Below method will help to get the app name using package name

```
private String getAppNameFromPackage(String packageName, Context context) {
    Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
    mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
    List<ResolveInfo> pkgAppsList = context.getPackageManager()
            .queryIntentActivities(mainIntent, 0);
    for (ResolveInfo app : pkgAppsList) {
        if (app.activityInfo.packageName.equals(packageName)) {
            return app.activityInfo.loadLabel(context.getPackageManager()).toString();
        }
    }
    return null;
}
```

Below method will help to get the app icon using package name,

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

Below method will help to get the list of installed application.

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {

    List<ApplicationInfo> list =
packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

> Note: above method will give the launcher application too.

Below method will help to hide the app icon from the launcher.

```
public static void hideLockerApp(Context context, boolean hide) {
    ComponentName componentName = new ComponentName(context.getApplicationContext(),
            SplashActivity.class);

    int setting = hide ? PackageManager.COMPONENT_ENABLED_STATE_DISABLED
            : PackageManager.COMPONENT_ENABLED_STATE_ENABLED;

    int current = context.getPackageManager().getComponentEnabledSetting(componentName);

    if (current != setting) {
        context.getPackageManager().setComponentEnabledSetting(componentName, setting,
                PackageManager.DONT_KILL_APP);
```

```
    }
}
```

> Note: After switch off the device and switch on this icon will come back in the launcher.

Chapter 138: Gesture Detection

Section 138.1: Swipe Detection

```
public class OnSwipeListener implements View.OnTouchListener {

    private final GestureDetector gestureDetector;

    public OnSwipeListener(Context context) {
        gestureDetector = new GestureDetector(context, new GestureListener());
    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return gestureDetector.onTouchEvent(event);
    }

    private final class GestureListener extends GestureDetector.SimpleOnGestureListener {

        private static final int SWIPE_VELOCITY_THRESHOLD = 100;
        private static final int SWIPE_THRESHOLD = 100;

        @Override
        public boolean onDown(MotionEvent e) {
            return true;
        }

        @Override
        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
            float diffY = e2.getY() - e1.getY();
            float diffX = e2.getX() - e1.getX();
            if (Math.abs(diffX) > Math.abs(diffY)) {
                if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) >
SWIPE_VELOCITY_THRESHOLD) {
                    if (diffX > 0) {
                        onSwipeRight();
                    } else {
                        onSwipeLeft();
                    }
                }
            } else if (Math.abs(diffY) > SWIPE_THRESHOLD && Math.abs(velocityY) >
SWIPE_VELOCITY_THRESHOLD) {
                if (diffY > 0) {
                    onSwipeBottom();
                } else {
                    onSwipeTop();
                }
            }
            return true;
        }
    }

    public void onSwipeRight() {
    }

    public void onSwipeLeft() {
    }

    public void onSwipeTop() {
    }
```

```
    public void onSwipeBottom() {
    }

}
```

Applied to a view...

```
view.setOnTouchListener(new OnSwipeListener(context) {
    public void onSwipeTop() {
        Log.d("OnSwipeListener", "onSwipeTop");
    }
    public void onSwipeRight() {
        Log.d("OnSwipeListener", "onSwipeRight");
    }
    public void onSwipeLeft() {
        Log.d("OnSwipeListener", "onSwipeLeft");
    }
    public void onSwipeBottom() {
        Log.d("OnSwipeListener", "onSwipeBottom");
    }

});
```

Section 138.2: Basic Gesture Detection

```
public class GestureActivity extends Activity implements
        GestureDetector.OnDoubleTapListener,
        GestureDetector.OnGestureListener {

    private GestureDetector mGestureDetector;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mGestureDetector = new GestureDetector(this, this);
        mGestureDetector.setOnDoubleTapListener(this);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event){
        mGestureDetector.onTouchEvent(event);
        return super.onTouchEvent(event);
    }

    @Override
    public boolean onDown(MotionEvent event) {
        Log.d("GestureDetector","onDown");
        return true;
    }

    @Override
    public boolean onFling(MotionEvent event1, MotionEvent event2, float velocityX, float
velocityY) {
        Log.d("GestureDetector","onFling");
        return true;
    }

    @Override
    public void onLongPress(MotionEvent event) {
```

```
        Log.d("GestureDetector","onLongPress");
    }

    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
        Log.d("GestureDetector","onScroll");
        return true;
    }

    @Override
    public void onShowPress(MotionEvent event) {
        Log.d("GestureDetector","onShowPress");
    }

    @Override
    public boolean onSingleTapUp(MotionEvent event) {
        Log.d("GestureDetector","onSingleTapUp");
        return true;
    }

    @Override
    public boolean onDoubleTap(MotionEvent event) {
        Log.d("GestureDetector","onDoubleTap");
        return true;
    }

    @Override
    public boolean onDoubleTapEvent(MotionEvent event) {
        Log.d("GestureDetector","onDoubleTapEvent");
        return true;
    }

    @Override
    public boolean onSingleTapConfirmed(MotionEvent event) {
        Log.d("GestureDetector","onSingleTapConfirmed");
        return true;
    }

}
```

Chapter 139: Doze Mode

Section 139.1: Whitelisting an Android application programmatically

Whitelisting won't disable the doze mode for your app, but you can do that by using network and hold-wake locks.

Whitelisting an Android application programmatically can be done as follows:

```
boolean isIgnoringBatteryOptimizations = pm.isIgnoringBatteryOptimizations(getPackageName());
if(!isIgnoringBatteryOptimizations){
    Intent intent = new Intent();
    intent.setAction(Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
    intent.setData(Uri.parse("package:" + getPackageName()));
    startActivityForResult(intent, MY_IGNORE_OPTIMIZATION_REQUEST);
}
```

The result of starting the activity above can be verfied by the following code:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == MY_IGNORE_OPTIMIZATION_REQUEST) {
        PowerManager pm = (PowerManager)getSystemService(Context.POWER_SERVICE);
        boolean isIgnoringBatteryOptimizations =
pm.isIgnoringBatteryOptimizations(getPackageName());
        if(isIgnoringBatteryOptimizations){
            // Ignoring battery optimization
        }else{
           // Not ignoring battery optimization
        }
    }
}
```

Section 139.2: Exclude app from using doze mode

1. Open phone's settings
2. open battery
3. open menu and select "battery optimization"
4. from the dropdown menu select "all apps"
5. select the app you want to whitelist
6. select "don't optimize"

Now this app will show under not optimized apps.

An app can check whether it's whitelisted by calling `isIgnoringBatteryOptimizations()`

Chapter 140: Colors

Section 140.1: Color Manipulation

To manipulate colors we will modify the argb (Alpha, Red, Green and Blue) values of a color.

First extract RGB values from your color.

```
int yourColor = Color.parse("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

Now you can reduce or increase red, green, and blue values and combine them to be a color again:

```
int newColor = Color.rgb(red, green, blue);
```

Or if you want to add some alpha to it, you can add it while creating the color:

```
int newColor = Color.argb(alpha, red, green, blue);
```

Alpha and RGB values should be in the range [0-225].

Chapter 141: Keyboard

Section 141.1: Register a callback for keyboard open and close

The idea is to measure a layout before and after each change and if there is a significant change you can be somewhat certain that its the softkeyboard.

```
// A variable to hold the last content layout hight
private int mLastContentHeight = 0;

private ViewTreeObserver.OnGlobalLayoutListener keyboardLayoutListener = new
ViewTreeObserver.OnGlobalLayoutListener() {
    @Override public void onGlobalLayout() {
        int currentContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();

        if (mLastContentHeight > currentContentHeight + 100) {
            Timber.d("onGlobalLayout: Keyboard is open");
            mLastContentHeight = currentContentHeight;
        } else if (currentContentHeight > mLastContentHeight + 100) {
            Timber.d("onGlobalLayout: Keyboard is closed");
            mLastContentHeight = currentContentHeight;
        }
    }
};
```

then in our `onCreate` set the initial value for `mLastContentHeight`

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

and add the listener

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

don't forget to remove the listener on `destroy`

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

Section 141.2: Hide keyboard when user taps anywhere else on the screen

Add code in your **Activity**.

This would work for **Fragment** also, **no need** to add this code in **Fragment**.

```
@Override
public boolean dispatchTouchEvent(MotionEvent ev) {
    View view = getCurrentFocus();
    if (view != null && (ev.getAction() == MotionEvent.ACTION_UP || ev.getAction() ==
MotionEvent.ACTION_MOVE) && view instanceof EditText &&
!view.getClass().getName().startsWith("android.webkit.")) {
        int scrcoords[] = new int[2];
        view.getLocationOnScreen(scrcoords);
        float x = ev.getRawX() + view.getLeft() - scrcoords[0];
        float y = ev.getRawY() + view.getTop() - scrcoords[1];
        if (x < view.getLeft() || x > view.getRight() || y < view.getTop() || y > view.getBottom())
```

```
((InputMethodManager)this.getSystemService(Context.INPUT_METHOD_SERVICE)).hideSoftInputFromWindow((
this.getWindow().getDecorView().getApplicationWindowToken()), 0);
    }
    return super.dispatchTouchEvent(ev);
}
```

Chapter 142: RenderScript

RenderScript is a scripting language that allows you to write high performance graphic rendering and raw computational code. It provides a means of writing performance critical code that the system later compiles to native code for the processor it can run on. This could be the CPU, a multi-core CPU, or even the GPU. Which it ultimately runs on depends on many factors that aren't readily available to the developer, but also depends on what architecture the internal platform compiler supports.

Section 142.1: Getting Started

RenderScript is a framework to allow high performance parallel computation on Android. Scripts you write will be executed across all available processors (e.g. CPU, GPU etc) in parallel allowing you to focus on the task you want to achieve instead of how it is scheduled and executed.

Scripts are written in a C99 based language (C99 being an old version of the C programming language standard). For each Script a Java class is created which allows you to easily interact with RenderScript in your Java code.

Setting up your project

There exist two different ways to access RenderScript in your app, with the Android Framework libraries or the Support Library. Even if you don't want to target devices before API level 11 you should always use the Support Library implementation because it ensures devices compatibility across many different devices. To use the support library implementation you need to use at least build tools version `18.1.0`!

Now lets setup the build.gradle file of your application:

```
android {
    compileSdkVersion 24
    buildToolsVersion '24.0.1'

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 24

        renderscriptTargetApi 18
        renderscriptSupportModeEnabled true
    }
}
```

- `renderscriptTargetApi`: This should be set to the version earliest API level which provides all RenderScript functionality you require.
- `renderscriptSupportModeEnabled`: This enables the use of the Support Library RenderScript implementation.

How RenderScript works

A typical RenderScript consists of two things: Kernels and Functions. A function is just what it sounds like - it accepts an input, does something with that input and returns an output. A Kernel is where the real power of RenderScript comes from.

A Kernel is a function which is executed against every element inside an `Allocation`. An `Allocation` can be used to pass data like a `Bitmap` or a `byte` array to a `RenderScript` and they are also used to get a result from a Kernel. Kernels can either take one `Allocation` as input and another as output or they can modify the data inside just one `Allocation`.

You can write your one Kernels, but there are also many predefined Kernels which you can use to perform common operations like a Gaussian Image Blur.

As already mentioned for every RenderScript file a class is generated to interact with it. These classes always start with the prefix `ScriptC_` followed by the name of the RenderScript file. For example if your RenderScript file is called `example` then the generated Java class will be called `ScriptC_example`. All predefined Scripts just start with the prefix `Script` - for example the Gaussian Image Blur Script is called `ScriptIntrinsicBlur`.

Writing your first RenderScript

The following example is based on an example on GitHub. It performs basic image manipulation by modifying the saturation of an image. You can find the source code **here** and check it out if you want to play around with it yourself. Here's a quick gif of what the result is supposed to look like:

RenderScript Boilerplate

RenderScript files reside in the folder `src/main/rs` in your project. Each file has the file extension `.rs` and has to contain two `#pragma` statements at the top:

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- `#pragma version(1)`: This can be used to set the version of RenderScript you are using. Currently there is only version 1.
- `#pragma rs java_package_name(your.package.name)`: This can be used to set the package name of the Java class generated to interact with this particular RenderScript.

There is another `#pragma` you should usually set in each of your RenderScript files and it is used to set the floating

point precision. You can set the floating point precision to three different levels:

- `#pragma rs_fp_full`: This is the strictest setting with the highest precision and it is also the default value if don't specify anything. You should use this if you require high floating point precision.
- `#pragma rs_fp_relaxed`: This is ensures not quite as high floating point precision, but on some architectures it enables a bunch of optimizations which can cause your scripts to run faster.
- `#pragma rs_fp_imprecise`: This ensures even less precision and should be used if floating point precision does not really matter to your script.

Most scripts can just use `#pragma rs_fp_relaxed` unless you really need high floating point precision.

Global Variables

Now just like in C code you can define global variables or constants:

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

The variable `gMonoMult` is of type `float3`. This means it is a vector consisting of 3 float numbers. The other **`float`** variable called `saturationValue` is not constant, therefore you can set it at runtime to a value you like. You can use variables like this in your Kernels or functions and therefore they are another way to give input to or receive output from your RenderScripts. For each not constant variable a getter and setter method will be generated on the associated Java class.

Kernels

But now lets get started implementing the Kernel. For the purposes of this example I am not going to explain the math used in the Kernel to modify the saturation of the image, but instead will focus on how to implement a Kernel and and how to use it. At the end of this chapter I will quickly explain what the code in this Kernel is actually doing.

Kernels in general

Let's take a look at the source code first:

```
uchar4 __attribute__((kernel)) saturation(uchar4 in) {
    float4 f4 = rsUnpackColor8888(in);
    float3 dotVector = dot(f4.rgb, gMonoMult);
    float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
    return rsPackColorTo8888(newColor);
}
```

As you can see it looks like a normal C function with one exception: The `__attribute__((kernel))` between the return type and method name. This is what tells RenderScript that this method is a Kernel. Another thing you might notice is that this method accepts a `uchar4` parameter and returns another `uchar4` value. `uchar4` is - like the `float3` variable we discussed in the chapter before - a vector. It contains 4 `uchar` values which are just byte values in the range from 0 to 255.

You can access these individual values in many different ways, for example `in.r` would return the byte which corresponds to the red channel of a pixel. We use a `uchar4` since each pixel is made up of 4 values - `r` for red, `g` for green, `b` for blue and `a` for alpha - and you can access them with this shorthand. RenderScript also allows you to take any number of values from a vector and create another vector with them. For example `in.rgb` would return a `uchar3` value which just contains the red, green and blue parts of the pixel without the alpha value.

At runtime RenderScript will call this Kernel method for each pixel of an image which is why the return value and

parameter are just one `uchar4` value. RenderScript will run many of these calls in parallel on all available processors which is why RenderScript is so powerful. This also means that you don't have to worry about threading or thread safety, you can just implement whatever you want to do to each pixel and RenderScript takes care of the rest.

When calling a Kernel in Java you supply two `Allocation` variables, one which contains the input data and another one which will receive the output. Your Kernel method will be called for each value in the input `Allocation` and will write the result to the output `Allocation`.

RenderScript Runtime API methods

In the Kernel above a few methods are used which are provided out of the box. RenderScript provides many such methods and they are vital for almost anything you are going to do with RenderScript. Among them are methods to do math operations like `sin()` and helper methods like `mix()` which mixes two values according to another values. But there are also methods for more complex operations when dealing with vectors, quaternions and matrices.

The official **RenderScript Runtime API Reference** is the best resource out there if you want to know more about a particular method or are looking for a specific method which performs a common operation like calculating the dot product of a matrix. You can find this documentation **here**.

Kernel Implementation

Now let's take a look at the specifics of what this Kernel is doing. Here's the first line in the Kernel:

```
float4 f4 = rsUnpackColor8888(in);
```

The first line calls the built in method `rsUnpackColor8888()` which transforms the `uchar4` value to a `float4` values. Each color channel is also transformed to the range `0.0f - 1.0f` where `0.0f` corresponds to a byte value of `0` and `1.0f` to `255`. The main purpose of this is to make all the math in this Kernel a lot simpler.

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

This next line uses the built in method `dot()` to calculate the dot product of two vectors. `gMonoMult` is a constant value we defined a few chapters above. Since both vectors need to be of the same length to calculate the dot product and also since we just want to affect the color channels and not the alpha channel of a pixel we use the shorthand `.rgb` to get a new `float3` vector which just contains the red, green and blue color channels. Those of us who still remember from school how the dot product works will quickly notice that the dot product should return just one value and not a vector. Yet in the code above we are assigning the result to a `float3` vector. This is again a feature of RenderScript. When you assign a one dimensional number to a vector all elements in the vector will be set to this value. For example the following snippet will assign `2.0f` to each of the three values in the `float3` vector:

```
float3 example = 2.0f;
```

So the result of the dot product above is assigned to each element in the `float3` vector above.

Now comes the part in which we actually use the global variable `saturationLevel` to modify the saturation of the image:

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

This uses the built in method `mix()` to mix together the original color with the dot product vector we created above. How they are mixed together is determined by the global `saturationLevel` variable. So a `saturationLevel` of `0.0f` will cause the resulting color to have no part of the original color values and will only consist of values in the `dotVector` which results in a black and white or grayed out image. A value of `1.0f` will cause the resulting color to

be completely made up of the original color values and values above `1.0f` will multiply the original colors to make them more bright and intense.

```
return rsPackColorTo8888(newColor);
```

This is the last part in the Kernel. `rsPackColorTo8888()` transforms the `float3` vector back to a `uchar4` value which is then returned. The resulting byte values are clamped to a range between 0 and 255, so float values higher than `1.0f` will result in a byte value of 255 and values lower than `0.0` will result in a byte value of `0`.

And that is the whole Kernel implementation. Now there is only one part remaining: How to call a Kernel in Java.

Calling RenderScript in Java
Basics

As was already mentioned above for each RenderScript file a Java class is generated which allows you to interact with the your scripts. These files have the prefix `ScriptC_` followed by the name of the RenderScript file. To create an instance of these classes you first need an instance of the `RenderScript` class:

```
final RenderScript renderScript = RenderScript.create(context);
```

The static method `create()` can be used to create a `RenderScript` instance from a `Context`. You can then instantiate the Java class which was generated for your script. If you called the RenderScript file `saturation.rs` then the class will be called `ScriptC_saturation`:

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

On this class you can now set the saturation level and call the Kernel. The setter which was generated for the `saturationLevel` variable will have the prefix `set_` followed by the name of the variable:

```
script.set_saturationLevel(1.0f);
```

There is also a getter prefixed with `get_` which allows you to get the saturation level currently set:

```
float saturationLevel = script.get_saturationLevel();
```

Kernels you define in your RenderScript are prefixed with `forEach_` followed by the name of the Kernel method. The Kernel we have written expects an input `Allocation` and an output `Allocation` as its parameters:

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

The input `Allocation` needs to contain the input image, and after the `forEach_saturation` method has finished the output allocation will contain the modified image data.

Once you have an `Allocation` instance you can copy data from and to those `Allocations` by using the methods `copyFrom()` and `copyTo()`. For example you can copy a new image into your input `Allocation by calling:

```
inputAllocation.copyFrom(inputBitmap);
```

The same way you can retrieve the result image by calling `copyTo()` on the output `Allocation`:

```
outputAllocation.copyTo(outputBitmap);
```

Creating Allocation instances

There are many ways to create an `Allocation`. Once you have an `Allocation` instance you can copy new data from and to those `Allocations` with `copyTo()` and `copyFrom()` like explained above, but to create them initially you have to know with what kind of data you are exactly working with. Let's start with the input `Allocation`:

We can use the static method `createFromBitmap()` to quickly create our input `Allocation` from a `Bitmap`:

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

In this example the input image never changes so we never need to modify the input `Allocation` again. We can reuse it each time the `saturationLevel` changes to create a new output `Bitmap`.

Creating the output `Allocation` is a little more complex. First we need to create what's called a `Type`. A `Type` is used to tell an `Allocation` with what kind of data it's dealing with. Usually one uses the `Type.Builder` class to quickly create an appropriate `Type`. Let's take a look at the code first:

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();
```

We are working with a normal 32 bit (or in other words 4 byte) per pixel `Bitmap` with 4 color channels. That's why we are choosing `Element.RGBA_8888` to create the `Type`. Then we use the methods `setX()` and `setY()` to set the width and height of the output image to the same size as the input image. The method `create()` then creates the `Type` with the parameters we specified.

Once we have the correct `Type` we can create the output `Allocation` with the static method `createTyped()`:

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

Now we are almost done. We also need an output `Bitmap` in which we can copy the data from the output `Allocation`. To do this we use the static method `createBitmap()` to create a new empty `Bitmap` with the same size and configuration as the input `Bitmap`.

```
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);
```

And with that we have all the puzzle pieces to execute our RenderScript.

Full example

Now let's put all this together in one example:

```
// Create the RenderScript instance
final RenderScript renderScript = RenderScript.create(context);

// Create the input Allocation
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, inputBitmap);

// Create the output Type.
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();
```

```
// And use the Type to create am output Allocation
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);

// Create an empty output Bitmap from the input Bitmap
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);

// Create an instance of our script
final ScriptC_saturation script = new ScriptC_saturation(renderScript);

// Set the saturation level
script.set_saturationLevel(2.0f);

// Execute the Kernel
script.forEach_saturation(inputAllocation, outputAllocation);

// Copy the result data to the output Bitmap
outputAllocation.copyTo(outputBitmap);

// Display the result Bitmap somewhere
someImageView.setImageBitmap(outputBitmap);
```

Conclusion

With this introduction you should be all set to write your own RenderScript Kernels for simple image manipulation. However there are a few things you have to keep in mind:

- **RenderScript only works in Application projects**: Currently RenderScript files cannot be part of a library project.
- **Watch out for memory**: RenderScript is very fast, but it can also be memory intensive. There should never be more than one instance of `RenderScript` at any time. You should also reuse as much as possible. Normally you just need to create your `Allocation` instances once and can reuse them in the future. The same goes for output `Bitmaps` or your script instances. Reuse as much as possible.
- **Do your work in the background**: Again RenderScript is very fast, but not instant in any way. Any Kernel, especially complex ones should be executed off the UI thread in an `AsyncTask` or something similar. However for the most part you don't have to worry about memory leaks. All RenderScript related classes only use the application `Context` and therefore don't cause memory leaks. But you still have to worry about the usual stuff like leaking `View`, `Activity` or any `Context` instance which you use yourself!
- **Use built in stuff**: There are many predefined scripts which perform tasks like image blurring, blending, converting, resizing. And there are many more built in methods which help you implement your kernels. Chances are that if you want to do something there is either a script or method which already does what you are trying to do. Don't reinvent the wheel.

If you want to quickly get started and play around with actual code I recommend you take a look at the example GitHub project which implements the exact example talked about in this tutorial. You can find the project **here**. Have fun with RenderScript!

Section 142.2: Blur a View

BlurBitmapTask.java

```
public class BlurBitmapTask extends AsyncTask<Bitmap, Void, Bitmap> {
    private final WeakReference<ImageView> imageViewReference;
    private final RenderScript renderScript;
```

```
    private boolean shouldRecycleSource = false;

    public BlurBitmapTask(@NonNull Context context, @NonNull ImageView imageView) {
        // Use a WeakReference to ensure
        // the ImageView can be garbage collected
        imageViewReference = new WeakReference<>(imageView);
        renderScript = RenderScript.create(context);
    }

    // Decode image in background.
    @Override
    protected Bitmap doInBackground(Bitmap... params) {
        Bitmap bitmap = params[0];
        return blurBitmap(bitmap);
    }

    // Once complete, see if ImageView is still around and set bitmap.
    @Override
    protected void onPostExecute(Bitmap bitmap) {
        if (bitmap == null || isCancelled()) {
            return;
        }

        final ImageView imageView = imageViewReference.get();
        if (imageView == null) {
            return;
        }

        imageView.setImageBitmap(bitmap);
    }

    public Bitmap blurBitmap(Bitmap bitmap) {
        // https://plus.google.com/+MarioViviani/posts/fhuzYkji9zz

        //Let's create an empty bitmap with the same size of the bitmap we want to blur
        Bitmap outBitmap = Bitmap.createBitmap(bitmap.getWidth(), bitmap.getHeight(),
                Bitmap.Config.ARGB_8888);

        //Instantiate a new Renderscript


        //Create an Intrinsic Blur Script using the Renderscript
        ScriptIntrinsicBlur blurScript = ScriptIntrinsicBlur.create(renderScript,
Element.U8_4(renderScript));

        //Create the in/out Allocations with the Renderscript and the in/out bitmaps
        Allocation allIn = Allocation.createFromBitmap(renderScript, bitmap);
        Allocation allOut = Allocation.createFromBitmap(renderScript, outBitmap);

        //Set the radius of the blur
        blurScript.setRadius(25.f);

        //Perform the Renderscript
        blurScript.setInput(allIn);
        blurScript.forEach(allOut);

        //Copy the final bitmap created by the out Allocation to the outBitmap
        allOut.copyTo(outBitmap);

        // recycle the original bitmap
        // nope, we are using the original bitmap as well :/
        if (shouldRecycleSource) {
```

```
            bitmap.recycle();
        }

        //After finishing everything, we destroy the Renderscript.
        renderScript.destroy();

        return outBitmap;
    }

    public boolean isShouldRecycleSource() {
        return shouldRecycleSource;
    }

    public void setShouldRecycleSource(boolean shouldRecycleSource) {
        this.shouldRecycleSource = shouldRecycleSource;
    }
}
```

Usage:

```
ImageView imageViewOverlayOnViewToBeBlurred
        .setImageDrawable(ContextCompat.getDrawable(this, android.R.color.transparent));
View viewToBeBlurred.setDrawingCacheQuality(View.DRAWING_CACHE_QUALITY_LOW);
viewToBeBlurred.setDrawingCacheEnabled(true);
BlurBitmapTask blurBitmapTask = new BlurBitmapTask(this, imageViewOverlayOnViewToBeBlurred);
blurBitmapTask.execute(Bitmap.createBitmap(viewToBeBlurred.getDrawingCache()));
viewToBeBlurred.setDrawingCacheEnabled(false);
```

Section 142.3: Blur an image

This example demonstrates how to use Renderscript API to blur an image (using Bitmap). This example uses ScriptInstrinsicBlur provided by android Renderscript API (API >= 17).

```
public class BlurProcessor {

    private RenderScript rs;
    private Allocation inAllocation;
    private Allocation outAllocation;
    private int width;
    private int height;

    private ScriptIntrinsicBlur blurScript;

    public BlurProcessor(RenderScript rs) {
        this.rs = rs;
    }

    public void initialize(int width, int height) {
        blurScript = ScriptIntrinsicBlur.create(rs, Element.U8_4(rs));
        blurScript.setRadius(7f); // Set blur radius. 25 is max

        if (outAllocation != null) {
            outAllocation.destroy();
            outAllocation = null;
        }

        // Bitmap must have ARGB_8888 config for this type
        Type bitmapType = new Type.Builder(rs, Element.RGBA_8888(rs))
            .setX(width)
            .setY(height)
            .setMipmaps(false) // We are using MipmapControl.MIPMAP_NONE
```

```
            .create();

        // Create output allocation
        outAllocation = Allocation.createTyped(rs, bitmapType);

        // Create input allocation with same type as output allocation
        inAllocation = Allocation.createTyped(rs, bitmapType);
    }

    public void release() {

        if (blurScript != null) {
            blurScript.destroy();
            blurScript = null;
        }

        if (inAllocation != null) {
            inAllocation.destroy();
            inAllocation = null;
        }

        if (outAllocation != null) {
            outAllocation.destroy();
            outAllocation = null;
        }
    }

    public Bitmap process(Bitmap bitmap, boolean createNewBitmap) {
        if (bitmap.getWidth() != width || bitmap.getHeight() != height) {
            // Throw error if required
            return null;
        }

        // Copy data from bitmap to input allocations
        inAllocation.copyFrom(bitmap);

        // Set input for blur script
        blurScript.setInput(inAllocation);

        // process and set data to the output allocation
        blurScript.forEach(outAllocation);

        if (createNewBitmap) {
            Bitmap returnVal = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
            outAllocation.copyTo(returnVal);
            return returnVal;
        }

        outAllocation.copyTo(bitmap);
        return bitmap;
    }
}
```

Each script has a kernel which processes the data and it is generally invoked via **`forEach`** method.

```
public class BlurActivity extends AppCompatActivity {
    private BlurProcessor blurProcessor;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // setup layout and other stuff
```

```
        blurProcessor = new BlurProcessor(Renderscript.create(getApplicationContext()));
    }

    private void loadImage(String path) {
        // Load image to bitmap
        Bitmap bitmap = loadBitmapFromPath(path);

        // Initialize processor for this bitmap
        blurProcessor.release();
        blurProcessor.initialize(bitmap.getWidth(), bitmap.getHeight());

        // Blur image
        Bitmap blurImage = blurProcessor.process(bitmap, true); // Use newBitamp as false if you
don't want to create a new bitmap
    }
}
```

This concluded the example here. It is advised to do the processing in a background thread.

Chapter 143: Fresco

Fresco is a powerful system for displaying images in Android applications.

In Android 4.x and lower, Fresco puts images in a special region of **Android memory** (called ashmem). This lets your application run faster - and suffer the dreaded OutOfMemoryError much less often.

Fresco also supports streaming of JPEGs.

Section 143.1: Getting Started with Fresco

First, add Fresco to your `build.gradle` as shown in the Remarks section:

If you need additional features, like animated GIF or WebP support, you have to add the corresponding Fresco artifacts as well.

Fresco needs to be initialized. You should only do this 1 time, so placing the initialization in your `Application` is a good idea. An example for this would be:

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        Fresco.initialize(this);
    }
}
```

If you want to load remote images from a server, your app needs the internt permission. Simply add it to your `AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Then, add a `SimpleDraweeView` to your XML layout. Fresco does not support `wrap_content` for image dimensions since you might have multiple images with different dimensions (placeholder image, error image, actual image, ...).

So you can either add a `SimpleDraweeView` with fixed dimensions (or `match_parent`):

```
<com.facebook.drawee.view.SimpleDraweeView
   android:id="@+id/my_image_view"
   android:layout_width="120dp"
   android:layout_height="120dp"
   fresco:placeholderImage="@drawable/placeholder" />
```

Or supply an *aspect ratio* for your image:

```
<com.facebook.drawee.view.SimpleDraweeView
    android:id="@+id/my_image_view"
    android:layout_width="120dp"
    android:layout_height="wrap_content"
    fresco:viewAspectRatio="1.33"
    fresco:placeholderImage="@drawable/placeholder" />
```

Finally, you can set your image URI in Java:

```
SimpleDraweeView draweeView = (SimpleDraweeView) findViewById(R.id.my_image_view);
```

```
draweeView.setImageURI("http://yourdomain.com/yourimage.jpg");
```

That's it! You should see your placeholder drawable until the network image has been fetched.

Section 143.2: Using OkHttp 3 with Fresco

First, in addition to the normal Fresco Gradle dependency, you have to add the OkHttp 3 dependency to your `build.gradle`:

```
compile "com.facebook.fresco:imagepipeline-okhttp3:1.2.0" // Or a newer version.
```

When you initialize Fresco (usually in your custom `Application` implementation), you can now specify your OkHttp client:

```
OkHttpClient okHttpClient = new OkHttpClient(); // Build on your own OkHttpClient.

Context context = ... // Your Application context.
ImagePipelineConfig config = OkHttpImagePipelineConfigFactory
        .newBuilder(context, okHttpClient)
        .build();
Fresco.initialize(context, config);
```

Section 143.3: JPEG Streaming with Fresco using DraweeController

This example assumes that you have already added Fresco to your app (see this example):

```
SimpleDraweeView img = new SimpleDraweeView(context);
ImageRequest request = ImageRequestBuilder
        .newBuilderWithSource(Uri.parse("http://example.com/image.png"))
        .setProgressiveRenderingEnabled(true) // This is where the magic happens.
        .build();

DraweeController controller = Fresco.newDraweeControllerBuilder()
        .setImageRequest(request)
        .setOldController(img.getController()) // Get the current controller from our
SimpleDraweeView.
        .build();

img.setController(controller); // Set the new controller to the SimpleDraweeView to enable
progressive JPEGs.
```

Chapter 144: Swipe to Refresh

Section 144.1: How to add Swipe-to-Refresh To your app

Make sure the following dependency is added to your app's `build.gradle` file under dependencies:

```
compile 'com.android.support:support-core-ui:24.2.0'
```

Then add the `SwipeRefreshLayout` in your layout:

```
<android.support.v4.widget.SwipeRefreshLayout
       android:id="@+id/swipe_refresh_layout"
       android:layout_width="match_parent"
       android:layout_height="wrap_content">

     <!-- place your view here -->

</android.support.v4.widget.SwipeRefreshLayout>
```

Finally implement the `SwipeRefreshLayout.OnRefreshListener` listener.

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {
    @Override
    public void onRefresh() {
         // your code
    }
});
```

Section 144.2: Swipe To Refresh with RecyclerView

To add a **Swipe To Refresh** layout with a **RecyclerView** add the following to your Activity/Fragment layout file:

```
<android.support.v4.widget.SwipeRefreshLayout
   android:id="@+id/refresh_layout"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <android.support.v7.widget.RecyclerView
       android:id="@+id/recycler_view"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:orientation="vertical"
       android:scrollbars="vertical" />

</android.support.v4.widget.SwipeRefreshLayout>
```

In your Activity/Fragment add the following to initialize the **SwipeToRefreshLayout**:

```
    SwipeRefreshLayout mSwipeRefreshLayout = (SwipeRefreshLayout)
findViewById(R.id.refresh_layout);
    mSwipeRefreshLayout.setColorSchemeResources(R.color.green_bg,
            android.R.color.holo_green_light,
            android.R.color.holo_orange_light,
            android.R.color.holo_red_light);
```

```
    mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {
            // Execute code when refresh layout swiped
        }
    });
```

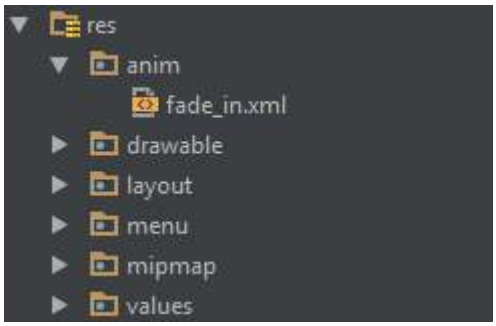
Chapter 145: Creating Splash screen

Section 145.1: Splash screen with animation

This example shows a simple but effective splash screen with animation that can be created by using Android Studio.

Step 1: Create an animation

Create a new directory named *anim* in the *res* directory. Right-click it and create a new *Animation Resource* file named *fade_in.xml*:



Then, put the following code into the *fade_in.xml* file:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:fillAfter="true" >
    <alpha
       android:duration="1000"
       android:fromAlpha="0.0"
       android:interpolator="@android:anim/accelerate_interpolator"
       android:toAlpha="1.0" />
</set>
```

Step 2: Create an activity

Create an *empty activity* using Android Studio named `Splash`. Then, put the following code into it:

```
public class Splash extends AppCompatActivity {
    Animation anim;
    ImageView imageView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        imageView=(ImageView)findViewById(R.id.imageView2); // Declare an imageView to show the
animation.
        anim = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.fade_in); // Create the
animation.
        anim.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
            }

            @Override
            public void onAnimationEnd(Animation animation) {
                startActivity(new Intent(this,HomeActivity.class));
                // HomeActivity.class is the activity to go after showing the splash screen.
            }
```

```
            @Override
            public void onAnimationRepeat(Animation animation) {
            }
        });
        imageView.startAnimation(anim);
    }
}
```

Next, put the following code into the layout file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_splash"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="your_packagename"
   android:orientation="vertical"
   android:background="@android:color/white">
    <ImageView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:id="@+id/imageView2"
       android:layout_weight="1"
       android:src="@drawable/Your_logo_or_image" />
</LinearLayout>
```

Step 3: Replace the default launcher

Turn your `Splash` activity into a launcher by adding the following code to the *AndroidManifest* file:

```
<activity
   android:name=".Splash"
   android:theme="@style/AppTheme.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Then, remove the default launcher activity by removing the following code from the *AndroidManifest* file:

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Section 145.2: A basic splash screen

A splash screen is just like any other activity, but it can handle all of your startup-needs in the background. Example:

Manifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.package"
    android:versionCode="1"
    android:versionName="1.0" >

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".Splash"
            android:label="@string/app_name"
             >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Now our splash-screen will be called as the first activity.

Here is an example splashscreen that also handles some critical app elements:

```
public class Splash extends Activity{

    public final int SPLASH_DISPLAY_LENGTH = 3000;

    private void checkPermission() {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.WAKE_LOCK) !=
PackageManager.PERMISSION_GRANTED ||
                ContextCompat.checkSelfPermission(this,Manifest.permission.INTERNET) !=
PackageManager.PERMISSION_GRANTED ||
                ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_NETWORK_STATE)
!= PackageManager.PERMISSION_GRANTED) {//Can add more as per requirement


            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.WAKE_LOCK,
                            Manifest.permission.INTERNET,
                            Manifest.permission.ACCESS_NETWORK_STATE},
                    123);
        }

    }
    @Override
    protected void onCreate(Bundle sis){
        super.onCreate(sis);
        //set the content view. The XML file can contain nothing but an image, such as a logo or the
app icon
        setContentView(R.layout.splash);


        //we want to display the splash screen for a few seconds before it automatically
```

```
        //disappears and loads the game. So we create a thread:
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {

                //request permissions. NOTE: Copying this and the manifest will cause the app to
crash as the permissions requested aren't defined in the manifest.
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M ) {
                        checkPermission();
                    }
                    String lang = [load or determine the system language and set to default if it
isn't available.]
                    Locale locale = new Locale(lang);
                    Locale.setDefault(locale);
                    Configuration config = new Configuration    ();
                    config.locale = locale;
                    Splash.this.getResources().updateConfiguration(config,
                            Splash.this.getResources().getDisplayMetrics())    ;

                    //after three seconds, it will execute all of this code.
                    //as such, we then want to redirect to the master-activity
                    Intent mainIntent = new Intent(Splash.this, MainActivity.class);
                    Splash.this.startActivity(mainIntent);

                //then we finish this class. Dispose of it as it is longer needed
                Splash.this.finish();
            }
        }, SPLASH_DISPLAY_LENGTH);

    }

    public void onPause(){
        super.onPause();
        finish();
    }

}
```

Chapter 146: IntentService

Section 146.1: Creating an IntentService

To create an IntentService, create a class which extends `IntentService`, and within it, a method which overrides `onHandleIntent`:

```
package com.example.myapp;
public class MyIntentService extends IntentService {
    @Override
     protected void onHandleIntent (Intent workIntent) {
         //Do something in the background, based on the contents of workIntent.
     }
}
```

Section 146.2: Basic IntentService Example

The abstract class `IntentService` is a base class for services, which run in the background without any user interface. Therefore, in order to update the UI, we have to make use of a receiver, which may be either a `BroadcastReceiver` or a `ResultReceiver`:

- A `BroadcastReceiver` should be used if your service needs to communicate with multiple components that want to listen for communication.
- A `ResultReceiver`: should be used if your service needs to communicate with only the parent application (i.e. your application).

Within the `IntentService`, we have one key method, `onHandleIntent()`, in which we will do all actions, for example, preparing notifications, creating alarms, etc.

If you want to use you own `IntentService`, you have to extend it as follows:

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
        super("YourIntentService ");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // TODO: Write your own code here.
    }
}
```

Calling/starting the activity can be done as follows:

```
Intent i = new Intent(this, YourIntentService.class);
startService(i);  // For the service.
startActivity(i); // For the activity; ignore this for now.
```

Similar to any activity, you can pass extra information such as bundle data to it as follows:

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo","bar");
startService(passDataIntent);
```

Now assume that we passed some data to the `YourIntentService` class. Based on this data, an action can be

performed as follows:

```
public class YourIntentService extends IntentService {
    private String actvityValue="bar";
    String retrivedValue=intent.getStringExtra("foo");

    public YourIntentService () {
        super("YourIntentService ");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        if(retrivedValue.equals(actvityValue)){
            // Send the notification to foo.
        } else {
            // Retrieving data failed.
        }
    }
}
```

The code above also shows how to handle constraints in the `OnHandleIntent()` method.

Section 146.3: Sample Intent Service

Here is an example of an `IntentService` that pretends to load images in the background. All you need to do to implement an `IntentService` is to provide a constructor that calls the **`super`**`(String)` constructor, and you need to implement the `onHandleIntent(Intent)` method.

```
public class ImageLoaderIntentService extends IntentService {

    public static final String IMAGE_URL = "url";

    /**
     * Define a constructor and call the super(String) constructor, in order to name the worker
     * thread - this is important if you want to debug and know the name of the thread upon
     * which this Service is operating its jobs.
     */
    public ImageLoaderIntentService() {
        super("Example");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // This is where you do all your logic - this code is executed on a background thread

        String imageUrl = intent.getStringExtra(IMAGE_URL);

        if (!TextUtils.isEmpty(imageUrl)) {
            Drawable image = HttpUtils.loadImage(imageUrl); // HttpUtils is made-up for the example
        }

        // Send your drawable back to the UI now, so that you can use it - there are many ways
        // to achieve this, but they are out of reach for this example
    }
}
```

In order to start an `IntentService`, you need to send an `Intent` to it. You can do so from an `Activity`, for an example. Of course, you're not limited to that. Here is an example of how you would summon your new `Service` from an `Activity` class.

```
Intent serviceIntent = new Intent(this, ImageLoaderIntentService.class); // you can use 'this' as
the first parameter if your class is a Context (i.e. an Activity, another Service, etc.), otherwise,
supply the context differently
serviceIntent.putExtra(IMAGE_URL, "http://www.example-site.org/some/path/to/an/image");
startService(serviceIntent); // if you are not using 'this' in the first line, you also have to put
the call to the Context object before startService(Intent) here
```

The `IntentService` processes the data from its `Intents` sequentially, so that you can send multiple `Intents` without worrying whether they will collide with each other. Only one `Intent` at a time is processed, the rest go in a queue. When all the jobs are complete, the `IntentService` will shut itself down automatically.

Chapter 147: Implicit Intents

Parameters	Details
o	Intent
action	`String`: The Intent action, such as `ACTION_VIEW`.
uri	`Uri`: The Intent data URI.
packageContext	`Context`: A Context of the application package implementing this class.
cls	`Class`: The component class that is to be used for the intent.

Section 147.1: Implicit and Explicit Intents

An explicit intent is used for starting an activity or service within the same application package. In this case the name of the intended class is explicitly mentioned:

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

However, an implicit intent is sent across the system for any application installed on the user's device that can handle that intent. This is used to share information between different applications.

```
Intent intent = new Intent("com.stackoverflow.example.VIEW");

//We need to check to see if there is an application installed that can handle this intent
if (getPackageManager().resolveActivity(intent, 0) != null){
    startActivity(intent);
}else{
    //Handle error
}
```

More details on the differences can be found in the Android Developer docs here: Intent Resolution

Section 147.2: Implicit Intents

Implicit intents do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.

For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

Example:

```
// Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

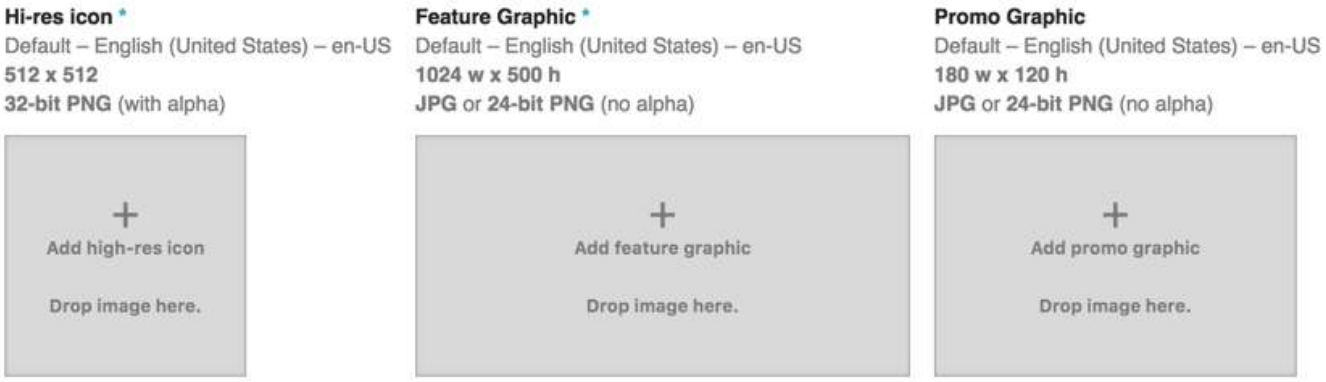
Chapter 148: Publish to Play Store

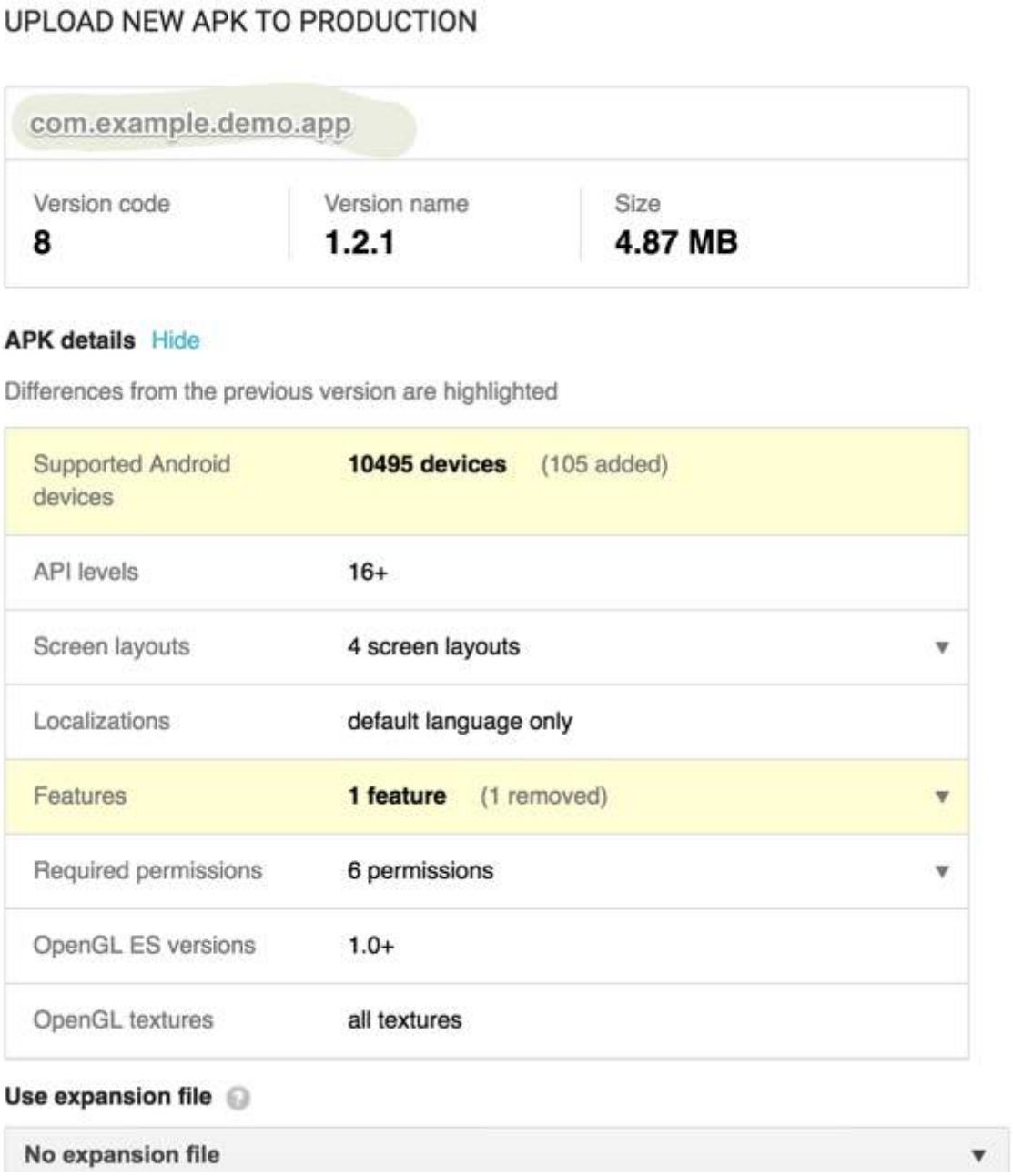
Section 148.1: Minimal app submission guide

Requirements:

- A developer account
- An apk already built and signed with a non-debug key
- A free app that doesn't have in-app billing
- no Firebase Cloud Messaging or Game Services

1. Head to https://play.google.com/apps/publish/
   1a) Create your developer account if you do not have one
2. Click button `Create new Application`
3. Click submit APK
4. Fill in all required fields in the form, including some assets that will be displayed on the Play Store (see image below)
5. When satisfied hit `Publish app` button





See more about signing in Configure Signing Settings

Chapter 149: Universal Image Loader

Section 149.1: Basic usage

1. Load an image, decode it into a bitmap, and display the bitmap in an `ImageView` (or any other view which implements the `ImageAware` interface):

```
ImageLoader.getInstance().displayImage(imageUri, imageView);
```

2. Load an image, decode it into a bitmap, and return the bitmap to a callback:

```
ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {
    @Override
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {
        // Do whatever you want with the bitmap.
    }
});
```

3. Load an image, decode it into a bitmap and return the bitmap synchronously:

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

Section 149.2: Initialize Universal Image Loader

1. Add the following dependency to the *build.gradle* file:

```
compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
```

2. Add the following permissions to the *AndroidManifest.xml* file:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

3. Initialize the Universal Image Loader. This must be done before the first usage:

```
ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)
    // ...
    .build();
ImageLoader.getInstance().init(config);
```

The full configuration options can be found here.

Chapter 150: Image Compression

Section 150.1: How to compress image without size change

Get **Compressed Bitmap** from Singleton class:

```
ImageView imageView = (ImageView)findViewById(R.id.imageView);
Bitmap bitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
imageView.setImageBitmap(bitmap);
```

ImageUtils.java:

```
public class ImageUtils {

    public static ImageUtils mInstant;

    public static ImageUtils getInstant(){
        if(mInstant==null){
            mInstant = new ImageUtils();
        }
        return mInstant;
    }

    public  Bitmap getCompressedBitmap(String imagePath) {
        float maxHeight = 1920.0f;
        float maxWidth = 1080.0f;
        Bitmap scaledBitmap = null;
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        Bitmap bmp = BitmapFactory.decodeFile(imagePath, options);

        int actualHeight = options.outHeight;
        int actualWidth = options.outWidth;
        float imgRatio = (float) actualWidth / (float) actualHeight;
        float maxRatio = maxWidth / maxHeight;

        if (actualHeight > maxHeight || actualWidth > maxWidth) {
            if (imgRatio < maxRatio) {
                imgRatio = maxHeight / actualHeight;
                actualWidth = (int) (imgRatio * actualWidth);
                actualHeight = (int) maxHeight;
            } else if (imgRatio > maxRatio) {
                imgRatio = maxWidth / actualWidth;
                actualHeight = (int) (imgRatio * actualHeight);
                actualWidth = (int) maxWidth;
            } else {
                actualHeight = (int) maxHeight;
                actualWidth = (int) maxWidth;

            }
        }

        options.inSampleSize = calculateInSampleSize(options, actualWidth, actualHeight);
        options.inJustDecodeBounds = false;
        options.inDither = false;
        options.inPurgeable = true;
        options.inInputShareable = true;
        options.inTempStorage = new byte[16 * 1024];
```

```
        try {
            bmp = BitmapFactory.decodeFile(imagePath, options);
        } catch (OutOfMemoryError exception) {
            exception.printStackTrace();

        }
        try {
            scaledBitmap = Bitmap.createBitmap(actualWidth, actualHeight, Bitmap.Config.ARGB_8888);
        } catch (OutOfMemoryError exception) {
            exception.printStackTrace();
        }

        float ratioX = actualWidth / (float) options.outWidth;
        float ratioY = actualHeight / (float) options.outHeight;
        float middleX = actualWidth / 2.0f;
        float middleY = actualHeight / 2.0f;

        Matrix scaleMatrix = new Matrix();
        scaleMatrix.setScale(ratioX, ratioY, middleX, middleY);

        Canvas canvas = new Canvas(scaledBitmap);
        canvas.setMatrix(scaleMatrix);
        canvas.drawBitmap(bmp, middleX - bmp.getWidth() / 2, middleY - bmp.getHeight() / 2, new
Paint(Paint.FILTER_BITMAP_FLAG));

        ExifInterface exif = null;
        try {
            exif = new ExifInterface(imagePath);
            int orientation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION, 0);
            Matrix matrix = new Matrix();
            if (orientation == 6) {
                matrix.postRotate(90);
            } else if (orientation == 3) {
                matrix.postRotate(180);
            } else if (orientation == 8) {
                matrix.postRotate(270);
            }
            scaledBitmap = Bitmap.createBitmap(scaledBitmap, 0, 0, scaledBitmap.getWidth(),
scaledBitmap.getHeight(), matrix, true);
        } catch (IOException e) {
            e.printStackTrace();
        }
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        scaledBitmap.compress(Bitmap.CompressFormat.JPEG, 85, out);

        byte[] byteArray = out.toByteArray();

        Bitmap updatedBitmap = BitmapFactory.decodeByteArray(byteArray, 0, byteArray.length);

        return updatedBitmap;
    }

    private int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int reqHeight) {
        final int height = options.outHeight;
        final int width = options.outWidth;
        int inSampleSize = 1;

        if (height > reqHeight || width > reqWidth) {
            final int heightRatio = Math.round((float) height / (float) reqHeight);
            final int widthRatio = Math.round((float) width / (float) reqWidth);
            inSampleSize = heightRatio < widthRatio ? heightRatio : widthRatio;
        }
```

```
        final float totalPixels = width * height;
        final float totalReqPixelsCap = reqWidth * reqHeight * 2;

        while (totalPixels / (inSampleSize * inSampleSize) > totalReqPixelsCap) {
            inSampleSize++;
        }
        return inSampleSize;
    }
}
```

Dimensions are same after compressing Bitmap.

How did I checked ?

```
Bitmap beforeBitmap = BitmapFactory.decodeFile("Your_Image_Path_Here");
Log.i("Before Compress Dimension", beforeBitmap.getWidth()+"-"+beforeBitmap.getHeight());

Bitmap afterBitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
Log.i("After Compress Dimension", afterBitmap.getWidth() + "-" + afterBitmap.getHeight());
```

Output:

```
Before Compress : Dimension: 1080-1452
After Compress : Dimension: 1080-1452
```

Chapter 151: 9-Patch Images

Section 151.1: Basic rounded corners

The key to correctly stretching is in the top and left border.

The top border controls horizontal stretching and the left border controls vertical stretching.

This example creates rounded corners suitable for a Toast.

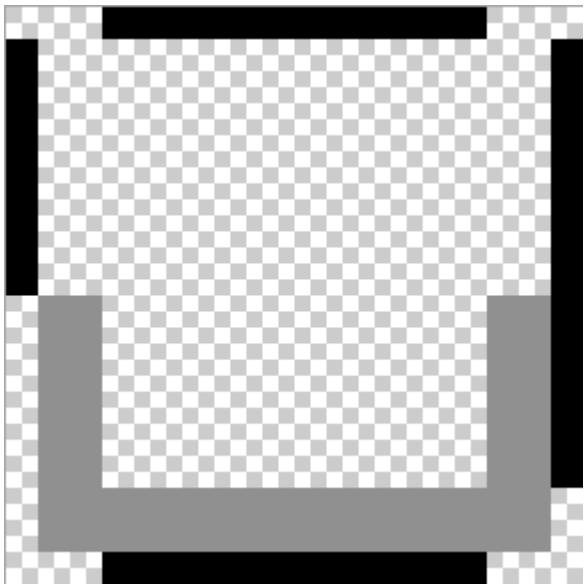
The parts of the image that are below the *top border* and to the right of the *left border* will expand to fill all unused space.

This example will stretch to all combinations of sizes, as shown below:

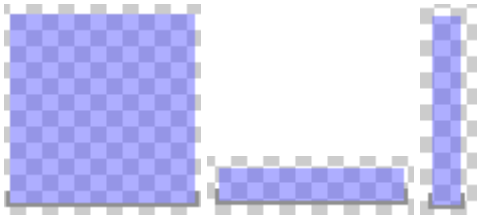
Section 151.2: Optional padding lines

Nine-patch images allow optional definition of the padding lines in the image. The padding lines are the lines on the right and at the bottom.

If a View sets the 9-patch image as its background, the padding lines are used to define the space for the View's content (e.g. the text input in an `EditText`). If the padding lines are not defined, the left and top lines are used instead.



The content area of the stretched image then looks like this:



Section 151.3: Basic spinner

The `Spinner` can be reskinned according to your own style requirements using a Nine Patch.

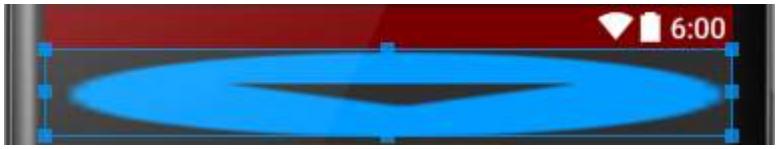
As an example, see this Nine Patch:

As you can see, it has 3 extremely small areas of stretching marked.

The top border has only left of the icon marked. That indicates that I want the left side (complete transparency) of the drawable to fill the `Spinner` view until the icon is reached.

The left border has marked transparent segments at the top and bottom of the icon marked. That indicates that both the top and the bottom will expand to the size of the `Spinner` view. This will leave the icon itself centered vertically.

Using the image without Nine Patch metadata:



Using the image with Nine Patch metadata:

Chapter 152: Email Validation

Section 152.1: Email address validation

Add the following method to check whether an email address is valid or not:

```
private boolean isValidEmailId(String email){
  return Pattern.compile("^(([\\w-]+\\.)+[\\w-]+|([a-zA-Z]{1}|[\\w-]{2,}))@"
              + "((([0-1]?[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\.([0-1]?"
              + "[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\."
              + "([0-1]?[0-9]{1,2}|25[0-5]|2[0-4][0-9])\\.([0-1]?"
              + "[0-9]{1,2}|25[0-5]|2[0-4][0-9])){1}|"
              + "([a-zA-Z]+[\\w-]+\\.)+[a-zA-Z]{2,4})$").matcher(email).matches();
}
```

The above method can easily be verified by converting the text of an `EditText` widget into a `String`:

```
if(isValidEmailId(edtEmailId.getText().toString().trim())){
  Toast.makeText(getApplicationContext(), "Valid Email Address.", Toast.LENGTH_SHORT).show();
}else{
  Toast.makeText(getApplicationContext(), "InValid Email Address.", Toast.LENGTH_SHORT).show();
}
```

Section 152.2: Email Address validation with using Patterns

```
  if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){
       Log.i("EmailCheck","It is valid");
  }
```

Chapter 153: Bottom Sheets

A bottom sheet is a sheet that slides up from the bottom edge of the screen.

Section 153.1: Quick Setup

Make sure the following dependency is added to your app's build.gradle file under dependencies:

```
compile 'com.android.support:design:25.3.1'
```

Then you can use the Bottom sheet using these options:

- `BottomSheetBehavior` to be used with `CoordinatorLayout`
- `BottomSheetDialog` which is a dialog with a bottom sheet behavior
- `BottomSheetDialogFragment` which is an extension of `DialogFragment`, that creates a `BottomSheetDialog` instead of a standard dialog.

Section 153.2: BottomSheetBehavior like Google maps

`Version ≥ 2.1.x`

This example depends on Support Library 23.4.0.+.

BottomSheetBehavior is characterized by :

1. Two toolbars with animations that respond to the bottom sheet movements.
2. A FAB that hides when it is near to the "modal toolbar" (the one that appears when you are sliding up).
3. A backdrop image behind bottom sheet with some kind of parallax effect.
4. A Title (TextView) in Toolbar that appears when bottom sheet reach it.
5. The notification satus bar can turn its background to transparent or full color.
6. A custom bottom sheet behavior with an "anchor" state.

Now let's check them one by one:

ToolBars
When you open that view in Google Maps, you can see a toolbar in where you can search, it's the only one that I'm not doing exactly like Google Maps, because I wanted to do it more generic. Anyway that `ToolBar` is inside an `AppBarLayout` and it got hidden when you start dragging the BottomSheet and it appears again when the BottomSheet reach the `COLLAPSED` state.
To achieve it you need to:

- create a `Behavior` and extend it from `AppBarLayout.ScrollingViewBehavior`
- override `layoutDependsOn` and `onDependentViewChanged` methods. Doing it you will listen for bottomSheet movements.
- create some methods to hide and unhide the AppBarLayout/ToolBar with animations.

This is how I did it for first toolbar or ActionBar:

```
@Override
```

```
public boolean layoutDependsOn(CoordinatorLayout parent, View child, View dependency) {
    return dependency instanceof NestedScrollView;
}

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                      View dependency) {

    if (mChild == null) {
        initValues(child, dependency);
        return false;
    }

    float dVerticalScroll = dependency.getY() - mPreviousY;
    mPreviousY = dependency.getY();

    //going up
    if (dVerticalScroll <= 0 && !hidden) {
        dismissAppBar(child);
        return true;
    }

    return false;
}

private void initValues(final View child, View dependency) {

    mChild = child;
    mInitialY = child.getY();

    BottomSheetBehaviorGoogleMapsLike bottomSheetBehavior =
BottomSheetBehaviorGoogleMapsLike.from(dependency);
    bottomSheetBehavior.addBottomSheetCallback(new
BottomSheetBehaviorGoogleMapsLike.BottomSheetCallback() {
        @Override
        public void onStateChanged(@NonNull View bottomSheet,
@BottomSheetBehaviorGoogleMapsLike.State int newState) {
            if (newState == BottomSheetBehaviorGoogleMapsLike.STATE_COLLAPSED ||
                    newState == BottomSheetBehaviorGoogleMapsLike.STATE_HIDDEN)
                showAppBar(child);
        }

        @Override
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {

        }
    });
}

private void dismissAppBar(View child){
    hidden = true;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_shor
tAnimTime));
    mToolbarAnimation.y(-(mChild.getHeight()+25)).start();
}

private void showAppBar(View child) {
    hidden = false;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
```

```
appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_medi
umAnimTime));
    mToolbarAnimation.y(mInitialY).start();
}
```

Here is the complete file if you need it

The second Toolbar or "Modal" toolbar:
You have to override the same methods, but in this one you have to take care of more behaviors:

- show/hide the ToolBar with animations
- change status bar color/background
- show/hide the BottomSheet title in the ToolBar
- close the bottomSheet or send it to collapsed state

The code for this one is a little extensive, so I will let the link

The FAB

This is a Custom Behavior too, but extends from `FloatingActionButton.Behavior`. In `onDependentViewChanged` you have to look when it reach the "offSet" or point in where you want to hide it. In my case I want to hide it when it's near to the second toolbar, so I dig into FAB parent (a CoordinatorLayout) looking for the AppBarLayout that contains the ToolBar, then I use the ToolBar position like `OffSet`:

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, FloatingActionButton child, View
dependency) {

    if (offset == 0)
        setOffsetValue(parent);

    if (dependency.getY() <=0)
        return false;

    if (child.getY() <= (offset + child.getHeight()) && child.getVisibility() == View.VISIBLE)
        child.hide();
    else if (child.getY() > offset && child.getVisibility() != View.VISIBLE)
        child.show();

    return false;
}
```

Complete Custom FAB Behavior link

The image behind the BottomSheet with parallax effect:
Like the others, it's a custom behavior, the only "complicated" thing in this one is the little algorithm that keeps the image anchored to the BottomSheet and avoid the image collapse like default parallax effect:

```
@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                      View dependency) {

    if (mYmultiplier == 0) {
        initValues(child, dependency);
```

```
        return true;
    }

    float dVerticalScroll = dependency.getY() - mPreviousY;
    mPreviousY = dependency.getY();

    //going up
    if (dVerticalScroll <= 0 && child.getY() <= 0) {
        child.setY(0);
        return true;
    }

    //going down
    if (dVerticalScroll >= 0 && dependency.getY() <= mImageHeight)
        return false;

    child.setY( (int)(child.getY() + (dVerticalScroll * mYmultiplier) ) );

    return true;
}
```

The complete file for backdrop image with parallax effect

Now for the end: **The Custom BottomSheet Behavior**
To achieve the 3 steps, first you need to understand that default BottomSheetBehavior has 5 states: `STATE_DRAGGING`, `STATE_SETTLING`, `STATE_EXPANDED`, `STATE_COLLAPSED`, `STATE_HIDDEN` and for the Google Maps behavior you need to add a middle state between collapsed and expanded: `STATE_ANCHOR_POINT`.
I tried to extend the default bottomSheetBehavior with no success, so I just copy pasted all the code and modified what I need.
To achieve what I'm talking about follow the next steps:

1. Create a Java class and extend it from `CoordinatorLayout.Behavior<V>`
2. Copy paste code from default `BottomSheetBehavior` file to your new one.
3. Modify the method `clampViewPositionVertical` with the following code:

```
@Override
public int clampViewPositionVertical(View child, int top, int dy) {
    return constrain(top, mMinOffset, mHideable ? mParentHeight : mMaxOffset);
}
int constrain(int amount, int low, int high) {
    return amount < low ? low : (amount > high ? high : amount);
}
```

4. Add a new state

   public static final int STATE_ANCHOR_POINT = X;

5. Modify the next methods: `onLayoutChild`, `onStopNestedScroll`, `BottomSheetBehavior<V> from(V view)` and `setState` (optional)

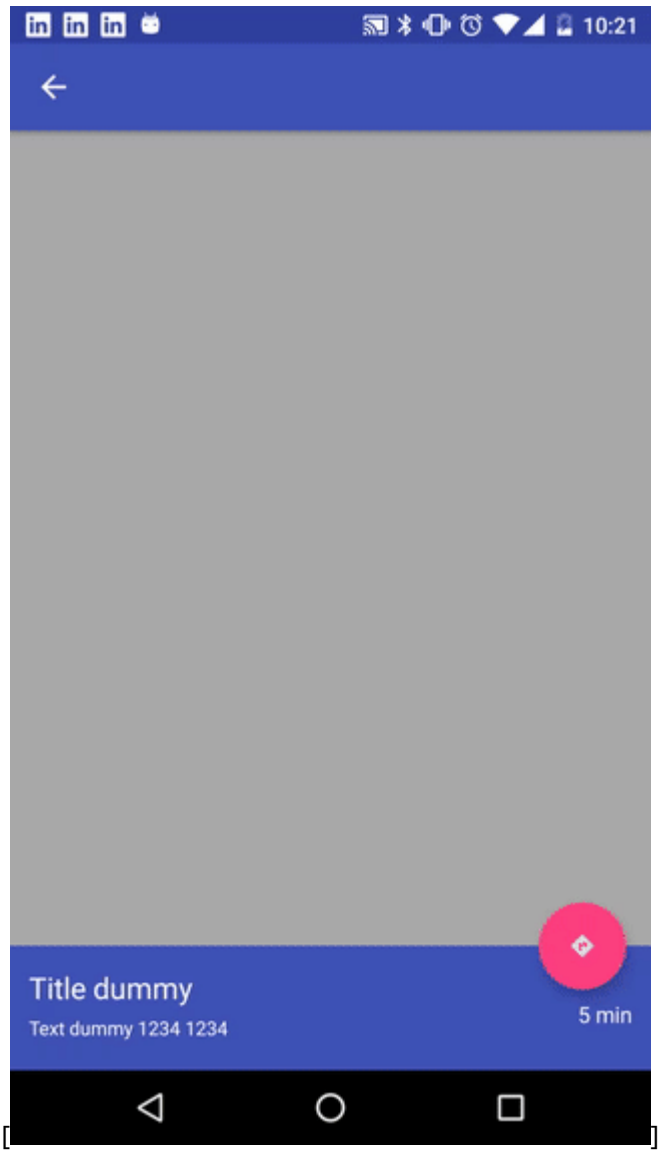
```
public boolean onLayoutChild(CoordinatorLayout parent, V child, int layoutDirection) {
    // First let the parent lay it out
    if (mState != STATE_DRAGGING && mState != STATE_SETTLING) {
        if (ViewCompat.getFitsSystemWindows(parent) &&
                !ViewCompat.getFitsSystemWindows(child)) {
            ViewCompat.setFitsSystemWindows(child, true);
        }
        parent.onLayoutChild(child, layoutDirection);
    }
    // Offset the bottom sheet
    mParentHeight = parent.getHeight();
    mMinOffset = Math.max(0, mParentHeight - child.getHeight());
    mMaxOffset = Math.max(mParentHeight - mPeekHeight, mMinOffset);

    //if (mState == STATE_EXPANDED) {
    //    ViewCompat.offsetTopAndBottom(child, mMinOffset);
    //} else if (mHideable && mState == STATE_HIDDEN...
    if (mState == STATE_ANCHOR_POINT) {
        ViewCompat.offsetTopAndBottom(child, mAnchorPoint);
    } else if (mState == STATE_EXPANDED) {
        ViewCompat.offsetTopAndBottom(child, mMinOffset);
    } else if (mHideable && mState == STATE_HIDDEN) {
        ViewCompat.offsetTopAndBottom(child, mParentHeight);
    } else if (mState == STATE_COLLAPSED) {
        ViewCompat.offsetTopAndBottom(child, mMaxOffset);
    }
    if (mViewDragHelper == null) {
        mViewDragHelper = ViewDragHelper.create(parent, mDragCallback);
    }
    mViewRef = new WeakReference<>(child);
    mNestedScrollingChildRef = new WeakReference<>(findScrollingChild(child));
    return true;
}


public void onStopNestedScroll(CoordinatorLayout coordinatorLayout, V child, View target) {
    if (child.getTop() == mMinOffset) {
        setStateInternal(STATE_EXPANDED);
        return;
    }
    if (target != mNestedScrollingChildRef.get() || !mNestedScrolled) {
        return;
    }
    int top;
    int targetState;
    if (mLastNestedScrollDy > 0) {
        //top = mMinOffset;
        //targetState = STATE_EXPANDED;
        int currentTop = child.getTop();
        if (currentTop > mAnchorPoint) {
            top = mAnchorPoint;
            targetState = STATE_ANCHOR_POINT;
        }
        else {
            top = mMinOffset;
            targetState = STATE_EXPANDED;
        }
    } else if (mHideable && shouldHide(child, getYVelocity())) {
        top = mParentHeight;
        targetState = STATE_HIDDEN;
    } else if (mLastNestedScrollDy == 0) {
        int currentTop = child.getTop();
```

```
        if (Math.abs(currentTop - mMinOffset) < Math.abs(currentTop - mMaxOffset)) {
            top = mMinOffset;
            targetState = STATE_EXPANDED;
        } else {
            top = mMaxOffset;
            targetState = STATE_COLLAPSED;
        }
    } else {
        //top = mMaxOffset;
        //targetState = STATE_COLLAPSED;
        int currentTop = child.getTop();
        if (currentTop > mAnchorPoint) {
            top = mMaxOffset;
            targetState = STATE_COLLAPSED;
        }
        else {
            top = mAnchorPoint;
            targetState = STATE_ANCHOR_POINT;
        }
    }
    if (mViewDragHelper.smoothSlideViewTo(child, child.getLeft(), top)) {
        setStateInternal(STATE_SETTLING);
        ViewCompat.postOnAnimation(child, new SettleRunnable(child, targetState));
    } else {
        setStateInternal(targetState);
    }
    mNestedScrolled = false;
}

public final void setState(@State int state) {
    if (state == mState) {
        return;
    }
    if (mViewRef == null) {
        // The view is not laid out yet; modify mState and let onLayoutChild handle it later
        /**
         * New behavior (added: state == STATE_ANCHOR_POINT ||)
         */
        if (state == STATE_COLLAPSED || state == STATE_EXPANDED ||
                state == STATE_ANCHOR_POINT ||
                (mHideable && state == STATE_HIDDEN)) {
            mState = state;
        }
        return;
    }
    V child = mViewRef.get();
    if (child == null) {
        return;
    }
    int top;
    if (state == STATE_COLLAPSED) {
        top = mMaxOffset;
    } else if (state == STATE_ANCHOR_POINT) {
        top = mAnchorPoint;
    } else if (state == STATE_EXPANDED) {
        top = mMinOffset;
    } else if (mHideable && state == STATE_HIDDEN) {
        top = mParentHeight;
    } else {
        throw new IllegalArgumentException("Illegal state argument: " + state);
    }
    setStateInternal(STATE_SETTLING);
```

```
    if (mViewDragHelper.smoothSlideViewTo(child, child.getLeft(), top)) {
        ViewCompat.postOnAnimation(child, new SettleRunnable(child, state));
    }
}


public static <V extends View> BottomSheetBehaviorGoogleMapsLike<V> from(V view) {
    ViewGroup.LayoutParams params = view.getLayoutParams();
    if (!(params instanceof CoordinatorLayout.LayoutParams)) {
        throw new IllegalArgumentException("The view is not a child of CoordinatorLayout");
    }
    CoordinatorLayout.Behavior behavior = ((CoordinatorLayout.LayoutParams) params)
            .getBehavior();
    if (!(behavior instanceof BottomSheetBehaviorGoogleMapsLike)) {
        throw new IllegalArgumentException(
                "The view is not associated with BottomSheetBehaviorGoogleMapsLike");
    }
    return (BottomSheetBehaviorGoogleMapsLike<V>) behavior;
}
```

Link to the whole project where you can see all the Custom Behaviors

And here it is how it looks like:



Section 153.3: Modal bottom sheets with BottomSheetDialog

The `BottomSheetDialog` is a dialog styled as a bottom sheet

Just use:

```
//Create a new BottomSheetDialog
BottomSheetDialog dialog = new BottomSheetDialog(context);
//Inflate the layout R.layout.my_dialog_layout
dialog.setContentView(R.layout.my_dialog_layout);
//Show the dialog
dialog.show();
```

In this case you don't need to attach a BottomSheet behavior.

Section 153.4: Modal bottom sheets with BottomSheetDialogFragment

You can realize a modal bottom sheets using a `BottomSheetDialogFragment`.

The `BottomSheetDialogFragment` is a modal bottom sheet.
This is a version of `DialogFragment` that shows a bottom sheet using `BottomSheetDialog` instead of a floating dialog.

Just define the fragment:

```
public class MyBottomSheetDialogFragment extends BottomSheetDialogFragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        return inflater.inflate(R.layout.my_fragment_bottom_sheet, container);
    }
}
```

Then use this code to show the fragment:

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

This Fragment will create a `BottomSheetDialog`.

Section 153.5: Persistent Bottom Sheets

You can achieve a Persistent Bottom Sheet attaching a `BottomSheetBehavior` to a child View of a `CoordinatorLayout`:

```
<android.support.design.widget.CoordinatorLayout >

    <!-- .....   -->

    <LinearLayout
      android:id="@+id/bottom_sheet"
      android:elevation="4dp"
      android:minHeight="120dp"
```

```
      app:behavior_peekHeight="120dp"
      ...
      app:layout_behavior="android.support.design.widget.BottomSheetBehavior">

          <!-- .....   -->

       </LinearLayout>

</android.support.design.widget.CoordinatorLayout>
```

Then in your code you can create a reference using:

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

You can set the state of your BottomSheetBehavior using the setState() method:

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

You can use one of these states:

- STATE_COLLAPSED: this collapsed state is the default and shows just a portion of the layout along the bottom. The height can be controlled with the `app:behavior_peekHeight` attribute (defaults to 0)
- STATE_EXPANDED: the fully expanded state of the bottom sheet, where either the whole bottom sheet is visible (if its height is less than the containing `CoordinatorLayout`) or the entire `CoordinatorLayout` is filled
- STATE_HIDDEN: disabled by default (and enabled with the `app:behavior_hideable` attribute), enabling this allows users to swipe down on the bottom sheet to completely hide the bottom sheet

If you'd like to receive callbacks of state changes, you can add a `BottomSheetCallback`:

```
mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
      // React to state change
    }
      @Override
      public void onSlide(@NonNull View bottomSheet, float slideOffset) {
       // React to dragging events
   }
 });
```

Section 153.6: Open BottomSheet DialogFragment in Expanded mode by default

BottomSheet DialogFragment opens up in `STATE_COLLAPSED` by default. Which can be forced to open to `STATE_EXPANDED`and take up the full device screen with help of the following code template.

@NonNull @Override public Dialog onCreateDialog(Bundle savedInstanceState) {

```
    BottomSheetDialog dialog = (BottomSheetDialog) super.onCreateDialog(savedInstanceState);

    dialog.setOnShowListener(new DialogInterface.OnShowListener() {
        @Override
        public void onShow(DialogInterface dialog) {
```

```
            BottomSheetDialog d = (BottomSheetDialog) dialog;

            FrameLayout bottomSheet = (FrameLayout)
d.findViewById(android.support.design.R.id.design_bottom_sheet);
            BottomSheetBehavior.from(bottomSheet).setState(BottomSheetBehavior.STATE_EXPANDED);
        }
    });

    // Do something with your dialog like setContentView() or whatever
    return dialog;
}
```

Although dialog animation is slightly noticeable but does the task of opening the DialogFragment in full screen very well.

Chapter 154: EditText

Section 154.1: Working with EditTexts

The EditText is the standard text entry widget in Android apps. If the user needs to enter text into an app, this is the primary way for them to do that.

EditText

There are many important properties that can be set to customize the behavior of an EditText. Several of these are listed below. Check out the official text fields guide for even more input field details.

Usage

An EditText is added to a layout with all default behaviors with the following XML:

```
<EditText
   android:id="@+id/et_simple"
   android:layout_height="wrap_content"
   android:layout_width="match_parent">
</EditText>
```

Note that an EditText is simply a thin extension of the TextView and inherits all of the same properties.

Retrieving the Value

Getting the value of the text entered into an EditText is as follows:

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);
String strValue = simpleEditText.getText().toString();
```

Further Entry Customization

We might want to limit the entry to a single-line of text (avoid newlines):

```
<EditText
 android:singleLine="true"
 android:lines="1"
/>
```

You can limit the characters that can be entered into a field using the digits attribute:

```
<EditText
  android:inputType="number"
  android:digits="01"
/>
```

This would restrict the digits entered to just "0" and "1". We might want to limit the total number of characters with:

```
<EditText
  android:maxLength="5"
/>
```

Using these properties we can define the expected input behavior for text fields.

Adjusting Colors

You can adjust the highlight background color of selected text within an EditText with the `android:textColorHighlight` property:

```
<EditText
   android:textColorHighlight="#7cff88"
/>
```

Displaying Placeholder Hints

You may want to set the hint for the EditText control to prompt a user for specific input with:

```
<EditText
    ...
    android:hint="@string/my_hint">
</EditText>
```

Hints

Changing the bottom line color

Assuming you are using the AppCompat library, you can override the styles colorControlNormal, colorControlActivated, and colorControlHighlight:

```
<style name="Theme.App.Base" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorControlNormal">#d32f2f</item>
    <item name="colorControlActivated">#ff5722</item>
    <item name="colorControlHighlight">#f44336</item>
</style>
```

If you do not see these styles applied within a DialogFragment, there is a known bug when using the LayoutInflater passed into the onCreateView() method.

The issue has already been fixed in the AppCompat v23 library. See this guide about how to upgrade. Another temporary workaround is to use the Activity's layout inflater instead of the one passed into the onCreateView() method:

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment, container);
  }
```

Listening for EditText Input

Check out the basic event listeners cliffnotes for a look at how to listen for changes to an EditText and perform an action when those changes occur.

Displaying Floating Label Feedback

Traditionally, the EditText hides the hint message (explained above) after the user starts typing. In addition, any validation error messages had to be managed manually by the developer.

With the `TextInputLayout` you can setup a floating label to display hints and error messages. You can find more details here.

Section 154.2: Customizing the InputType

Text fields can have different input types, such as number, date, password, or email address. The type determines what kind of characters are allowed inside the field, and may prompt the virtual keyboard to optimize its layout for frequently used characters.

By default, any text contents within an `EditText` control is displayed as plain text. By setting the `inputType` attribute, we can facilitate input of different types of information, like phone numbers and passwords:

```
<EditText
    ...
    android:inputType="phone">
</EditText>
```

Most common input types include:

Type	Description
textUri	Text that will be used as a URI
textEmailAddress	Text that will be used as an e-mail address
textPersonName	Text that is the name of a person
textPassword	Text that is a password that should be obscured
number	A numeric only field
phone	For entering a phone number
date	For entering a date
time	For entering a time
textMultiLine	Allow multiple lines of text in the field

The `android:inputType` also allows you to specify certain keyboard behaviors, such as whether to capitalize all new words or use features like auto-complete and spelling suggestions.
Here are some of the common input type values that define keyboard behaviors:

Type	Description
textCapSentences	Normal text keyboard that capitalizes the first letter for each new sentence
textCapWords	Normal text keyboard that capitalizes every word. Good for titles or person names
textAutoCorrect	Normal text keyboard that corrects commonly misspelled words

You can set multiple `inputType` attributes if needed (separated by '|').
Example:

```
<EditText
    android:id="@+id/postal_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/postal_address_hint"
    android:inputType="textPostalAddress|
                       textCapWords|
                       textNoSuggestions" />
```

You can see a list of all available input types here.

Section 154.3: Icon or button inside Custom Edit Text and its

action and click listeners

This example will help to have the Edit text with the icon at the right side.

> Note: In this just I am using setCompoundDrawablesWithIntrinsicBounds, So if you want to change the icon position you can achieve that using setCompoundDrawablesWithIntrinsicBounds in setIcon.

```
public class MKEditText extends AppCompatEditText {

    public interface IconClickListener {
        public void onClick();
    }

    private IconClickListener mIconClickListener;

    private static final String TAG = MKEditText.class.getSimpleName();

    private final int EXTRA_TOUCH_AREA = 50;
    private Drawable mDrawable;
    private boolean touchDown;

    public MKEditText(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public MKEditText(Context context) {
        super(context);
    }

    public MKEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public void showRightIcon() {
        mDrawable = ContextCompat.getDrawable(getContext(), R.drawable.ic_android_black_24dp);

        setIcon();
    }

    public void setIconClickListener(IconClickListener iconClickListener) {
        mIconClickListener = iconClickListener;
    }

    private void setIcon() {
        Drawable[] drawables = getCompoundDrawables();

        setCompoundDrawablesWithIntrinsicBounds(drawables[0], drawables[1], mDrawable,
drawables[3]);

        setInputType(InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
        setSelection(getText().length());
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        final int right = getRight();
        final int drawableSize = getCompoundPaddingRight();
        final int x = (int) event.getX();
        switch (event.getAction()) {
```

```
            case MotionEvent.ACTION_DOWN:
                if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right + EXTRA_TOUCH_AREA)
{
                    touchDown = true;
                    return true;
                }
                break;
            case MotionEvent.ACTION_UP:
                if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right + EXTRA_TOUCH_AREA
&& touchDown) {
                    touchDown = false;
                    if (mIconClickListener != null) {
                        mIconClickListener.onClick();
                    }
                    return true;
                }
                touchDown = false;
                break;

        }
        return super.onTouchEvent(event);
    }
}
```

> If you want to change the touch area you can change the EXTRA_TOUCH_AREA values default I gave as 50.

And for Enable the button and click listener you can call from your Activity or Fragment like this,

```
MKEditText mkEditText = (MKEditText) findViewById(R.id.password);
mkEditText.showRightIcon();
mkEditText.setIconClickListener(new MKEditText.IconClickListener() {
            @Override
            public void onClick() {
              // You can do action here for the icon.

            }
        });
```

Section 154.4: Hiding SoftKeyboard

Hiding Softkeyboard is a **basic requirement** usually when working with EditText. The softkeyboard by *default* can only be closed by pressing back button and so most developers use InputMethodManager to force Android to hide the virtual keyboard calling hideSoftInputFromWindow and passing in the token of the window containing your focused view. The code to do the following:

```
public void hideSoftKeyboard()
{
        InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
}
```

The code is direct, but another major problems that arises is that the hide function needs to be called when some event occurs. What to do when you need the Softkeyboard hidden upon pressing anywhere other than your EditText? The following code gives a neat function that needs to be called in your onCreate() method just once.

```
public void setupUI(View view)
{
        String s = "inside";
        //Set up touch listener for non-text box views to hide keyboard.
        if (!(view instanceof EditText)) {

            view.setOnTouchListener(new View.OnTouchListener() {

                public boolean onTouch(View v, MotionEvent event) {
                    hideSoftKeyboard();
                    return false;
                }

            });
        }

        //If a layout container, iterate over children and seed recursion.
        if (view instanceof ViewGroup) {

            for (int i = 0; i < ((ViewGroup) view).getChildCount(); i++) {

                View innerView = ((ViewGroup) view).getChildAt(i);

                setupUI(innerView);
            }
        }
}
```

Section 154.5: `inputype` attribute

`inputype` attribute in `EditText` widget: *(tested on Android 4.4.3 and 2.3.3)*

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. Case: lowercase. Suggestion: yes. Add. chars: **,** and **.** and everything

textFilter= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. Case: lowercase. **Suggestion: no**. Add. chars: **,** and **.** and everything

textCapWords= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. **Case: Camel Case**. Suggestion: yes. Add. chars: **,** and **.** and everything

textCapSentences= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. **Case: Sentence case**. Suggestion: yes. Add. chars: **,** and **.** and everything

time= Keyboard: numeric. Enter button: Send/Next. Emotion: no. Case: -. **Suggestion: no**. Add. chars: :

textMultiLine= Keyboard: alphabet/default. **Enter button: nextline**. Emotion: yes. Case: lowercase. Suggestion: yes. Add. chars: **,** and **.** and everything

number= Keyboard: numeric. Enter button: Send/Next. Emotion: no. Case: -. Suggestion: no. **Add. chars: nothing**

textEmailAddress= Keyboard: alphabet/default. Enter button: Send/Next. **Emotion: no**. Case: lowercase. **Suggestion: no**. Add. chars: **@** and **.** and everything

(No type)= Keyboard: alphabet/default. **Enter button: nextline**. Emotion: yes. Case: lowercase. Suggestion: yes.

Add. chars: **,** and **.** and everything

textPassword= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: no. Case: lowercase. **Suggestion: no**. Add. chars: **,** and **.** and everything

text= Keyboard: Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. Case: lowercase. Suggestion: yes. Add. chars: **,** and **.** and everything

textShortMessage= Keyboard: alphabet/default. **Enter button: emotion**. Emotion: yes. Case: lowercase. Suggestion: yes. Add. chars: **,** and **.** and everything

textUri= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: no. Case: lowercase. **Suggestion: no**. Add. chars: **/** and **.** and everything

textCapCharacters= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. **Case: UPPERCASE**. Suggestion: yes. Add. chars: **,** and **.** and everything

phone= Keyboard: numeric. Enter button: Send/Next. Emotion: no. Case: -. **Suggestion: no**. Add. chars: *** # . - / () W P N , +**

textPersonName= Keyboard: alphabet/default. Enter button: Send/Next. Emotion: yes. Case: lowercase. Suggestion: yes. Add. chars: **,** and **.** and everything

Note: `Auto-capitalization` setting will change the default behavior.

Note 2: In the `Numeric keyboard`, ALL numbers are English 1234567890.

Note 3: `Correction/Suggestion` setting will change the default behavior.

Chapter 155: Speech to Text Conversion

Section 155.1: Speech to Text With Default Google Prompt Dialog

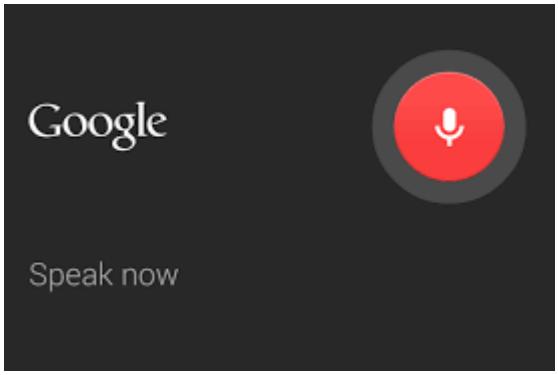
Trigger speech to text translation

```
private void startListening() {

     //Intent to listen to user vocal input and return result in same activity
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

   //Use a language model based on free-form speech recognition.
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());

    //Message to display in dialog box
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
            getString(R.string.speech_to_text_info));
    try {
        startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);
    } catch (ActivityNotFoundException a) {
        Toast.makeText(getApplicationContext(),
                getString(R.string.speech_not_supported),
                Toast.LENGTH_SHORT).show();
    }
}
```

Get translated results in onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    switch (requestCode) {
    case REQ_CODE_SPEECH_INPUT: {
        if (resultCode == RESULT_OK && null != data) {

            ArrayList<String> result = data
                    .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            txtSpeechInput.setText(result.get(0));
        }
        break;
    }

    }
}
```

Output



Section 155.2: Speech to Text without Dialog

The following code can be used to trigger speech-to-text translation without showing a dialog:

```
public void startListeningWithoutDialog() {
    // Intent to listen to user vocal input and return the result to the same activity.
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

    // Use a language model based on free-form speech recognition.
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 5);
    intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
            appContext.getPackageName());

    // Add custom listeners.
    CustomRecognitionListener listener = new CustomRecognitionListener();
    SpeechRecognizer sr = SpeechRecognizer.createSpeechRecognizer(appContext);
    sr.setRecognitionListener(listener);
    sr.startListening(intent);
}
```

The custom listener class `CustomRecognitionListener` used in the code above is implemented as follows:

```
class CustomRecognitionListener implements RecognitionListener {
    private static final String TAG = "RecognitionListener";

    public void onReadyForSpeech(Bundle params) {
        Log.d(TAG, "onReadyForSpeech");
    }

    public void onBeginningOfSpeech() {
        Log.d(TAG, "onBeginningOfSpeech");
    }

    public void onRmsChanged(float rmsdB) {
        Log.d(TAG, "onRmsChanged");
    }

    public void onBufferReceived(byte[] buffer) {
        Log.d(TAG, "onBufferReceived");
    }

    public void onEndOfSpeech() {
        Log.d(TAG, "onEndofSpeech");
    }
```

```
    public void onError(int error) {
        Log.e(TAG, "error " + error);

        conversionCallaback.onErrorOccured(TranslatorUtil.getErrorText(error));
    }

    public void onResults(Bundle results) {
        ArrayList<String> result = data
                .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
        txtSpeechInput.setText(result.get(0));
    }

    public void onPartialResults(Bundle partialResults) {
        Log.d(TAG, "onPartialResults");
    }

    public void onEvent(int eventType, Bundle params) {
        Log.d(TAG, "onEvent " + eventType);
    }
}
```

Chapter 156: Installing apps with ADB

Section 156.1: Uninstall an app

Write the following command in your terminal to uninstall an app with a provided package name:

```
adb uninstall <packagename>
```

Section 156.2: Install all apk file in directory

Windows :

```
for %f in (C:\your_app_path\*.apk) do adb install "%f"
```

Linux :

```
for f in *.apk ; do adb install "$f" ; done
```

Section 156.3: Install an app

Write the following command in your terminal:

```
adb install [-rtsdg] <file>
```

Note that you have to pass a file that is on your computer and not on your device.

If you append `-r` at the end, then any existing conflicting apks will be overwritten. Otherwise, the command will quit with an error.

`-g` will immediately grant all runtime permissions.

`-d` allows version code downgrade (only appliable on debuggable packages).

Use `-s` to install the application on the external SD card.

`-t` will allow to use test applications.

Chapter 157: Count Down Timer

Parameter	Details
`long millisInFuture`	The total duration the timer will run for, a.k.a how far in the future you want the timer to end. In milliseconds.
`long countDownInterval`	The interval at which you would like to receive timer updates. In milliseconds.
`long millisUntilFinished`	A parameter provided in `onTick()` that tells how long the CountDownTimer has remaining. In milliseconds

Section 157.1: Creating a simple countdown timer

CountDownTimer is useful for repeatedly performing an action in a steady interval for a set duration. In this example, we will update a text view every second for 30 seconds telling how much time is remaining. Then when the timer finishes, we will set the TextView to say "Done."

```
TextView textView = (TextView)findViewById(R.id.text_view);

CountDownTimer countDownTimer = new CountDownTimer(30000, 1000) {
    public void onTick(long millisUntilFinished) {
        textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished /
1000L));
    }

    public void onFinish() {
        textView.setText("Done.");
    }
}.start();
```

Section 157.2: A More Complex Example

In this example, we will pause/resume the CountDownTimer based off of the Activity lifecycle.

```
private static final long TIMER_DURATION = 60000L;
private static final long TIMER_INTERVAL = 1000L;

private CountDownTimer mCountDownTimer;
private TextView textView;

private long mTimeRemaining;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    textView = (TextView)findViewById(R.id.text_view); // Define in xml layout.

    mCountDownTimer = new CountDownTimer(TIMER_DURATION, TIMER_INTERVAL) {

        @Override
        public void onTick(long millisUntilFinished) {
            textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished /
1000L));
            mTimeRemaining = millisUntilFinished; // Saving timeRemaining in Activity for
pause/resume of CountDownTimer.
        }
```

```
        @Override
        public void onFinish() {
            textView.setText("Done.");
        }
    }.start();
}


@Override
protected void onResume() {
    super.onResume();

    if (mCountDownTimer == null) { // Timer was paused, re-create with saved time.
        mCountDownTimer = new CountDownTimer(timeRemaining, INTERVAL) {
            @Override
            public void onTick(long millisUntilFinished) {
                textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished
/ 1000L));
                timeRemaining = millisUntilFinished;
            }

            @Override
            public void onFinish() {
                textView.setText("Done.");
            }
        }.start();
    }
}

@Override
protected void onPause() {
    super.onPause();
    mCountDownTimer.cancel();
    mCountDownTimer = null;
}
```

Chapter 158: Barcode and QR code reading

Section 158.1: Using QRCodeReaderView (based on Zxing)

QRCodeReaderView implements an Android view which show camera and notify when there's a QR code inside the preview.

It uses the zxing open-source, multi-format 1D/2D barcode image processing library.

Adding the library to your project

Add QRCodeReaderView dependency to your build.gradle

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

First use

- Add to your layout a `QRCodeReaderView`

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
      android:id="@+id/qrdecoderview"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />
```

- Create an Activity which implements `onQRCodeReadListener`, and use it as a listener of the `QrCodeReaderView`.
- Make sure you have camera permissions in order to use the library. (https://developer.android.com/training/permissions/requesting.html)

Then in your Activity, you can use it as follows:

```
    public class DecoderActivity extends Activity implements OnQRCodeReadListener {

    private TextView resultTextView;
    private QRCodeReaderView qrCodeReaderView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_decoder);

        qrCodeReaderView = (QRCodeReaderView) findViewById(R.id.qrdecoderview);
        qrCodeReaderView.setOnQRCodeReadListener(this);

        // Use this function to enable/disable decoding
        qrCodeReaderView.setQRDecodingEnabled(true);

        // Use this function to change the autofocus interval (default is 5 secs)
        qrCodeReaderView.setAutofocusInterval(2000L);

        // Use this function to enable/disable Torch
        qrCodeReaderView.setTorchEnabled(true);

        // Use this function to set front camera preview
        qrCodeReaderView.setFrontCamera();
```

```
        // Use this function to set back camera preview
        qrCodeReaderView.setBackCamera();
    }

    // Called when a QR is decoded
    // "text" : the text encoded in QR
    // "points" : points where QR control points are placed in View
    @Override
    public void onQRCodeRead(String text, PointF[] points) {
        resultTextView.setText(text);
    }

    @Override
    protected void onResume() {
        super.onResume();
        qrCodeReaderView.startCamera();
    }

    @Override
    protected void onPause() {
        super.onPause();
        qrCodeReaderView.stopCamera();
    }
}
```

Chapter 159: Android PayPal Gateway Integration

Section 159.1: Setup PayPal in your android code

1) First go through Paypal Developer web site and create an application.

2) Now open your manifest file and give the below permissions

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

3) And some required Activity and Services

```
<service
       android:name="com.paypal.android.sdk.payments.PayPalService"
       android:exported="false" />
   <activity android:name="com.paypal.android.sdk.payments.PaymentActivity" />
   <activity android:name="com.paypal.android.sdk.payments.LoginActivity" />
   <activity android:name="com.paypal.android.sdk.payments.PaymentMethodActivity" />
   <activity android:name="com.paypal.android.sdk.payments.PaymentConfirmActivity" />
   <activity android:name="com.paypal.android.sdk.payments.PayPalFuturePaymentActivity" />
   <activity android:name="com.paypal.android.sdk.payments.FuturePaymentConsentActivity" />
   <activity android:name="com.paypal.android.sdk.payments.FuturePaymentInfoActivity" />
   <activity
       android:name="io.card.payment.CardIOActivity"
       android:configChanges="keyboardHidden|orientation" />
   <activity android:name="io.card.payment.DataEntryActivity" />
```

4) Open your Activity class and set Configuration for your app

```
//set the environment for production/sandbox/no netowrk
 private static final String CONFIG_ENVIRONMENT = PayPalConfiguration.ENVIRONMENT_PRODUCTION;
```

5) Now set client id from the Paypal developer account

```
private static final String CONFIG_CLIENT_ID = "PUT YOUR CLIENT ID";
```

6) Inside onCreate method call the Paypal service

```
Intent intent = new Intent(this, PayPalService.class);
intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION, config);
startService(intent);
```

7) Now you are ready to make a payment just on button press call the Payment Activity

```
PayPalPayment thingToBuy = new PayPalPayment(new BigDecimal(1),"USD", "androidhub4you.com",
                                PayPalPayment.PAYMENT_INTENT_SALE);
                            Intent intent = new Intent(MainActivity.this, PaymentActivity.class);
                            intent.putExtra(PaymentActivity.EXTRA_PAYMENT, thingToBuy);

                            startActivityForResult(intent, REQUEST_PAYPAL_PAYMENT);
```

8) And finally from the onActivityResult get the payment response

```
@Override
          protected void onActivityResult(int requestCode, int resultCode, Intent data) {
              if (requestCode == REQUEST_PAYPAL_PAYMENT) {
                  if (resultCode == Activity.RESULT_OK) {
                      PaymentConfirmation confirm = data
                              .getParcelableExtra(PaymentActivity.EXTRA_RESULT_CONFIRMATION);
                      if (confirm != null) {
                          try {
                          System.out.println("Responseeee"+confirm);
                              Log.i("paymentExample", confirm.toJSONObject().toString());

                              JSONObject jsonObj=new
JSONObject(confirm.toJSONObject().toString());

                              String paymentId=jsonObj.getJSONObject("response").getString("id");
                              System.out.println("payment id:-=="+paymentId);
                              Toast.makeText(getApplicationContext(), paymentId,
Toast.LENGTH_LONG).show();
                          } catch (JSONException e) {
                              Log.e("paymentExample", "an extremely unlikely failure occurred: ",
e);
                          }
                      }
                  } else if (resultCode == Activity.RESULT_CANCELED) {
                      Log.i("paymentExample", "The user canceled.");
                  } else if (resultCode == PaymentActivity.RESULT_EXTRAS_INVALID) {
                      Log.i("paymentExample", "An invalid Payment was submitted. Please see the
docs.");
                  }
              }


          }
```

Chapter 160: Drawables

Section 160.1: Custom Drawable

Extend your class with Drawable and override these methods

```
public class IconDrawable extends Drawable {
    /**
     * Paint for drawing the shape
     */
    private Paint paint;
    /**
     * Icon drawable to be drawn to the center of the shape
     */
    private Drawable icon;
    /**
     * Desired width and height of icon
     */
    private int desiredIconHeight, desiredIconWidth;

    /**
     * Public constructor for the Icon drawable
     *
     * @param icon            pass the drawable of the icon to be drawn at the center
     * @param backgroundColor background color of the shape
     */
    public IconDrawable(Drawable icon, int backgroundColor) {
        this.icon = icon;
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setColor(backgroundColor);
        desiredIconWidth = 50;
        desiredIconHeight = 50;
    }

    @Override
    public void draw(Canvas canvas) {
        //if we are setting this drawable to a 80dpX80dp imageview
        //getBounds will return that measurements,we can draw according to that width.
        Rect bounds = getBounds();
        //drawing the circle with center as origin and center distance as radius
        canvas.drawCircle(bounds.centerX(), bounds.centerY(), bounds.centerX(), paint);
        //set the icon drawable's bounds to the center of the shape
        icon.setBounds(bounds.centerX() - (desiredIconWidth / 2), bounds.centerY() -
(desiredIconHeight / 2), (bounds.centerX() - (desiredIconWidth / 2)) + desiredIconWidth,
(bounds.centerY() - (desiredIconHeight / 2)) + desiredIconHeight);
        //draw the icon to the bounds
        icon.draw(canvas);

    }

    @Override
    public void setAlpha(int alpha) {
        //sets alpha to your whole shape
        paint.setAlpha(alpha);
    }

    @Override
    public void setColorFilter(ColorFilter colorFilter) {
       //sets color filter to your whole shape
        paint.setColorFilter(colorFilter);
```

```
    }

    @Override
    public int getOpacity() {
        //give the desired opacity of the shape
        return PixelFormat.TRANSLUCENT;
    }
}
```

Declare a ImageView in your layout

```
<ImageView
  android:layout_width="80dp"
  android:id="@+id/imageView"
  android:layout_height="80dp" />
```

Set your custom drawable to the ImageView

```
 IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColo
r(this,R.color.pink_300));
 imageView.setImageDrawable(iconDrawable);
```

Screenshot

Section 160.2: Tint a drawable

A drawable can be tinted a certain color. This is useful for supporting different themes within your application, and reducing the number of drawable resource files.

Using framework APIs on SDK 21+:

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

Using android.support.v4 library on SDK 4+:

```
//Load the untinted resource
final Drawable drawableRes = ContextCompat.getDrawable(context, R.drawable.ic_launcher);
//Wrap it with the compatibility library so it can be altered
Drawable tintedDrawable = DrawableCompat.wrap(drawableRes);
//Apply a coloured tint
DrawableCompat.setTint(tintedDrawable, Color.WHITE);
//At this point you may use the tintedDrawable just as you usually would
//(and drawableRes can be discarded)

//NOTE: If your original drawableRes was in use somewhere (i.e. it was the result of
//a call to a `getBackground()` method then at this point you still need to replace
//the background. setTint does *not* alter the instance that drawableRes points to,
//but instead creates a new drawable instance
```

Please not that `int color` is **not** referring to a color Resource, however you are not limited to those colours defined

in the 'Color' class. When you have a colour defined in your XML which you want to use you must just first get it's value.

You can replace usages of `Color.WHITE` using the methods below

When targetting older API's:

```
getResources().getColor(R.color.your_color);
```

Or on newer targets:

```
ContextCompat.getColor(context, R.color.your_color);
```

Section 160.3: Circular View

For a circular View (in this case `TextView`) create a drawble **round_view.xml** in **drawble** folder:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="oval">
    <solid android:color="#FAA23C" />
    <stroke android:color="#FFF" android:width="2dp" />
</shape>
```

Assign the drawable to the View:

```
    <TextView
        android:id="@+id/game_score"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:background="@drawable/round_score"
        android:padding="6dp"
        android:text="100"
        android:textColor="#fff"
        android:textSize="20sp"
        android:textStyle="bold"
        android:gravity="center" />
```

Now it should look like the orange circle:

Section 160.4: Make View with rounded corners

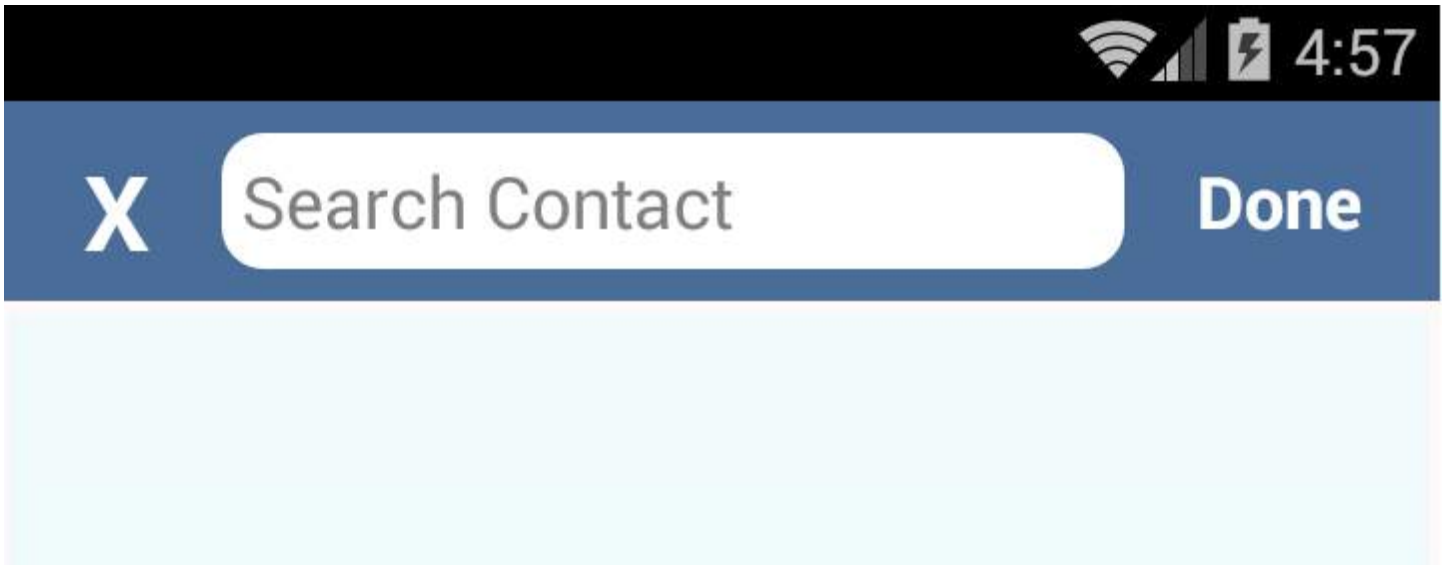
Create **drawable** file named with **custom_rectangle.xml** in **drawable** folder:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <solid android:color="@android:color/white" />

    <corners android:radius="10dip" />

    <stroke
        android:width="1dp"
        android:color="@android:color/white" />

</shape>
```

Now apply **rectangle background** on **View**:

```
mView.setBackGround(R.drawlable.custom_rectangle);
```

Reference screenshot:



Chapter 161: TransitionDrawable

Section 161.1: Animate views background color (switch-color) with TransitionDrawable

```
public void setCardColorTran(View view) {
   ColorDrawable[] color = {new ColorDrawable(Color.BLUE), new ColorDrawable(Color.RED)};
   TransitionDrawable trans = new TransitionDrawable(color);
    if(Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.JELLY_BEAN) {
        view.setBackgroundDrawable(trans);
        }else {
        view.setBackground(trans);
    }
    trans.startTransition(5000);
}
```

Section 161.2: Add transition or Cross-fade between two images

Step 1: Create a transition drawable in XML

Save this file `transition.xml` in `res/drawable` folder of your project.

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/image1" />
    <item android:drawable="@drawable/image2" />
</transition>
```

The image1 and image2 are the two images that we want to transition and they should be put in your `res/drawable` folder too.

Step 2: Add code for ImageView in your XML layout to display the above drawable.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context=".MainActivity" >

    <ImageView
       android:id="@+id/image_view"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:src="@drawable/image1" />

</LinearLayout>
```

Step 3: Access the XML transition drawable in onCreate() method of your Activity and start transition in onClick() event.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    imageView = (ImageView) findViewById(R.id.image_view);
    transitionDrawable = (TransitionDrawable)
```

```
        ContextCompat.getDrawable(this, R.drawable.transition);

    birdImageView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(final View view) {
            birdImageView.setImageDrawable(transitionDrawable);
            transitionDrawable.startTransition(1000);
        }
    });
}
```

Chapter 162: Vector Drawables

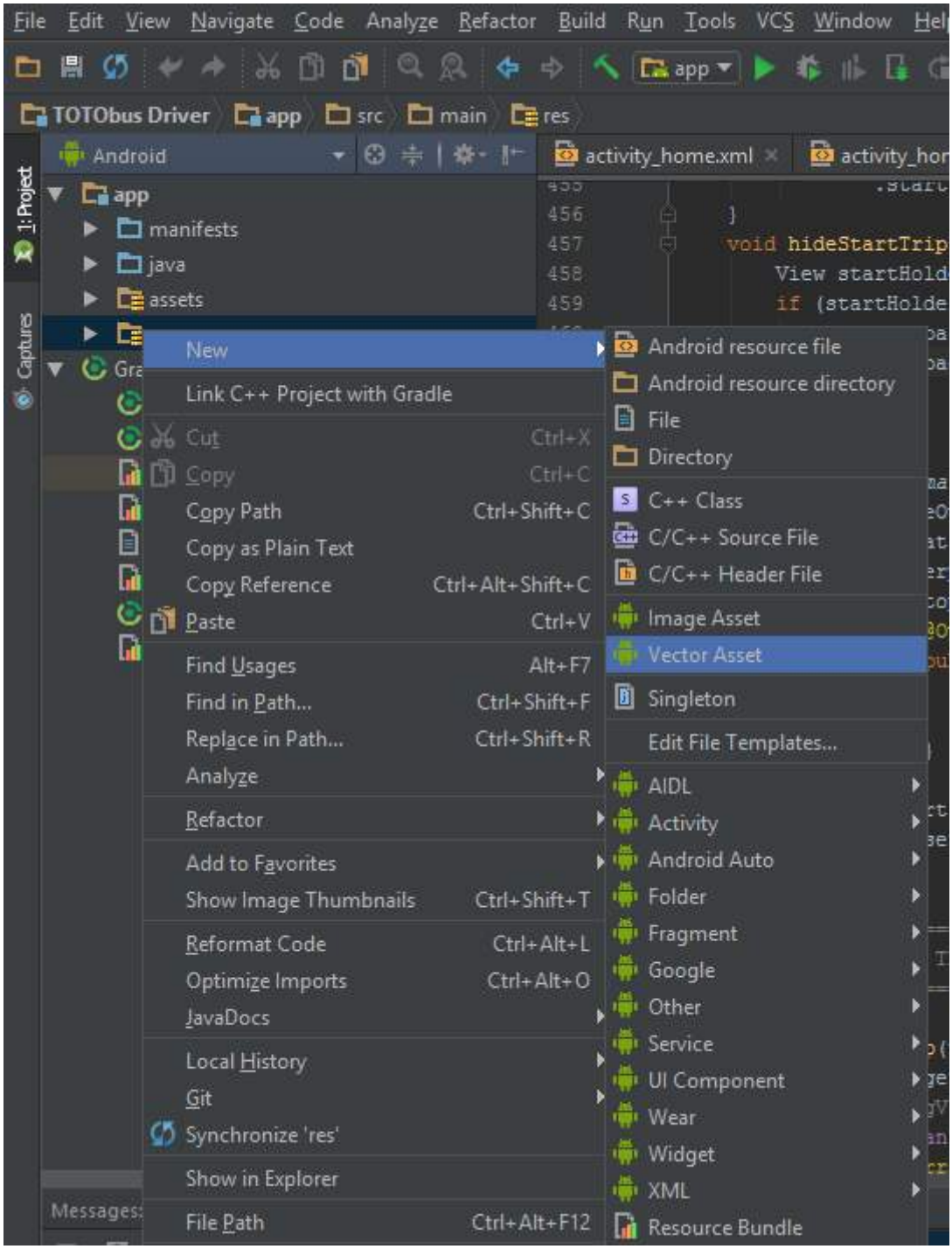
Parameter	Details
`<vector>`	Used to define a vector drawable
`<group>`	Defines a group of paths or subgroups, plus transformation information. The transformations are defined in the same coordinates as the viewport. And the transformations are applied in the order of scale, rotate then translate.
`<path>`	Defines paths to be drawn.
`<clip-path>`	Defines path to be the current clip. Note that the clip path only apply to the current group and its children.

As the name implies, vector drawables are based on vector graphics. Vector graphics are a way of describing graphical elements using geometric shapes. This lets you create a drawable based on an XML vector graphic. Now there is no need to design different size image for mdpi, hdpi, xhdpi and etc. With Vector Drawable you need to create image only once as an xml file and you can scale it for all dpi and for different devices. This also not save space but also simplifies maintenance.

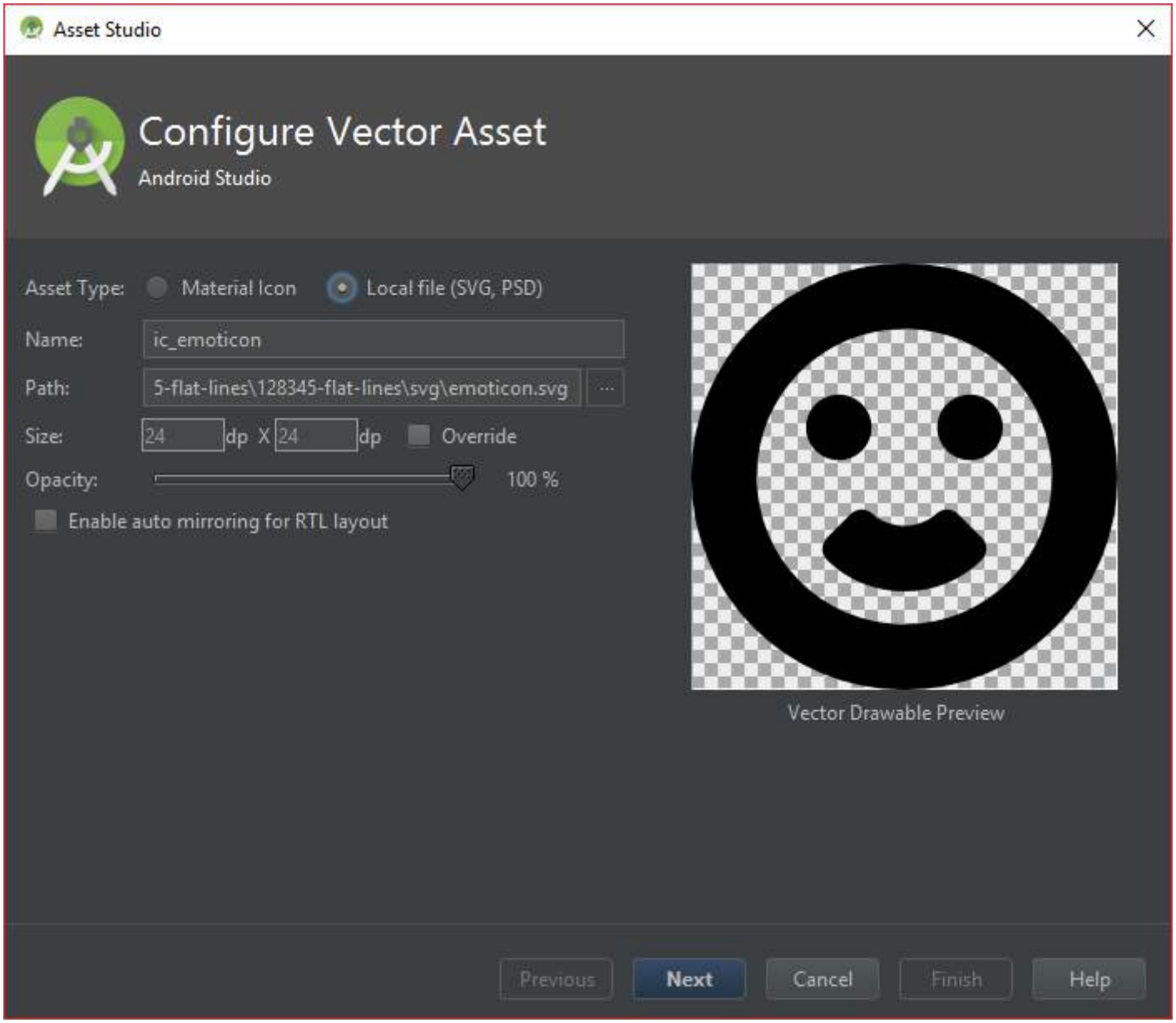
Section 162.1: Importing SVG file as VectorDrawable

You can import an *SVG* file as a `VectorDrawable` in Android Studio, follow these steps :

"Right-click" on the `res` folder and select **new** > **Vector Asset**.



Select the **Local File** option and browse to your .svg file. Change the options to your liking and hit next. Done.



Section 162.2: VectorDrawable Usage Example

Here's an example vector asset which we're actually using in AppCompat:

res/drawable/ic_search.xml

```
<vector xmlns:android="..."
        android:width="24dp"
        android:height="24dp"
        android:viewportWidth="24.0"
        android:viewportHeight="24.0"
        android:tint="?attr/colorControlNormal">

    <path
        android:pathData="..."
        android:fillColor="@android:color/white"/>

</vector>
```

Using this drawable, an example `ImageView` declaration would be:

```
<ImageView
```

```
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   app:srcCompat="@drawable/ic_search"/>
```

You can also set it at run-time:

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

The same attribute and calls work for `ImageButton` too.

Section 162.3: VectorDrawable xml example

Here is a simple `VectorDrawable` in this **vectordrawable.xml** file.

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:height="64dp"
   android:width="64dp"
   android:viewportHeight="600"
   android:viewportWidth="600" >
    <group
       android:name="rotationGroup"
       android:pivotX="300.0"
       android:pivotY="300.0"
       android:rotation="45.0" >
        <path
           android:name="v"
           android:fillColor="#000000"
           android:pathData="M300,70 l 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

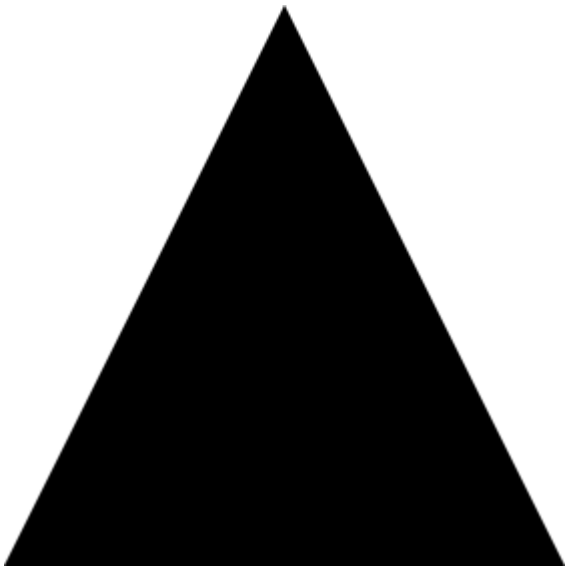
Chapter 163: VectorDrawable and AnimatedVectorDrawable

Section 163.1: Basic VectorDrawable

A `VectorDrawable` should consist of at least one **`<path>`** tag defining a shape

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <path
        android:fillColor="#FF000000"
        android:pathData="M0,24 l12,-24 l12,24 z"/>
</vector>
```

This would produce a black triangle:



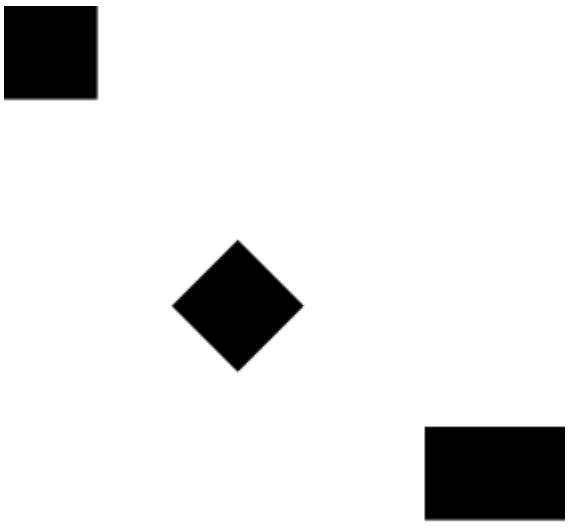
Section 163.2: <group> tags

A **`<group>`** tag allows the scaling, rotation, and position of one or more elements of a `VectorDrawable` to be adjusted:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
android:width="24dp"
android:height="24dp"
android:viewportWidth="24.0"
android:viewportHeight="24.0">
    <path
       android:pathData="M0,0 h4 v4 h-4 z"
       android:fillColor="#FF000000"/>

    <group
       android:name="middle square group"
       android:translateX="10"
       android:translateY="10"
       android:rotation="45">
        <path
```

```
            android:pathData="M0,0 h4 v4 h-4 z"
            android:fillColor="#FF000000"/>
    </group>

    <group
       android:name="last square group"
       android:translateX="18"
       android:translateY="18"
       android:scaleX="1.5">
        <path
           android:pathData="M0,0 h4 v4 h-4 z"
           android:fillColor="#FF000000"/>
    </group>
</vector>
```

The example code above contains three identical `<path>` tags, all describing black squares. The first square is unadjusted. The second square is wrapped in a `<group>` tag which moves it and rotates it by 45°. The third square is wrapped in a `<group>` tag which moves it and stretches it horizontally by 50%. The result is as follows:



A `<group>` tag can contain multiple `<path>` and `<clip-path>` tags. It can even contain another `<group>`.

Section 163.3: Basic AnimatedVectorDrawable

An `AnimatedVectorDrawable` requires at least 3 components:

- A `VectorDrawable` which will be manipulated
- An `objectAnimator` which defines what property to change and how
- The `AnimatedVectorDrawable` itself which connects the `objectAnimator` to the `VectorDrawable` to create the animation

The following creates a triangle that transitions its color from black to red.

The `VectorDrawable`, filename: `triangle_vector_drawable.xml`

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:width="24dp"
   android:height="24dp"
   android:viewportWidth="24.0"
   android:viewportHeight="24.0">

    <path
```

```
        android:name="triangle"
        android:fillColor="@android:color/black"
        android:pathData="M0,24 l12,-24 l12,24 z"/>

</vector>
```

The `objectAnimator`, filename: `color_change_animator.xml`

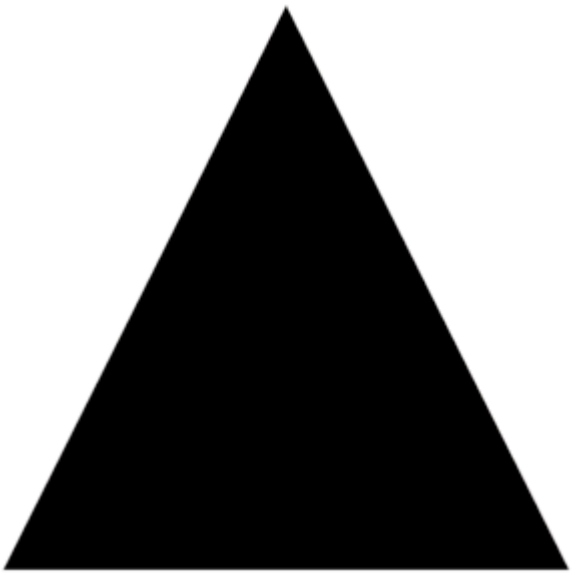
```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="fillColor"
    android:duration="2000"
    android:repeatCount="infinite"
    android:valueFrom="@android:color/black"
    android:valueTo="@android:color/holo_red_light"/>
```

The `AnimatedVectorDrawable`, filename: `triangle_animated_vector.xml`

```
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/triangle_vector_drawable">

     <target
        android:animation="@animator/color_change_animator"
        android:name="triangle"/>

</animated-vector>
```

Note that the **`<target>`** specifies `android:name="triangle"` which matches the **`<path>`** in the `VectorDrawable`. A `VectorDrawable` may contain multiple elements and the `android:name` property is used to define which element is being targeted.

Result:



Section 163.4: Using Strokes

Using SVG stroke makes it easier to create a Vector drawable with unified stroke length, as per Material Design guidelines:

> Consistent stroke weights are key to unifying the overall system icon family. Maintain a 2dp width for all

> stroke instances, including curves, angles, and both interior and exterior strokes.

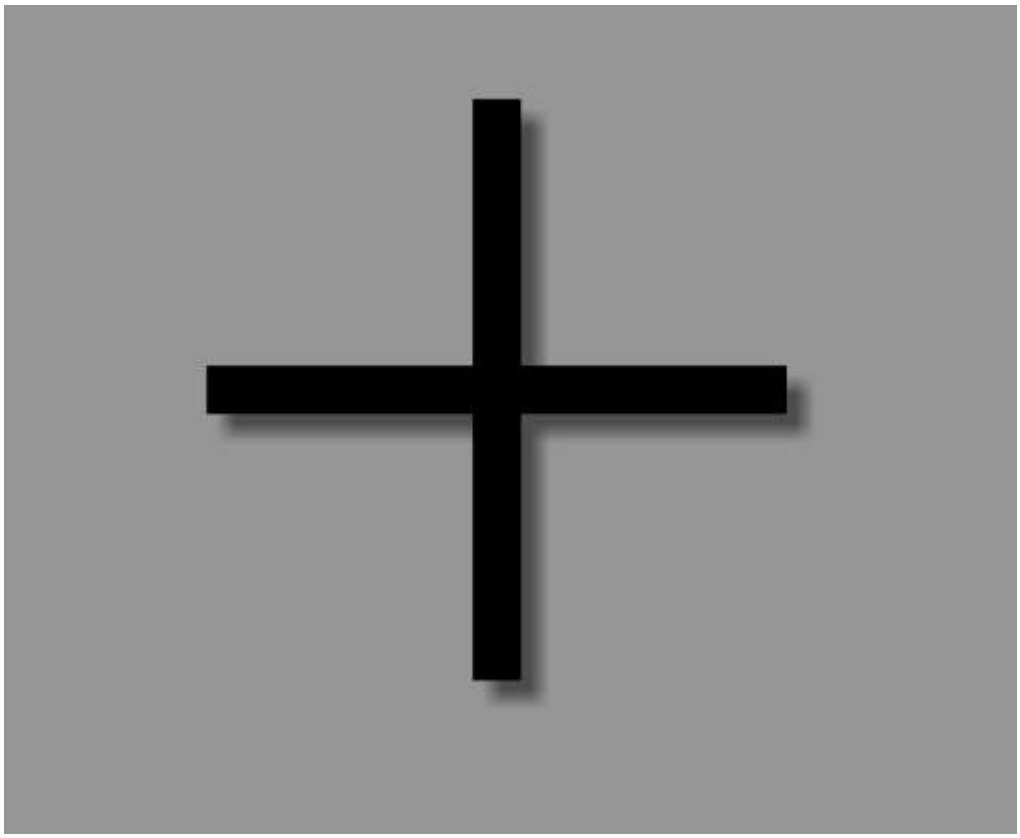
So, for example, this is how you would create a "plus" sign using strokes:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
android:width="24dp"
android:height="24dp"
android:viewportHeight="24.0"
android:viewportWidth="24.0">
    <path
       android:fillColor="#FF000000"
       android:strokeColor="#F000"
       android:strokeWidth="2"
       android:pathData="M12,0 V24 M0,12 H24" />
</vector>
```

- `strokeColor` defines the color of the stroke.
- `strokeWidth` defines the width (in dp) of the stroke (2dp in this case, as suggested by the guidelines).
- `pathData` is where we describe our SVG image:
- `M12,0` moves the "cursor" to the position 12,0
- `V24` creates a vertical line to the position 12, 24

etc., see SVG documentation and this useful "SVG Path" tutorial from w3schools to learn more about the specific path commands.

As a result, we got this no-frills plus sign:



This is **especially useful** for creating an `AnimatedVectorDrawable`, since you are now operating with a single stroke with an unified length, instead of an otherwise complicated path.

Section 163.5: Using <clip-path>

A `<clip-path>` defines a shape which acts as a window, only allowing parts of a `<path>` to show if they are within the `<clip-path>` shape and cutting off the rest.

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <clip-path
        android:name="square clip path"
        android:pathData="M6,6 h12 v12 h-12 z"/>
    <path
        android:name="triangle"
        android:fillColor="#FF000000"
        android:pathData="M0,24 l12,-24 l12,24 z"/>

</vector>
```

In this case the `<path>` produces a black triangle, but the `<clip-path>` defines a smaller square shape, only allowing part of the triangle to show through:

Section 163.6: Vector compatibility through AppCompat

A few pre-requisites in the `build.gradle` for vectors to work all the way down to API 7 for VectorDrawables and API 13 for AnimatedVectorDrawables (with some caveats currently):

```
//Build Tools has to be 24+
buildToolsVersion '24.0.0'

defaultConfig {
    vectorDrawables.useSupportLibrary = true
    generatedDensities = []
    aaptOptions {
        additionalParameters "--no-version-vectors"
    }
}

dependencies {
```

```
    compile 'com.android.support:appcompat-v7:24.1.1'
}
```

In your `layout.xml`:

```
<ImageView
   android:id="@+id/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   appCompat:src="@drawable/vector_drawable"
   android:contentDescription="@null" />
```

Chapter 164: Port Mapping using Cling library in Android

Section 164.1: Mapping a NAT port

```
String myIp = getIpAddress();
int port = 55555;

//creates a port mapping configuration with the external/internal port, an internal host IP, the
protocol and an optional description
PortMapping[] desiredMapping = new PortMapping[2];
desiredMapping[0] = new PortMapping(port,myIp, PortMapping.Protocol.TCP);
desiredMapping[1] = new PortMapping(port,myIp, PortMapping.Protocol.UDP);

//starting the UPnP service
UpnpService upnpService = new UpnpServiceImpl(new AndroidUpnpServiceConfiguration());
RegistryListener registryListener = new PortMappingListener(desiredMapping);
upnpService.getRegistry().addListener(registryListener);
upnpService.getControlPoint().search();



//method for getting local ip
private String getIpAddress() {
    String ip = "";
    try {
        Enumeration<NetworkInterface> enumNetworkInterfaces = NetworkInterface
                .getNetworkInterfaces();
        while (enumNetworkInterfaces.hasMoreElements()) {
            NetworkInterface networkInterface = enumNetworkInterfaces
                    .nextElement();
            Enumeration<InetAddress> enumInetAddress = networkInterface
                    .getInetAddresses();
            while (enumInetAddress.hasMoreElements()) {
                InetAddress inetAddress = enumInetAddress.nextElement();

                if (inetAddress.isSiteLocalAddress()) {
                    ip +=inetAddress.getHostAddress();
                }
            }
        }
    } catch (SocketException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        ip += "Something Wrong! " + e.toString() + "\n";
    }
    return ip;
   }
```

Section 164.2: Adding Cling Support to your Android Project

build.gradle

```
repositories {
  maven { url 'http://4thline.org/m2' }
}

dependencies {
```

```
// Cling
compile 'org.fourthline.cling:cling-support:2.1.0'

//Other dependencies required by Cling
compile 'org.eclipse.jetty:jetty-server:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-servlet:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-client:8.1.18.v20150929'
compile 'org.slf4j:slf4j-jdk14:1.7.14'

}
```

Chapter 165: Creating Overlay (always-on-top) Windows

Section 165.1: Popup overlay

In order to put your view on top of every application, you have to assign your view to the corresponding window manager. For that you need the system alert permission, which can be requested by adding the following line to your manifest file:

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
```

Note: If your application gets destroyed, your view will be removed from the window manager. Therefore, it is better to create the view and assign it to the window manager by a foreground service.

Assigning a view to the WindowManager

You can retrieve a window manager instance as follows:

```
WindowManager mWindowManager = (WindowManager) mContext.getSystemService(Context.WINDOW_SERVICE);
```

In order to define the position of your view, you have to create some layout parameters as follows:

```
WindowManager.LayoutParams mLayoutParams = new WindowManager.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT,
        WindowManager.LayoutParams.TYPE_PHONE,
        WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON,
        PixelFormat.TRANSLUCENT);
mLayoutParams.gravity = Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL;
```

Now, you can assign your view together with the created layout parameters to the window manager instance as follows:

```
mWindowManager.addView(yourView, mLayoutParams);
```

Voila! Your view has been successfully placed on top of all other applications.

Note: You view will not be put on top of the keyguard.

Section 165.2: Granting SYSTEM_ALERT_WINDOW Permission on android 6.0 and above

From android 6.0 this permission needs to grant dynamically,

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
```

Throwing below permission denied error on 6.0,

```
Caused by: android.view.WindowManager$BadTokenException: Unable to add window
android.view.ViewRootImpl$W@86fb55b -- permission denied for this window type
```

Solution:

Requesting Overlay permission as below,

```
if(!Settings.canDrawOverlays(this)){
    // ask for setting
     Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION,
     Uri.parse("package:" + getPackageName()));
     startActivityForResult(intent, REQUEST_OVERLAY_PERMISSION);
}
```

Check for the result,

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_OVERLAY_PERMISSION) {
        if (Settings.canDrawOverlays(this)) {
            // permission granted...
        }else{
            // permission not granted...
        }
    }
}
```

Chapter 166: ExoPlayer

Section 166.1: Add ExoPlayer to the project

Via jCenter

including the following in your project's build.gradle file:

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

where rX.X.X is the your preferred version. For the latest version, see the project's Releases. For more details, see the project on Bintray.

Section 166.2: Using ExoPlayer

Instantiate your ExoPlayer:

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

To play audio only you can use these values:

```
RENDERER_COUNT = 1 //since you want to render simple audio
minBufferMs = 1000
minRebufferMs = 5000
```

Both buffer values can be tweaked according to your requirements.

Now you have to create a DataSource. When you want to stream mp3 you can use the DefaultUriDataSource. You have to pass the Context and a UserAgent. To keep it simple play a local file and pass null as userAgent:

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

Then create the sampleSource:

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(
                uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uri points to your file, as an Extractor you can use a simple default Mp3Extractor if you want to play mp3. requestedBufferSize can be tweaked again according to your requirements. Use 5000 for example.

Now you can create your audio track renderer using the sample source as follows:

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

Finally call prepare on your exoPlayer instance:

```
exoPlayer.prepare(audioRenderer);
```

To start playback call:

```
exoPlayer.setPlayWhenReady(true);
```

Section 166.3: Main steps to play video & audio using the standard TrackRenderer implementations

```
// 1. Instantiate the player.
player = ExoPlayer.Factory.newInstance(RENDERER_COUNT);
// 2. Construct renderers.
MediaCodecVideoTrackRenderer videoRenderer = ...
MediaCodecAudioTrackRenderer audioRenderer = ...
// 3. Inject the renderers through prepare.
player.prepare(videoRenderer, audioRenderer);
// 4. Pass the surface to the video renderer.
player.sendMessage(videoRenderer, MediaCodecVideoTrackRenderer.MSG_SET_SURFACE,     surface);
// 5. Start playback.
player.setPlayWhenReady(true);
...
player.release(); // Don't forget to release when done!
```

Chapter 167: XMPP register login and chat simple example

Section 167.1: XMPP register login and chat basic example

Install openfire or any chat server in your system or on server. For more details click here.

Create android project and add these libraries in gradle:

```
compile 'org.igniterealtime.smack:smack-android:4.2.0'
compile 'org.igniterealtime.smack:smack-tcp:4.2.0'
compile 'org.igniterealtime.smack:smack-im:4.2.0'
compile 'org.igniterealtime.smack:smack-android-extensions:4.2.0'
```

Next create one xmpp class from xmpp connection purpose:

```
public class XMPP {

public static final int PORT = 5222;
private static XMPP instance;
private XMPPTCPConnection connection;
private static String TAG = "XMPP-EXAMPLE";
public static final String ACTION_LOGGED_IN = "liveapp.loggedin";
private String HOST = "192.168.0.10";

private XMPPTCPConnectionConfiguration buildConfiguration() throws XmppStringprepException {
    XMPPTCPConnectionConfiguration.Builder builder =
            XMPPTCPConnectionConfiguration.builder();


    builder.setHost(HOST);
    builder.setPort(PORT);
    builder.setCompressionEnabled(false);
    builder.setDebuggerEnabled(true);
    builder.setSecurityMode(ConnectionConfiguration.SecurityMode.disabled);
    builder.setSendPresence(true);

    if (Build.VERSION.SDK_INT >= 14) {
        builder.setKeystoreType("AndroidCAStore");
        // config.setTruststorePassword(null);
        builder.setKeystorePath(null);
    } else {
        builder.setKeystoreType("BKS");
        String str = System.getProperty("javax.net.ssl.trustStore");
        if (str == null) {
            str = System.getProperty("java.home") + File.separator + "etc" + File.separator +
"security"
                    + File.separator + "cacerts.bks";
        }
        builder.setKeystorePath(str);
    }
    DomainBareJid serviceName = JidCreate.domainBareFrom(HOST);
    builder.setServiceName(serviceName);


    return builder.build();
}
```

```
private XMPPTCPConnection getConnection() throws XMPPException, SmackException, IOException,
InterruptedException {
    Log.logDebug(TAG, "Getting XMPP Connect");
    if (isConnected()) {
        Log.logDebug(TAG, "Returning already existing connection");
        return this.connection;
    }

    long l = System.currentTimeMillis();
    try {
        if(this.connection != null){
            Log.logDebug(TAG, "Connection found, trying to connect");
            this.connection.connect();
        }else{
            Log.logDebug(TAG, "No Connection found, trying to create a new connection");
            XMPPTCPConnectionConfiguration config = buildConfiguration();
            SmackConfiguration.DEBUG = true;
            this.connection = new XMPPTCPConnection(config);
            this.connection.connect();
        }
    } catch (Exception e) {
        Log.logError(TAG,"some issue with getting connection :" + e.getMessage());

    }

    Log.logDebug(TAG, "Connection Properties: " + connection.getHost() + " " +
connection.getServiceName());
    Log.logDebug(TAG, "Time taken in first time connect: " + (System.currentTimeMillis() - l));
    return this.connection;
}

public static XMPP getInstance() {
    if (instance == null) {
        synchronized (XMPP.class) {
            if (instance == null) {
                instance = new XMPP();
            }
        }
    }
    return instance;
}

public void close() {
    Log.logInfo(TAG, "Inside XMPP close method");
    if (this.connection != null) {
        this.connection.disconnect();
    }
}

private XMPPTCPConnection connectAndLogin(Context context) {
    Log.logDebug(TAG, "Inside connect and Login");
    if (!isConnected()) {
        Log.logDebug(TAG, "Connection not connected, trying to login and connect");
        try {
           // Save username and password then use here
            String username = AppSettings.getUser(context);
            String password = AppSettings.getPassword(context);
            this.connection = getConnection();
            Log.logDebug(TAG, "XMPP username :" + username);
            Log.logDebug(TAG, "XMPP password :" + password);
            this.connection.login(username, password);
            Log.logDebug(TAG, "Connect and Login method, Login successful");
```

```
            context.sendBroadcast(new Intent(ACTION_LOGGED_IN));
        } catch (XMPPException localXMPPException) {
            Log.logError(TAG, "Error in Connect and Login Method");
            localXMPPException.printStackTrace();
        } catch (SmackException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (IOException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (InterruptedException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (Exception e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        }
    }
    Log.logInfo(TAG, "Inside getConnection - Returning connection");
    return this.connection;
}

public boolean isConnected() {
    return (this.connection != null) && (this.connection.isConnected());
}

public EntityFullJid getUser() {
    if (isConnected()) {
        return connection.getUser();
    } else {
        return null;
    }
}

public void login(String user, String pass, String username)
        throws XMPPException, SmackException, IOException, InterruptedException,
PurplKiteXMPPConnectException {
    Log.logInfo(TAG, "inside XMPP getlogin Method");
    long l = System.currentTimeMillis();
    XMPPTCPConnection connect = getConnection();
    if (connect.isAuthenticated()) {
        Log.logInfo(TAG, "User already logged in");
        return;
    }

    Log.logInfo(TAG, "Time taken to connect: " + (System.currentTimeMillis() - l));

    l = System.currentTimeMillis();
    try{
        connect.login(user, pass);
    }catch (Exception e){
        Log.logError(TAG, "Issue in login, check the stacktrace");
        e.printStackTrace();
    }

    Log.logInfo(TAG, "Time taken to login: " + (System.currentTimeMillis() - l));

    Log.logInfo(TAG, "login step passed");
```

```
    PingManager pingManager = PingManager.getInstanceFor(connect);
    pingManager.setPingInterval(5000);

}

public void register(String user, String pass) throws XMPPException,
SmackException.NoResponseException, SmackException.NotConnectedException {
    Log.logInfo(TAG, "inside XMPP register method, " + user + " : " + pass);
    long l = System.currentTimeMillis();
    try {
        AccountManager accountManager = AccountManager.getInstance(getConnection());
        accountManager.sensitiveOperationOverInsecureConnection(true);
        accountManager.createAccount(Localpart.from(user), pass);
    } catch (SmackException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (PurplKiteXMPPConnectException e) {
        e.printStackTrace();
    }
    Log.logInfo(TAG, "Time taken to register: " + (System.currentTimeMillis() - l));
}


public void addStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection =  connectAndLogin(context);
    connection.addAsyncStanzaListener(stanzaListener, null);
}

public void removeStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection =  connectAndLogin(context);
    connection.removeAsyncStanzaListener(stanzaListener);
}

public void addChatListener(Context context, ChatManagerListener chatManagerListener){
    ChatManager.getInstanceFor(connectAndLogin(context))
            .addChatListener(chatManagerListener);
}

public void removeChatListener(Context context, ChatManagerListener chatManagerListener){
    ChatManager.getInstanceFor(connectAndLogin(context)).removeChatListener(chatManagerListener);
}

public void getSrvDeliveryManager(Context context){
    ServiceDiscoveryManager sdm = ServiceDiscoveryManager
            .getInstanceFor(XMPP.getInstance().connectAndLogin(
                    context));
    //sdm.addFeature("http://jabber.org/protocol/disco#info");
    //sdm.addFeature("jabber:iq:privacy");
    sdm.addFeature("jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/disco#info");
    sdm.addFeature("jabber:iq:privacy");

}

public String getUserLocalPart(Context context){
   return  connectAndLogin(context).getUser().getLocalpart().toString();
}
```

```
public EntityFullJid getUser(Context context){
    return  connectAndLogin(context).getUser();
}

public Chat getThreadChat(Context context, String party1, String party2){
    Chat chat = ChatManager.getInstanceFor(
            XMPP.getInstance().connectAndLogin(context))
            .getThreadChat(party1 + "-" + party2);
    return chat;
}

public Chat createChat(Context context, EntityJid jid, String party1, String party2,
ChatMessageListener messageListener){
    Chat chat = ChatManager.getInstanceFor(
            XMPP.getInstance().connectAndLogin(context))
            .createChat(jid, party1 + "-" + party2,
                    messageListener);
    return chat;
}

public void sendPacket(Context context, Stanza packet){
    try {
        connectAndLogin(context).sendStanza(packet);
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

}
```

Finally, add this activiy:

```
private UserLoginTask mAuthTask = null;
private ChatManagerListener chatListener;
private Chat chat;
private Jid opt_jid;
private ChatMessageListener messageListener;
private StanzaListener packetListener;

private boolean register(final String paramString1,final String paramString2) {
    try {
        XMPP.getInstance().register(paramString1, paramString2);
        return true;

    } catch (XMPPException localXMPPException) {
        localXMPPException.printStackTrace();
    } catch (SmackException.NoResponseException e) {
        e.printStackTrace();
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    }
    return false;
}

private boolean login(final String user,final String pass,final String username) {

    try {

        XMPP.getInstance().login(user, pass, username);
```

```
            sendBroadcast(new Intent("liveapp.loggedin"));

            return true;
        } catch (Exception e) {
            e.printStackTrace();
            try {

                XMPP.getInstance()
                        .login(user, pass, username);
                sendBroadcast(new Intent("liveapp.loggedin"));

                return true;
            } catch (XMPPException e1) {
                e1.printStackTrace();
            } catch (SmackException e1) {
                e1.printStackTrace();
            } catch (InterruptedException e1) {
                e1.printStackTrace();
            } catch (IOException e1) {
                e1.printStackTrace();
            }catch (Exception e1){
                e1.printStackTrace();
            }
        }
        return false;
    }

    public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

        public UserLoginTask() {
        }

        protected Boolean doInBackground(Void... paramVarArgs) {
            String mEmail = "abc";
            String mUsername = "abc";
            String mPassword = "welcome";

            if (register(mEmail, mPassword)) {
                try {
                    XMPP.getInstance().close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            return login(mEmail, mPassword, mUsername);

        }

        protected void onCancelled() {
            mAuthTask = null;

        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();

        }

        protected void onPostExecute(Boolean success) {
            mAuthTask = null;
            try {
```

```
            if (success) {

              messageListener = new ChatMessageListener() {
                @Override
                public void processMessage(Chat chat, Message message) {

                // here you will get only connected user by you

                }
        };


        packetListener = new StanzaListener() {
                @Override
                public void processPacket(Stanza packet) throws
SmackException.NotConnectedException, InterruptedException {

                    if (packet instanceof Message) {
                        final Message message = (Message) packet;

                 // here you will get all messages send by anybody
                    }
                }
        };

        chatListener = new ChatManagerListener() {

                @Override
                public void chatCreated(Chat chatCreated, boolean local) {
                    onChatCreated(chatCreated);
                }
        };


        try {
        String opt_jidStr = "abc";

        try {
            opt_jid = JidCreate.bareFrom(Localpart.from(opt_jidStr), Domainpart.from(HOST));
        } catch (XmppStringprepException e) {
            e.printStackTrace();
        }
        String addr1 = XMPP.getInstance().getUserLocalPart(getActivity());
        String addr2 = opt_jid.toString();
        if (addr1.compareTo(addr2) > 0) {
            String addr3 = addr2;
            addr2 = addr1;
            addr1 = addr3;
        }
        chat = XMPP.getInstance().getThreadChat(getActivity(), addr1, addr2);
        if (chat == null) {
            chat = XMPP.getInstance().createChat(getActivity(), (EntityJid) opt_jid, addr1, addr2,
messageListener);
            PurplkiteLogs.logInfo(TAG, "chat value single chat 1 :" + chat);
        } else {
            chat.addMessageListener(messageListener);
            PurplkiteLogs.logInfo(TAG, "chat value single chat  2:" + chat);
        }

        } catch (Exception e) {
        e.printStackTrace();
        }
```

```
        XMPP.getInstance().addStanzaListener(getActivity(), packetListener);
        XMPP.getInstance().addChatListener(getActivity(), chatListener);
        XMPP.getInstance().getSrvDeliveryManager(getActivity());

            } else {

            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
}

/**
 * user attemptLogin for xmpp
 *
 */

private void attemptLogin() {
    if (mAuthTask != null) {
        return;
    }

    boolean cancel = false;
    View focusView = null;

    if (cancel) {
        focusView.requestFocus();
    } else {
        try {
            mAuthTask = new UserLoginTask();
            mAuthTask.execute((Void) null);
        } catch (Exception e) {

        }

    }
}

void onChatCreated(Chat chatCreated) {
    if (chat != null) {
        if (chat.getParticipant().getLocalpart().toString().equals(
                chatCreated.getParticipant().getLocalpart().toString())) {
            chat.removeMessageListener(messageListener);
            chat = chatCreated;
            chat.addMessageListener(messageListener);
        }
    } else {
        chat = chatCreated;
        chat.addMessageListener(messageListener);
    }
}

private void sendMessage(String message) {
    if (chat != null) {
        try {
            chat.sendMessage(message);
        } catch (SmackException.NotConnectedException e) {
            e.printStackTrace();
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}

 @Override
public void onDestroy() {
    // TODO Auto-generated method stub
    super.onDestroy();
    try {
        XMPP.getInstance().removeChatListener(getActivity(), chatListener);
        if (chat != null && messageListener != null) {
            XMPP.getInstance().removeStanzaListener(getActivity(), packetListener);
            chat.removeMessageListener(messageListener);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

Make sure the internet permission is added in your manifest file.

Chapter 168: Android Authenticator

Section 168.1: Basic Account Authenticator Service

The Android Account Authenticator system can be used to make the client authenticate with a remote server. Three pieces of information are required:

- A service, triggered by the `android.accounts.AccountAuthenticator`. Its `onBind` method should return a subclass of `AbstractAccountAuthenticator`.
- An activity to prompt the user for credentials (Login activity)
- An xml resource file to describe the account

1. The service:

Place the following permissions in your AndroidManifest.xml:

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
```

Declare the service in the manifest file:

```
<service android:name="com.example.MyAuthenticationService">
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator" />
    </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator" />
</service>
```

Note that the `android.accounts.AccountAuthenticator` is included within the `intent-filter` tag. The xml resource (named `authenticator` here) is specified in the `meta-data` tag.

The service class:

```
public class MyAuthenticationService extends Service {

    private static final Object lock = new Object();
    private MyAuthenticator mAuthenticator;

    public MyAuthenticationService() {
        super();
    }

    @Override
    public void onCreate() {
        super.onCreate();

        synchronized (lock) {
            if (mAuthenticator == null) {
                mAuthenticator = new MyAuthenticator(this);
            }
        }
    }
```

```
    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }

}
```

2. The xml resource:

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
   android:accountType="com.example.account"
   android:icon="@drawable/appicon"
   android:smallIcon="@drawable/appicon"
   android:label="@string/app_name" />
```

Do not directly assign a string to `android:label` or assign missing drawables. It will crash without warning.

3. Extend the AbstractAccountAuthenticator class:

```
public class MyAuthenticator extends AbstractAccountAuthenticator {

    private Context mContext;

    public MyAuthenticator(Context context) {
        super(context);
        mContext = context;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse response,
                             String accountType,
                             String authTokenType,
                             String[] requiredFeatures,
                             Bundle options) throws NetworkErrorException {

        Intent intent = new Intent(mContext, LoginActivity.class);
        intent.putExtra(AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE, response);

        Bundle bundle = new Bundle();
        bundle.putParcelable(AccountManager.KEY_INTENT, intent);

        return bundle;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account, Bundle
options) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String
authTokenType, Bundle options) throws NetworkErrorException {
        return null;
    }
```

```
    @Override
    public String getAuthTokenLabel(String authTokenType) {
        return null;
    }

    @Override
    public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[]
features) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account, String
authTokenType, Bundle options) throws NetworkErrorException {
        return null;
    }
}
```

The `addAccount()` method in `AbstractAccountAuthenticator` class is important as this method is called when adding an account from the "Add Account" screen in under settings.

`AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE` is important, as it will include the AccountAuthenticatorResponse object that is needed to return the account keys upon successful user verification.

Chapter 169: AudioManager

Section 169.1: Requesting Transient Audio Focus

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN_TRANSIENT);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // You now have the audio focus and may play sound.
            // When the sound has been played you give the focus back.
            audioManager.abandonAudioFocus(changedListener);
        }
    }
}
```

Section 169.2: Requesting Audio Focus

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // You now have the audio focus and may play sound.
        }
        else if (focusChange == AudioManager.AUDIOFOCUS_REQUEST_FAILED) {
            // Handle the failure.
        }
    }
}
```

Chapter 170: AudioTrack

Section 170.1: Generate tone of a specific frequency

To play a sound of with a specific tone,we first have to create a sine wave sound.This is done in the following way.

```
final int duration = 10; // duration of sound
final int sampleRate = 22050; // Hz (maximum frequency is 7902.13Hz (B8))
final int numSamples = duration * sampleRate;
final double samples[] = new double[numSamples];
final short buffer[] = new short[numSamples];
for (int i = 0; i < numSamples; ++i)
{
 samples[i] = Math.sin(2 * Math.PI * i / (sampleRate / note[0])); // Sine wave
 buffer[i] = (short) (samples[i] * Short.MAX_VALUE);  // Higher amplitude increases volume
}
```

Now we have to configure AudioTrack to play in accordance with the generated buffer . It is done in the following manner

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                    sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                    AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                    AudioTrack.MODE_STATIC);
```

Write the generated buffer and play the track

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

Hope this helps :)

Chapter 171: Job Scheduling

Section 171.1: Basic usage

Create a new JobService

This is done by extending the `JobService` class and implementing/overriding the required methods `onStartJob()` and `onStopJob()`.

```
public class MyJobService extends JobService
{
    final String TAG = getClass().getSimpleName();

    @Override
    public boolean onStartJob(JobParameters jobParameters) {
        Log.i(TAG, "Job started");

        // ... your code here ...

        jobFinished(jobParameters, false);  // signal that we're done and don't want to reschedule
the job
        return false;                       // finished: no more work to be done
    }

    @Override
    public boolean onStopJob(JobParameters jobParameters) {
        Log.w(TAG, "Job stopped");
        return false;
    }
}
```

Add the new JobService to your AndroidManifest.xml

The following step is *mandatory*, otherwise you won't be able to run your job:

Declare your `MyJobService` class as a new **`<service>`** element between **`<application> </application>`** in your *AndroidManifest.xml*.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".MyJobService"
            android:permission="android.permission.BIND_JOB_SERVICE" />
    </application>
```

```
</manifest>
```

Setup and run the job

After you implemented a new JobService and added it to your *AndroidManifest.xml*, you can continue with the final steps.

- `onButtonClick_startJob()` prepares and runs a periodical job. Besides periodic jobs, `JobInfo.Builder` allows to specify many other settings and constraints. For example you can define that a *plugged in charger* or a *network connection* is required to run the job.
- `onButtonClick_stopJob()` cancels all running jobs

```
public class MainActivity extends AppCompatActivity
{
    final String TAG = getClass().getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onButtonClick_startJob(View v) {
        // get the jobScheduler instance from current context
        JobScheduler jobScheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);

        // MyJobService provides the implementation for the job
        ComponentName jobService = new ComponentName(getApplicationContext(), MyJobService.class);

        // define that the job will run periodically in intervals of 10 seconds
        JobInfo jobInfo = new JobInfo.Builder(1, jobService).setPeriodic(10 * 1000).build();

        // schedule/start the job
        int result = jobScheduler.schedule(jobInfo);
        if (result == JobScheduler.RESULT_SUCCESS)
            Log.d(TAG, "Successfully scheduled job: " + result);
        else
            Log.e(TAG, "RESULT_FAILURE: " + result);
    }

    public void onButtonClick_stopJob(View v) {
        JobScheduler jobScheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);
        Log.d(TAG, "Stopping all jobs...");
        jobScheduler.cancelAll(); // cancel all potentially running jobs
    }
}
```

After calling `onButtonClick_startJob()`, the job will approximately run in intervals of 10 seconds, even when the app is in the *paused* state (user pressed home button and app is no longer visible).

Instead of cancelling all running jobs inside `onButtonClick_stopJob()`, you can also call `jobScheduler.cancel()` to cancel a specific job based on it's job ID.

Chapter 172: Accounts and AccountManager

Section 172.1: Understanding custom accounts/authentication

The following example is high level coverage of the key concepts and basic skeletal setup:

1. Collects credentials from the user (Usually from a login screen you've created)
2. Authenticates the credentials with the server (stores custom authentication)
3. Stores the credentials on the device

Extend an AbstractAccountAuthenticator *(Primarily used to retrieve authentication & re-authenticate them)*

```
public class AccountAuthenticator extends AbstractAccountAuthenticator {

  @Override
  public Bundle addAccount(AccountAuthenticatorResponse response, String accountType,
      String authTokenType, String[] requiredFeatures, Bundle options) {
    //intent to start the login activity
  }


  @Override
  public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account, Bundle
options) {
  }

  @Override
  public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {
  }

  @Override
  public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String
authTokenType,
      Bundle options) throws NetworkErrorException {
    //retrieve authentication tokens from account manager storage or custom storage or re-
authenticate old tokens and return new ones
  }

  @Override
  public String getAuthTokenLabel(String authTokenType) {
  }

  @Override
  public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[]
features)
      throws NetworkErrorException {
    //check whether the account supports certain features
  }

  @Override
  public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account, String
authTokenType,
      Bundle options) {
    //when the user's session has expired or requires their previously available credentials to be
updated, here is the function to do it.
```

```
    }
}
```

Create a service *(Account Manager framework connects to the extended AbstractAccountAuthenticator through the service interface)*

```
public class AuthenticatorService extends Service {

    private AccountAuthenticator authenticator;

    @Override
    public void onCreate(){
        authenticator = new AccountAuthenticator(this);
    }

    @Override
    public IBinder onBind(Intent intent) {
        return authenticator.getIBinder();
    }
}
```

Authenticator XML configuration *(The account manager framework requires. This is what you'll see inside Settings -> Accounts in Android)*

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
   android:accountType="rename.with.your.applicationid"
   android:icon="@drawable/app_icon"
   android:label="@string/app_name"
   android:smallIcon="@drawable/app_icon" />
```

Changes to the AndroidManifest.xml *(Bring all the above concepts together to make it usable programmatically through the AccountManager)*

```
<application
...>
    <service
        android:name=".authenticator.AccountAuthenticatorService"
        android:exported="false"
        android:process=":authentication">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator"/>
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator"/>
    </service>
</application>
```

The next example will contain how to make use of this setup.

Chapter 173: Integrate OpenCV into Android Studio

Section 173.1: Instructions

Tested with A.S. v1.4.1 but should work with newer versions too.

1. Create a new Android Studio project using the project wizard (Menu:/File/New Project):

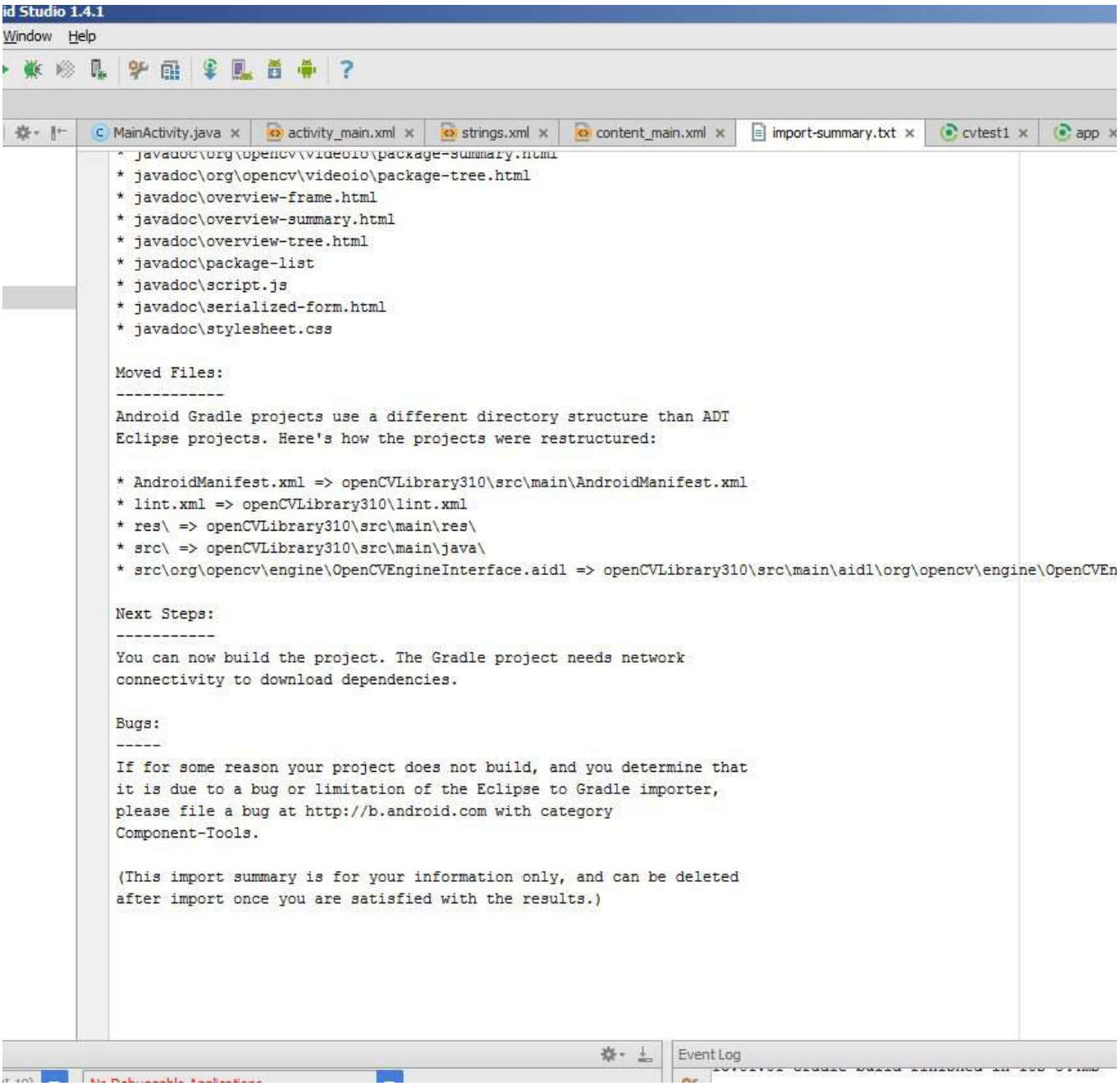
    - Call it "**cvtest1**"
    - Form factor: **API 19, Android 4.4 (KitKat)**
    - **Blank Activity** named **MainActivity**

    You should have a *cvtest1* directory where this project is stored. (the title bar of Android studio shows you where cvtest1 is when you open the project)

2. Verify that your app runs correctly. Try changing something like the "Hello World" text to confirm that the build/test cycle is OK for you. (I'm testing with an emulator of an API 19 device).
3. Download the OpenCV package for Android v3.1.0 and unzip it in some temporary directory somewhere. (Make sure it is the package specifically for Android and not just the OpenCV for Java package.) I'll call this directory "**unzip-dir**" Below **unzip-dir** you should have a **sdk/native/libs** directory with subdirectories that start with things like **arm**..., **mips**... and **x86**... (one for each type of "architecture" Android runs on)
4. From Android Studio import OpenCV into your project as a module: **Menu:/File/New/Import_Module**:

    - Source-directory: **{unzip-dir}/sdk/java**
    - Module name: Android studio automatically fills in this field with **openCVLibrary310** (the exact name probably doesn't matter but we'll go with this).
    - Click on **next**. You get a screen with three checkboxes and questions about jars, libraries and import options. All three should be checked. Click on **Finish.**

    Android Studio starts to import the module and you are shown an **import-summary.txt** file that has a list of what was not imported (mostly javadoc files) and other pieces of information.



But you also get an error message saying **failed to find target with hash string 'android-14'...**. This happens because the build.gradle file in the OpenCV zip file you downloaded says to compile using android API version 14, which by default you don't have with Android Studio v1.4.1.

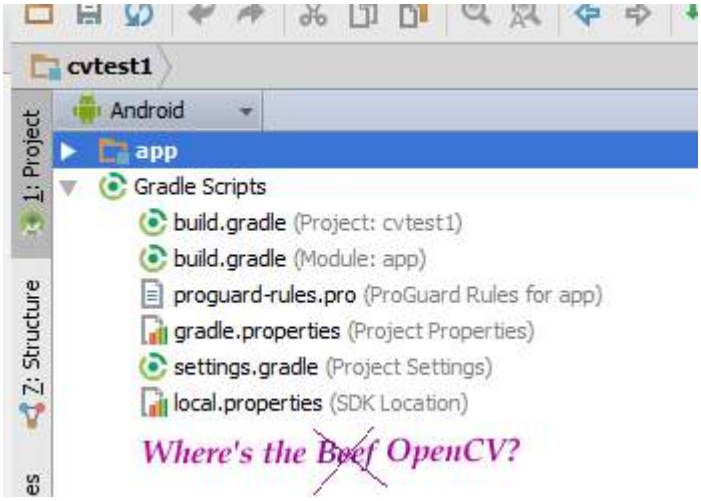
5. Open the project structure dialogue (**Menu:/File/Project_Structure**). Select the "app" module, click on the **Dependencies** tab and add **:openCVLibrary310** as a Module Dependency. When you select **Add/Module_Dependency** it should appear in the list of modules you can add. It will now show up as a dependency but you will get a few more **cannot-find-android-14** errors in the event log.

6. Look in the **build.gradle** file for your app module. There are multiple build.gradle files in an Android project.

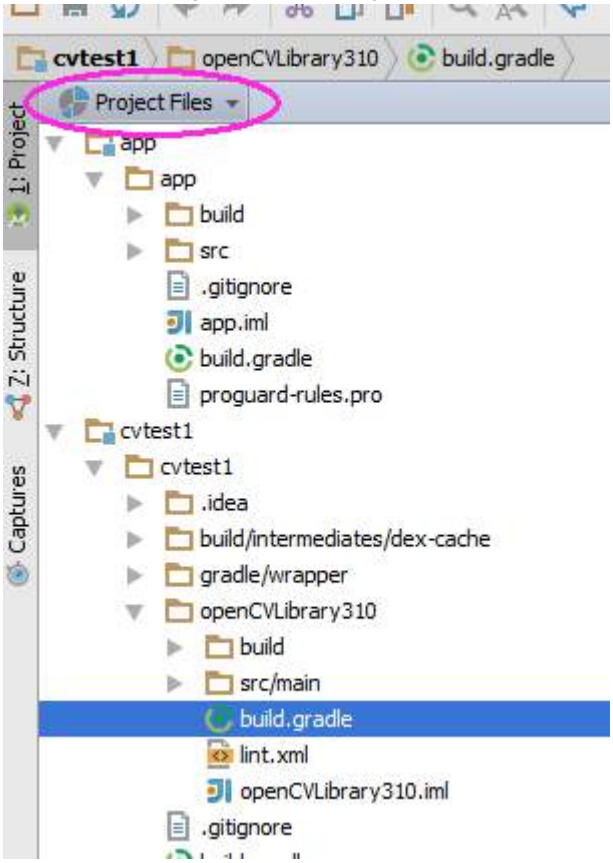
The one you want is in the **cvtest1/app** directory and from the project view it looks like **build.gradle (Module: app)**. Note the values of these four fields:

- compileSDKVersion (mine says 23)
- buildToolsVersion (mine says 23.0.2)
- minSdkVersion (mine says 19)
- targetSdkVersion (mine says 23)

7. Your project now has a **cvtest1/OpenCVLibrary310** directory but it is not visible from the project view:



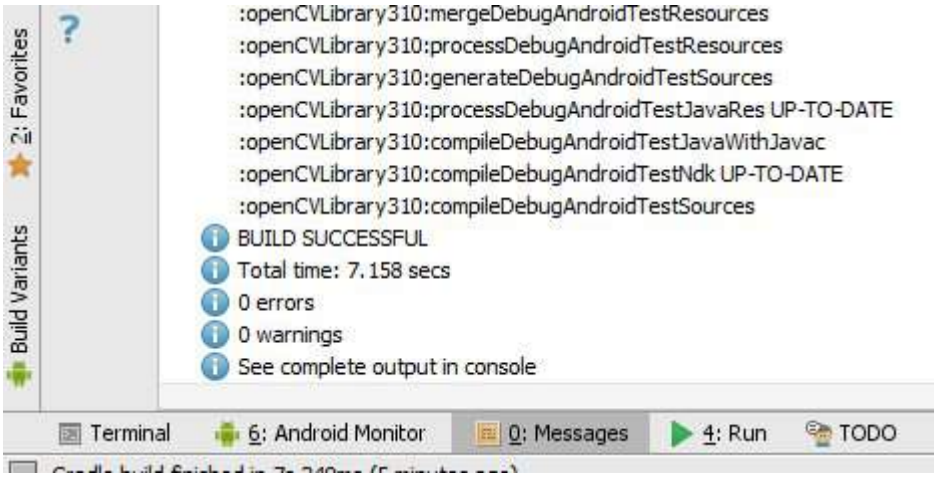
Use some other tool, such as any file manager, and go to this directory. You can also switch the project view from **Android** to **Project Files** and you can find this directory as shown in this screenshot:



Inside there is another **build.gradle** file (it's highlighted in the above screenshot). Update this file with the four values from step 6.

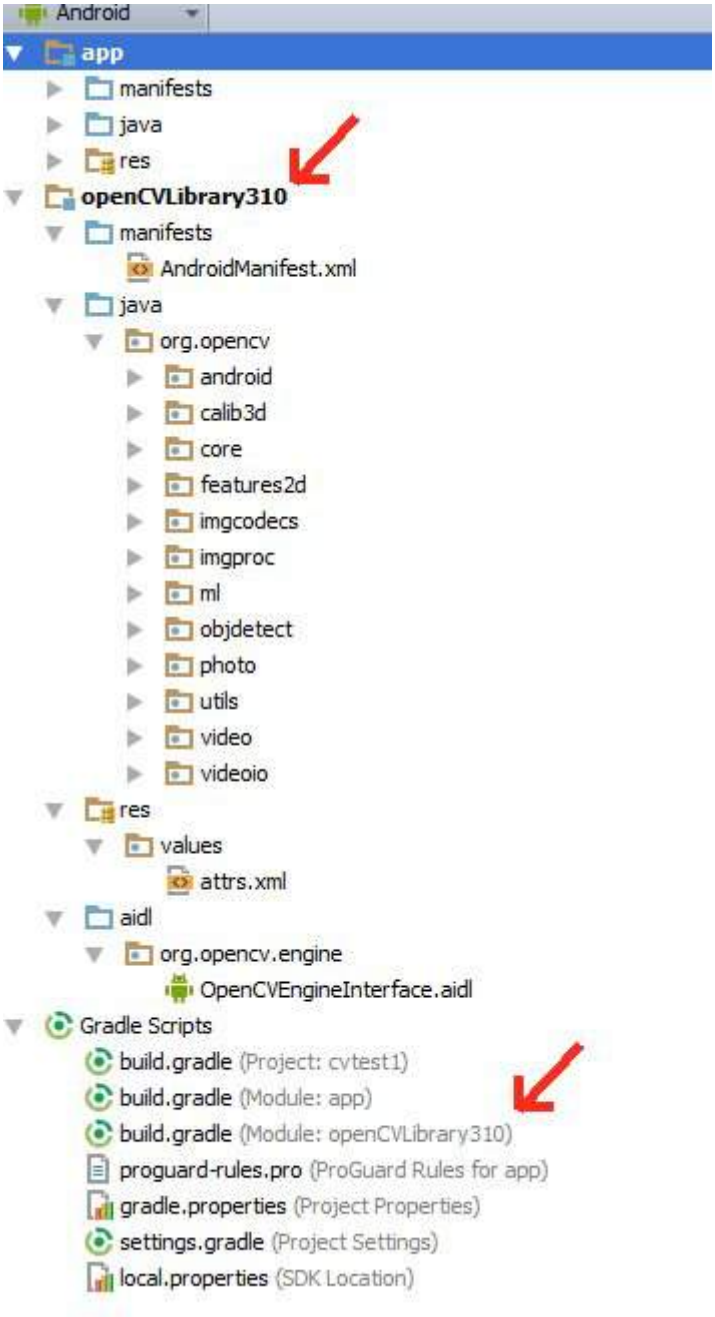
8. Resynch your project and then clean/rebuild it. (**Menu:/Build/Clean_Project**) It should clean and build

without errors and you should see many references to **:openCVLibrary310** in the **0:Messages** screen.

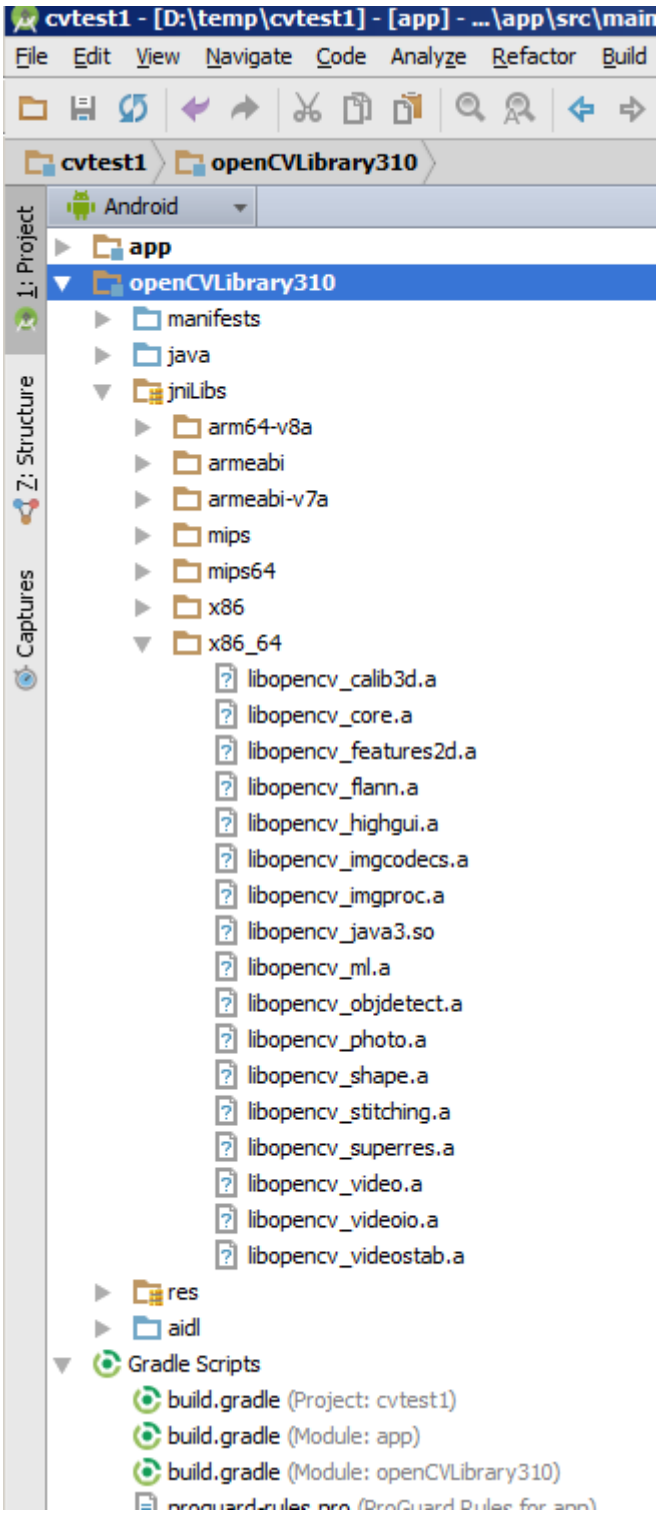


At this point the module should appear in the project hierarchy as **openCVLibrary310**, just like **app**. (Note that in that little drop-down menu I switched back from **Project View** to **Android View**). You should also see an additional **build.gradle** file under "Gradle Scripts" but I find the Android Studio interface a little bit glitchy and sometimes it does not do this right away. So try resynching, cleaning, even restarting Android Studio.

You should see the openCVLibrary310 module with all the OpenCV functions under java like in this screenshot:



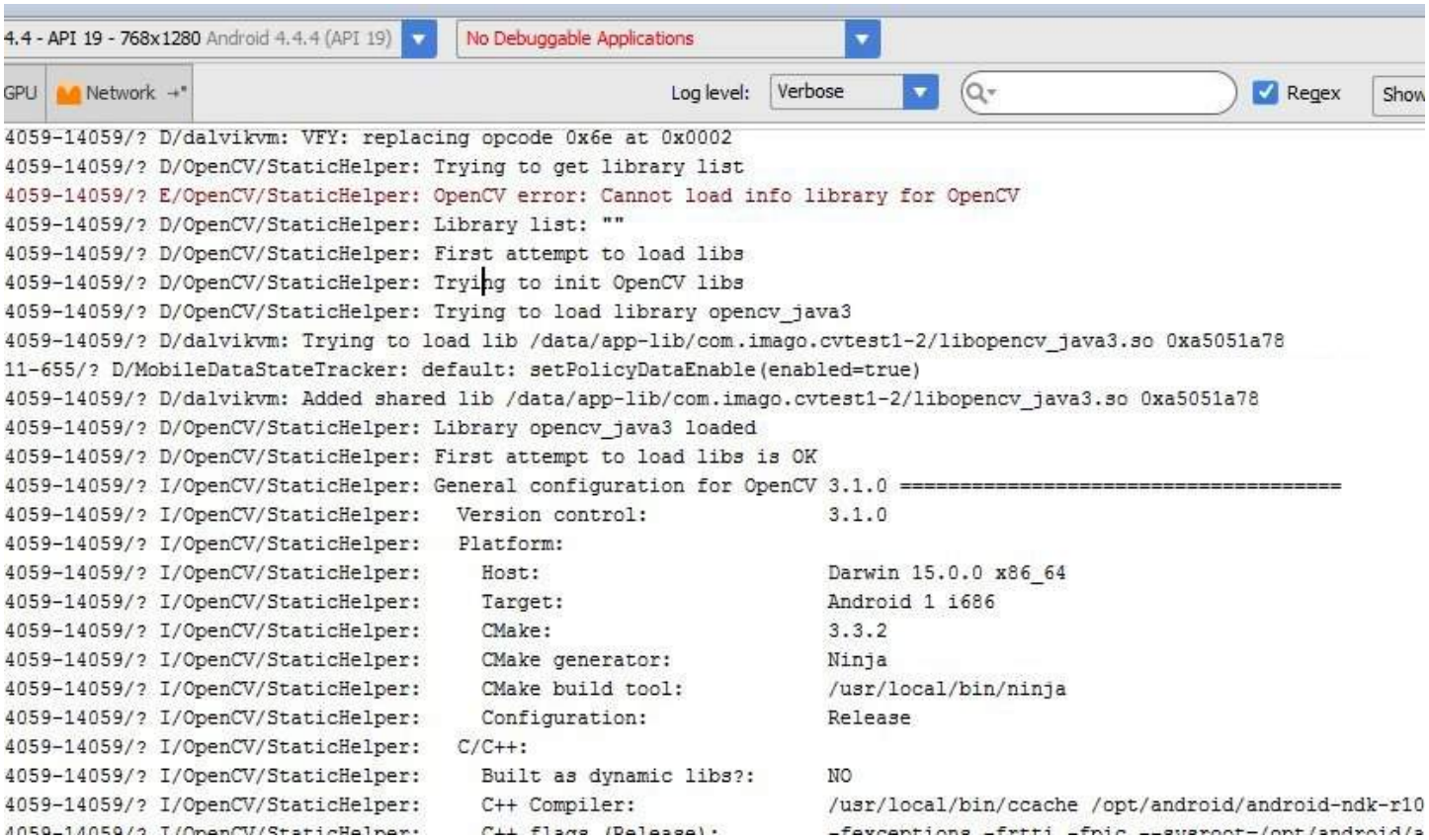
9. Copy the **{unzip-dir}/sdk/native/libs** directory (and everything under it) to your Android project, to **cvtest1/OpenCVLibrary310/src/main/**, and then rename your copy from **libs** to **jniLibs**. You should now have a **cvtest1/OpenCVLibrary310/src/main/jniLibs** directory. Resynch your project and this directory should now appear in the project view under **openCVLibrary310**.



10. Go to the onCreate method of *MainActivity.java* and append this code:

```
if (!OpenCVLoader.initDebug()) {
        Log.e(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), not working.");
    } else {
        Log.d(this.getClass().getSimpleName(), "  OpenCVLoader.initDebug(), working.");
    }
```

Then run your application. You should see lines like this in the Android Monitor:



(I don't know why that line with the error message is there)

11. Now try to actually use some openCV code. In the example below I copied a .jpg file to the cache directory of the cvtest1 application on the android emulator. The code below loads this image, runs the canny edge detection algorithm and then writes the results back to a .png file in the same directory.

Put **this** code just below the code from the previous step and alter it to match your own files/directories.

```
    String inputFileName="simm_01";
    String inputExtension = "jpg";
    String inputDir = getCacheDir().getAbsolutePath();  // use the cache directory for i/o
    String outputDir = getCacheDir().getAbsolutePath();
    String outputExtension = "png";
    String inputFilePath = inputDir + File.separator + inputFileName + "." + inputExtension;


    Log.d (this.getClass().getSimpleName(), "loading " + inputFilePath + "...");
    Mat image = Imgcodecs.imread(inputFilePath);
    Log.d (this.getClass().getSimpleName(), "width of " + inputFileName + ": " + image.width());
    // if width is 0 then it did not read your image.


    // for the canny edge detection algorithm, play with these to see different results
    int threshold1 = 70;
    int threshold2 = 100;

    Mat im_canny = new Mat();  // you have to initialize output image before giving it to the Canny
method
    Imgproc.Canny(image, im_canny, threshold1, threshold2);
    String cannyFilename = outputDir + File.separator + inputFileName + "_canny-" + threshold1 + "-
" + threshold2 + "." + outputExtension;
    Log.d (this.getClass().getSimpleName(), "Writing " + cannyFilename);
    Imgcodecs.imwrite(cannyFilename, im_canny);
```

12. Run your application. Your emulator should create a black and white "edge" image. You can use the Android Device Monitor to retrieve the output or write an activity to show it.

Chapter 174: MVVM (Architecture)

Section 174.1: MVVM Example using DataBinding Library

The whole point of MVVM is to separate layers containing logic from the view layer.

On Android we can use the DataBinding Library to help us with this and make most of our logic Unit-testable without worrying about Android dependencies.

In this example I'll show the central components for a stupid simple App that does the following:

- At start up fake a network call and show a loading spinner
- Show a view with a click counter TextView, a message TextView, and a button to increment the counter
- On button click update counter and update counter color and message text if counter reaches some number

Let's start with the view layer:

`activity_main.xml`:

If you're unfamiliar with how DataBinding works you should probably take 10 minutes to make yourself familiar with it. As you can see, all fields you would usually update with setters are bound to functions on the viewModel variable.

If you've got a question about the `android:visibility` or `app:textColor` properties check the 'Remarks' section.

```
<layout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools">

   <data>

       <import type="android.view.View" />

       <variable
          name="viewModel"
          type="de.walled.mvvmtest.viewmodel.ClickerViewModel"/>
   </data>

   <RelativeLayout
      android:id="@+id/activity_main"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:padding="@dimen/activity_horizontal_margin"

      tools:context="de.walled.mvvmtest.view.MainActivity">

       <LinearLayout
          android:id="@+id/click_counter"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_centerHorizontal="true"
          android:layout_marginTop="60dp"
          android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

          android:padding="8dp"

          android:orientation="horizontal">
```

```
        <TextView
            android:id="@+id/number_of_clicks"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            style="@style/ClickCounter"

            android:text="@{viewModel.numberOfClicks}"
            android:textAlignment="center"
            app:textColor="@{viewModel.counterColor}"

            tools:text="8"
            tools:textColor="@color/red"
        />

        <TextView
            android:id="@+id/static_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            android:layout_marginStart="4dp"
            style="@style/ClickCounter"

            android:text="@string/label.clicks"
            app:textColor="@{viewModel.counterColor}"
            android:textAlignment="center"

            tools:textColor="@color/red"
        />
    </LinearLayout>


    <TextView
        android:id="@+id/message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/click_counter"
        android:layout_centerHorizontal="true"
        android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

        android:text="@{viewModel.labelText}"
        android:textAlignment="center"
        android:textSize="18sp"

        tools:text="You're bad and you should feel bad!"
    />

    <Button
        android:id="@+id/clicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/message"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="8dp"
        android:visibility="@{viewModel.contentVisible ? View.VISIBLE : View.GONE}"

        android:padding="8dp"

        android:text="@string/label.button"

        android:onClick="@{() -> viewModel.onClickIncrement()}"
    />
```

```
        <android.support.v4.widget.ContentLoadingProgressBar
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_marginTop="90dp"
           android:layout_centerHorizontal="true"
           style="@android:style/Widget.ProgressBar.Inverse"
           android:visibility="@{viewModel.loadingVisible ? View.VISIBLE : View.GONE}"

           android:indeterminate="true"
       />

    </RelativeLayout>

</layout>
```

Next the model layer. Here I have:

- two fields that represent the state of the app
- getters to read the number of clicks and the state of excitement
- a method to increment my click count
- a method to restore some previous state (important for orientation changes)

Also I define here a 'state of excitement' that is dependent on the number of clicks. This will later be used to update color and message on the View.

It is important to note that there are no assumptions made in the model about how the state might be displayed to the user!

`ClickerModel.java`

```
import com.google.common.base.Optional;

import de.walled.mvvmtest.viewmodel.ViewState;

public class ClickerModel implements IClickerModel {

    private int numberOfClicks;
    private Excitement stateOfExcitement;

    public void incrementClicks() {
        numberOfClicks += 1;
        updateStateOfExcitement();
    }

    public int getNumberOfClicks() {
        return Optional.fromNullable(numberOfClicks).or(0);
    }

    public Excitement getStateOfExcitement() {
        return Optional.fromNullable(stateOfExcitement).or(Excitement.BOO);
    }

    public void restoreState(ViewState state) {
        numberOfClicks = state.getNumberOfClicks();
        updateStateOfExcitement();
    }

    private void updateStateOfExcitement() {
        if (numberOfClicks < 10) {
            stateOfExcitement = Excitement.BOO;
```

```
        } else if (numberOfClicks <= 20) {
            stateOfExcitement = Excitement.MEH;
        } else {
            stateOfExcitement = Excitement.WOOHOO;
        }
    }
}
```

Next the ViewModel.

This will trigger changes on the model and format data from the model to show them on the view. Note that it is here where we evaluate which GUI representation is appropriate for the state given by the model (`resolveCounterColor` and `resolveLabelText`). So we could for example easily implement an `UnderachieverClickerModel` that has lower thresholds for the state of excitement without touching any code in the viewModel or view.

Also note that the ViewModel does not hold any references to view objects. All properties are bound via the `@Bindable` annotations and updated when either `notifyChange()` (signals all properties need to be updated) or `notifyPropertyChanged(BR.propertyName)` (signals this properties need to be updated).

`ClickerViewModel.java`

```
import android.databinding.BaseObservable;

import android.databinding.Bindable;
import android.support.annotation.ColorRes;
import android.support.annotation.StringRes;

import com.android.databinding.library.baseAdapters.BR;

import de.walled.mvvmtest.R;
import de.walled.mvvmtest.api.IClickerApi;
import de.walled.mvvmtest.model.Excitement;
import de.walled.mvvmtest.model.IClickerModel;
import rx.Observable;

public class ClickerViewModel extends BaseObservable {

    private final IClickerApi api;
    boolean isLoading = false;
    private IClickerModel model;

    public ClickerViewModel(IClickerModel model, IClickerApi api) {
        this.model = model;
        this.api = api;
    }

    public void onClickIncrement() {
        model.incrementClicks();
        notifyChange();
    }

    public ViewState getViewState() {
        ViewState viewState = new ViewState();
        viewState.setNumberOfClicks(model.getNumberOfClicks());
        return viewState;
    }

    public Observable<ViewState> loadData() {
        isLoading = true;
```

```
        return api.fetchInitialState()
                .doOnNext(this::initModel)
                .doOnTerminate(() -> {
                    isLoading = false;
                    notifyPropertyChanged(BR.loadingVisible);
                    notifyPropertyChanged(BR.contentVisible);
                });
    }

    public void initFromSavedState(ViewState savedState) {
        initModel(savedState);
    }

    @Bindable
    public String getNumberOfClicks() {
        final int clicks = model.getNumberOfClicks();
        return String.valueOf(clicks);
    }

    @Bindable
    @StringRes
    public int getLabelText() {
        final Excitement stateOfExcitement = model.getStateOfExcitement();
        return resolveLabelText(stateOfExcitement);
    }

    @Bindable
    @ColorRes
    public int getCounterColor() {
        final Excitement stateOfExcitement = model.getStateOfExcitement();
        return resolveCounterColor(stateOfExcitement);
    }

    @Bindable
    public boolean isLoadingVisible() {
        return isLoading;
    }

    @Bindable
    public boolean isContentVisible() {
        return !isLoading;
    }

    private void initModel(final ViewState viewState) {
        model.restoreState(viewState);
        notifyChange();
    }

    @ColorRes
    private int resolveCounterColor(Excitement stateOfExcitement) {
        switch (stateOfExcitement) {
            case MEH:
                return R.color.yellow;
            case WOOHOO:
                return R.color.green;
            default:
                return R.color.red;
        }
    }

    @StringRes
    private int resolveLabelText(Excitement stateOfExcitement) {
```

```
        switch (stateOfExcitement) {
            case MEH:
                return R.string.label_indifferent;
            case WOOHOO:
                return R.string.label_excited;
            default:
                return R.string.label_negative;
        }
    }

}
```

Tying it all together in the Activity!

Here we see the view initializing the viewModel with all dependencies it might need, that have to be instantiated from an android context.

After the viewModel is initialized it is bound to the xml layout via the DataBindingUtil (Please check 'Syntax' section for naming of generated classes).

Note subscriptions are subscribed to on this layer because we have to handle unsubscribing them when the activity is paused or destroyed to avoid memory leaks and NPEs. Also persisting and reloading of the viewState on OrientationChanges is triggered here

`MainActivity.java`

```
import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

import de.walled.mvvmtest.R;
import de.walled.mvvmtest.api.ClickerApi;
import de.walled.mvvmtest.api.IClickerApi;
import de.walled.mvvmtest.databinding.ActivityMainBinding;
import de.walled.mvvmtest.model.ClickerModel;
import de.walled.mvvmtest.viewmodel.ClickerViewModel;
import de.walled.mvvmtest.viewmodel.ViewState;
import rx.Subscription;
import rx.subscriptions.Subscriptions;

public class MainActivity extends AppCompatActivity {

    private static final String KEY_VIEW_STATE = "state.view";

    private ClickerViewModel viewModel;
    private Subscription fakeLoader = Subscriptions.unsubscribed();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // would usually be injected but I feel Dagger would be out of scope
        final IClickerApi api = new ClickerApi();
        setupViewModel(savedInstanceState, api);

        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);
        binding.setViewModel(viewModel);
    }

    @Override
```

```
    protected void onPause() {
        fakeLoader.unsubscribe();
        super.onPause();
    }

    @Override
    protected void onDestroy() {
        fakeLoader.unsubscribe();
        super.onDestroy();
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putSerializable(KEY_VIEW_STATE, viewModel.getViewState());
    }

    private void setupViewModel(Bundle savedInstance, IClickerApi api) {
        viewModel = new ClickerViewModel(new ClickerModel(), api);
        final ViewState savedState = getViewStateFromBundle(savedInstance);

        if (savedState == null) {
            fakeLoader = viewModel.loadData().subscribe();
        } else {
            viewModel.initFromSavedState(savedState);
        }
    }

    private ViewState getViewStateFromBundle(Bundle savedInstance) {
        if (savedInstance != null) {
            return (ViewState) savedInstance.getSerializable(KEY_VIEW_STATE);
        }
        return null;
    }
}
```

To see everything in action check out this example project.

Chapter 175: ORMLite in android

Section 175.1: Android OrmLite over SQLite example

ORMLite is an Object Relational Mapping package that provides simple and lightweight functionality for persisting Java objects to SQL databases while avoiding the complexity and overhead of more standard ORM packages.

Speaking for Android, OrmLite is implemented over the out-of-the-box supported database, SQLite. It makes direct calls to the API to access SQLite.

Gradle setup

To get started you should include the package to the build gradle.

```
 // https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
POJO configuration
```

Then you should configure a POJO to be persisted to the database. Here care must be taken to the annotations:

- Add the @DatabaseTable annotation to the top of each class. You can also use @Entity.
- Add the @DatabaseField annotation right before each field to be persisted. You can also use @Column and others.
- Add a no-argument constructor to each class with at least package visibility.

```
@DatabaseTable(tableName = "form_model")
public class FormModel implements Serializable {

    @DatabaseField(generatedId = true)
    private Long id;
    @DatabaseField(dataType = DataType.SERIALIZABLE)
    ArrayList<ReviewItem> reviewItems;

    @DatabaseField(index = true)
    private String username;

    @DatabaseField
    private String createdAt;

    public FormModel() {
    }

    public FormModel(ArrayList<ReviewItem> reviewItems, String username, String createdAt) {
        this.reviewItems = reviewItems;
        this.username = username;
        this.createdAt = createdAt;
    }
}
```

At the example above there is one table (form_model) with 4 fields.

id field is auto generated index.

username is an index to the database.

More information about the annotation can be found at the official documentation.

Database Helper

To continue with, you will need to create a database helper class which should extend the OrmLiteSqliteOpenHelper class.

This class creates and upgrades the database when your application is installed and can also provide the DAO classes used by your other classes.

DAO stands for Data Access Object and it provides all the scrum functionality and specializes in the handling a single persisted class.

The helper class must implement the following two methods:

- onCreate(SQLiteDatabase sqliteDatabase, ConnectionSource connectionSource);

  onCreate creates the database when your app is first installed

- onUpgrade(SQLiteDatabase database, ConnectionSource connectionSource, int oldVersion, int newVersion);

  onUpgrade handles the upgrading of the database tables when you upgrade your app to a new version

Database Helper class example:

```
public class OrmLite extends OrmLiteSqliteOpenHelper {

      //Database name
      private static final String DATABASE_NAME = "gaia";
      //Version of the database. Changing the version will call {@Link OrmLite.onUpgrade}
      private static final int DATABASE_VERSION = 2;

      /**
       * The data access object used to interact with the Sqlite database to do C.R.U.D operations.
       */
      private Dao<FormModel, Long> todoDao;



      public OrmLite(Context context) {
          super(context, DATABASE_NAME, null, DATABASE_VERSION,
                  /**
                   * R.raw.ormlite_config is a reference to the ormlite_config2.txt file in the
                   * /res/raw/ directory of this project
                   * */
                  R.raw.ormlite_config2);
      }

      @Override
      public void onCreate(SQLiteDatabase database, ConnectionSource connectionSource) {
          try {

              /**
               * creates the database table
               */
              TableUtils.createTable(connectionSource, FormModel.class);

          } catch (SQLException e) {
```

```
                e.printStackTrace();
            } catch (java.sql.SQLException e) {
                e.printStackTrace();
            }
        }
        /*
            It is called when you construct a SQLiteOpenHelper with version newer than the version of
the opened database.
         */
        @Override
        public void onUpgrade(SQLiteDatabase database, ConnectionSource connectionSource,
                              int oldVersion, int newVersion) {
            try {
                /**
                 * Recreates the database when onUpgrade is called by the framework
                 */
                TableUtils.dropTable(connectionSource, FormModel.class, false);
                onCreate(database, connectionSource);

            } catch (SQLException | java.sql.SQLException e) {
                e.printStackTrace();
            }
        }

        /**
         * Returns an instance of the data access object
         * @return
         * @throws SQLException
         */
        public Dao<FormModel, Long> getDao() throws SQLException {
            if(todoDao == null) {
                try {
                    todoDao = getDao(FormModel.class);
                } catch (java.sql.SQLException e) {
                    e.printStackTrace();
                }
            }
            return todoDao;
        }
    }
```

Persisting Object to SQLite

Finally, the class that persists the object to the database.

```
     public class ReviewPresenter {
    Dao<FormModel, Long> simpleDao;


    public ReviewPresenter(Application application) {
        this.application = (GaiaApplication) application;
        simpleDao = this.application.getHelper().getDao();
    }

    public void storeFormToSqLite(FormModel form) {

        try {
            simpleDao.create(form);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        List<FormModel> list = null;
```

```
        try {
// query for all of the data objects in the database
            list = simpleDao.queryForAll();
        } catch (SQLException e) {
            e.printStackTrace();
        }
// our string builder for building the content-view
        StringBuilder sb = new StringBuilder();
        int simpleC = 1;
        for (FormModel simple : list) {
            sb.append('#').append(simpleC).append(": ").append(simple.getUsername()).append('\n');
            simpleC++;
        }
        System.out.println(sb.toString());
    }

    //Query to database to get all forms by username
    public List<FormModel> getAllFormsByUsername(String username) {
        List<FormModel> results = null;
        try {
            results = simpleDao.queryBuilder().where().eq("username",
PreferencesManager.getInstance().getString(Constants.USERNAME)).query();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return results;
    }
}
```

The accessor of the DOA at the constructor of the above class is defined as:

```
 private OrmLite dbHelper = null;

/*
Provides the SQLite Helper Object among the application
 */
public OrmLite getHelper() {
    if (dbHelper == null) {
        dbHelper = OpenHelperManager.getHelper(this, OrmLite.class);
    }
    return dbHelper;
}
```

Chapter 176: Retrofit2 with RxJava

Section 176.1: Retrofit2 with RxJava

First, add relevant dependencies into the build.gradle file.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.3.0'
    compile 'com.squareup.retrofit2:converter-gson:2.3.0'
    compile 'com.squareup.retrofit2:adapter-rxjava:2.3.0'
    ....
}
```

Then create the model you would like to receive:

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

Create an interface containing methods used to exchange data with remote server:

```
public interface ApiServerRequests {

    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

Then create a `Retrofit` instance:

```
public ApiRequests DeviceAPIHelper ()
{
    Gson gson = new GsonBuilder().create();

    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("http://example.com/")
            .addConverterFactory(GsonConverterFactory.create(gson))
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .build();

    api = retrofit.create(ApiServerRequests.class);
    return api;
}
```

Then, anywhere from the code, call the method:

```
apiRequests.getServers()
   .subscribeOn(Schedulers.io()) // the observable is emitted on io thread
   .observerOn(AndroidSchedulers.mainThread()) // Methods needed to handle request in background
thread
   .subscribe(new Subscriber<List<Server>>() {
       @Override
       public void onCompleted() {
```

```
        }

        @Override
        public void onError(Throwable e) {

        }

        @Override
        public void onNext(List<Server> servers) {
            //A list of servers is fetched successfully
        }
    });
```

Section 176.2: Nested requests example: multiple requests, combine results

Suppose we have an API which allows us to get object metadata in single request (`getAllPets`), and other request which have full data of single resource (`getSinglePet`). How we can query all of them in a single chain?

```
public class PetsFetcher {


static class PetRepository {
    List<Integer> ids;
}

static class Pet {
    int id;
    String name;
    int weight;
    int height;
}

interface PetApi {

    @GET("pets") Observable<PetRepository> getAllPets();

    @GET("pet/{id}") Observable<Pet> getSinglePet(@Path("id") int id);

}

PetApi petApi;

Disposable petsDisposable;

public void requestAllPets() {

    petApi.getAllPets()
            .doOnSubscribe(new Consumer<Disposable>() {
                @Override public void accept(Disposable disposable) throws Exception {
                    petsDisposable = disposable;
                }
            })
            .flatMap(new Function<PetRepository, ObservableSource<Integer>>() {
                @Override
                public ObservableSource<Integer> apply(PetRepository petRepository) throws
Exception {
                    List<Integer> petIds = petRepository.ids;
                    return Observable.fromIterable(petIds);
                }
```

```
            })
            .flatMap(new Function<Integer, ObservableSource<Pet>>() {
                @Override public ObservableSource<Pet> apply(Integer id) throws Exception {
                    return petApi.getSinglePet(id);
                }
            })
            .toList()
            .toObservable()
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(new Consumer<List<Pet>>() {
                @Override public void accept(List<Pet> pets) throws Exception {
                    //use your pets here
                }
            }, new Consumer<Throwable>() {
                @Override public void accept(Throwable throwable) throws Exception {
                    //show user something goes wrong
                }
            });

}

void cancelRequests(){
    if (petsDisposable!=null){
        petsDisposable.dispose();
        petsDisposable = null;
    }
}
```

}

Section 176.3: Retrofit with RxJava to fetch data asyncronously

From the GitHub repo of RxJava, *RxJava is a Java VM implementation of Reactive Extensions: a library for composing asynchronous and event-based programs by using observable sequences. It extends the observer pattern to support sequences of data/events and adds operators that allow you to compose sequences together declaratively while abstracting away concerns about things like low-level threading, synchronisation, thread-safety and concurrent data structures.*

Retrofit is a type-safe HTTP client for Android and Java, using this, developers can make all network stuff much more easier. As an example, we are going to download some JSON and show it in RecyclerView as a list.

Getting started:

Add RxJava, RxAndroid and Retrofit dependencies in your app level build.gradle file: `compile "io.reactivex:rxjava:1.1.6"`

```
compile "io.reactivex:rxandroid:1.2.1"
compile "com.squareup.retrofit2:adapter-rxjava:2.0.2"
compile "com.google.code.gson:gson:2.6.2"
compile "com.squareup.retrofit2:retrofit:2.0.2"
compile "com.squareup.retrofit2:converter-gson:2.0.2"
```

Define ApiClient and ApiInterface to exchange data from server

```
public class ApiClient {

private static Retrofit retrofitInstance = null;
private static final String BASE_URL = "https://api.github.com/";
```

```
public static Retrofit getInstance() {
    if (retrofitInstance == null) {
        retrofitInstance = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
                .addConverterFactory(GsonConverterFactory.create())
                .build();
    }
    return retrofitInstance;
}

public static <T> T createRetrofitService(final Class<T> clazz, final String endPoint) {
    final Retrofit restAdapter = new Retrofit.Builder()
            .baseUrl(endPoint)
            .build();

    return restAdapter.create(clazz);
}

public static String getBaseUrl() {
    return BASE_URL;
}}
```

public interface ApiInterface {

```
@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                  @Path("repo") String repositoryName,
                                  @Query("page") int pageNumber);}
```

Note the getRepos() is returning an Observable and not just a list of issues.

Define the models

An example for this is shown. You can use free services like JsonSchema2Pojo or this.

```
public class Comment {

@SerializedName("url")
@Expose
private String url;
@SerializedName("html_url")
@Expose
private String htmlUrl;

//Getters and Setters
}
```

Create Retrofit instance

```
ApiInterface apiService = ApiClient.getInstance().create(ApiInterface.class);
```

Then, Use this instance to fetch data from server

```
Observable<List<Issue>> issueObservable = apiService.getIssues(org, repo,
pageNumber);
    issueObservable.subscribeOn(Schedulers.newThread())
            .observeOn(AndroidSchedulers.mainThread())
            .map(issues -> issues)    //get issues and map to issues list
            .subscribe(new Subscriber<List<Issue>>() {
```

```
                @Override
                public void onCompleted() {
                    Log.i(TAG, "onCompleted: COMPLETED!");
                }

                @Override
                public void onError(Throwable e) {
                    Log.e(TAG, "onError: ", e);
                }

                @Override
                public void onNext(List<Issue> issues) {
                    recyclerView.setAdapter(new IssueAdapter(MainActivity.this, issues,
apiService));
                }
            });
```

Now, you have successfully fetched data from a server using Retrofit and RxJava.

Chapter 177: ShortcutManager

Section 177.1: Dynamic Launcher Shortcuts

```
ShortcutManager shortcutManager = getSystemService(ShortcutManager.class);

ShortcutInfo shortcut = new ShortcutInfo.Builder(this, "id1")
    .setShortLabel("Web site") // Shortcut Icon tab
    .setLongLabel("Open the web site") // Displayed When Long Pressing On App Icon
    .setIcon(Icon.createWithResource(context, R.drawable.icon_website))
    .setIntent(new Intent(Intent.ACTION_VIEW,
                   Uri.parse("https://www.mysite.example.com/")))
    .build();

shortcutManager.setDynamicShortcuts(Arrays.asList(shortcut));
```

We can remove all dynamic shortcuts easily by calling:

```
shortcutManager.removeAllDynamicShortcuts();
```

We can update existing Dynamic Shorcuts by Using

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut);
```

Please note that `setDynamicShortcuts(List)`is used to redefine the entire list of dynamic shortcuts, `addDynamicShortcuts(List)` is used to add dynamic shortcuts to existing list of dynamic shortcuts

Chapter 178: LruCache

Section 178.1: Adding a Bitmap(Resource) to the cache

To add a resource to the cache you must provide a key and the resource. First make sure that the value is not in the cache already

```
public void addResourceToMemoryCache(String key, Bitmap resource) {
    if (memoryCache.get(key) == null)
        memoryCache.put(key, resource);
}
```

Section 178.2: Initialising the cache

The Lru Cache will store all the added resources (values) for fast access until it reaches a memory limit, in which case it will drop the less used resource (value) to store the new one.

To initialise the Lru cache you need to provide a maximum memory value. This value depends on your application requirements and in how critical the resource is to keep a smooth app usage. A recommended value for an image gallery, for example, would be 1/8 of your maximum available memory.

Also note that the Lru Cache works on a key-value basis. In the following example, the key is a `String` and the value is a `Bitmap`:

```
int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);
int cacheSize = maxMemory / 8;

LruCache<String, Bitmap> = memoryCache = new LruCache<String, Bitmap>(cacheSize) {
    protected int sizeOf(String key, Bitmap bitmap) {
        return bitmap.getByteCount();
    }
};
```

Section 178.3: Getting a Bitmap(Resouce) from the cache

To get a resource from the cache simply pass the key of your resource (String in this example)

```
public Bitmap getResourceFromMemoryCache(String key) {
    memoryCache.get(key);
}
```

Chapter 179: Jenkins CI setup for Android Projects

Section 179.1: Step by step approach to set up Jenkins for Android

This is a step by step guide to set up the automated build process using Jenkins CI for your Android projects. The following steps assume that you have new hardware with just any flavor of Linux installed. It is also taken into account that you might have a remote machine.

PART I: Initial setup on your machine

1. Log in via *ssh* to your Ubuntu machine:

```
ssh username@xxx.xxx.xxx
```

2. Download a version of the Android SDK on your machine:

```
wget https://dl.google.com/android/android-sdk_r24.4.1-linux.tgz
```

3. Unzip the downloaded *tar* file:

```
sudo apt-get install tar
tar -xvf android-sdk_r24.4.1-linux.tgz
```

4. Now you need to install Java 8 on your Ubuntu machine, which is a requirement for Android builds on Nougat. Jenkins would require you to install JDK and JRE 7 using the steps below:

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
apt-get install openjdk-8-jdk
```

5. Now install Jenkins on your Ubuntu machine:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins
```

6. Download the latest supported Gradle version for your Android setup:

```
wget https://services.gradle.org/distributions/gradle-2.14.1-all.zip
unzip gradle-2.14.1-all.zip
```

7. Set up Android on your Ubuntu machine. First move to the *tools* folder in the Android SDK folder downloaded in step 2:

```
cd android-sdk-linux/tools // lists available SDK
android update sdk --no-ui // Updates SDK version
android list sdk -a | grep "SDK Build-tools" // lists available build tools
android update sdk -a -u -t 4 // updates build tools version to one listed as 4 by prev. cmd.
update java
```

8. Install *Git* or any other VCS on your machine:

```
sudo apt-get install git
```

9. Now log in to Jenkins using your internet browser. Type `ipAddress:8080` into the address bar.
10. In order to receive the password for the first-time login, please check the corresponding file as follows (you will need su permissions to access this file):

```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

PART II: Set up Jenkins to build Android Jobs

1. Once logged in, go to the following path:

```
Jenkins > Manage Jenkins > Global Tool Configuration
```

2. At this location, add `JAVA_HOME` with the following entries:

```
Name = JAVA_HOME
JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64
```

3. Also add the following values to *Git* and save the environment variables:

> Name = Default
> /usr/bin/git

4. Now go to the following path:

> Jenkins > Manage Jenkins > Configuration

5. At this location, add `ANDROID_HOME` to the "global properties":

> Name = ANDROID_HOME
> Value = /home/username/android-sdk-linux

Part III: Create a Jenkins Job for your Android project

1. Click on *New Item* in the Jenkins home screen.
2. Add a *Project Name* and *Description*.
3. In the *General* tab, select *Advanced*. Then select *Use custom workspace*:

> Directory /home/user/Code/ProjectFolder

4. In the source code management select *Git*. I am using *Bitbucket* for the purpose of this example:

> Repository URL = https://username:password@bitbucket.org/project/projectname.git

5. Select additional behaviors for your repository:

> Clean Before Checkout
> Checkout to a sub-directory. Local subdirectory for repo /home/user/Code/ProjectFolder

6. Select a branch you want to build:

> */master

7. In the *Build* tab, select *Execute Shell* in *Add build step*.
8. In the *Execute shell*, add the following command:

```
cd /home/user/Code/ProjectFolder && gradle clean assemble --no-daemon
```

9. If you want to run Lint on the project, then add another build step into the *Execute shell*:

```
/home/user/gradle/gradle-2.14.1/bin/gradle lint
```

Now your system is finally set up to build Android projects using Jenkins. This setup makes your life so much easier for releasing builds to QA and UAT teams.

PS: Since Jenkins is a different user on your Ubuntu machine, you should give it rights to create folders in your workspace by executing the following command:

```
chown -R jenkins .git
```

Chapter 180: fastlane

Section 180.1: Fastfile lane to build and install all flavors for given build type to a device

Add this lane to your **Fastfile** and run `fastlane installAll type:{BUILD_TYPE}` in command line. Replace `BUILD_TYPE` with the build type you want to build.

For example: `fastlane installAll type:Debug`

This command will build all flavors of given type and install it to your device. Currently, it doesn't work if you have more than one device attached. Make sure you have only one. In the future I'm planning to add option to select target device.

```
lane :installAll do |options|

    gradle(task: "clean")

    gradle(task: "assemble",
       build_type: options[:type])

    lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do | apk |

        puts "Uploading APK to Device: " + apk

        begin
            adb(
                command: "install -r #{apk}"
            )
        rescue => ex
            puts ex
        end
    end
end
```

Section 180.2: Fastfile to build and upload multiple flavors to Beta by Crashlytics

This is a sample **Fastfile** setup for a multi-flavor app. It gives you an option to build and deploy all flavors or a single flavor. After the deployment, it reports to **Slack** the status of the deployment, and sends a notification to testers in Beta by Crashlytics testers group.

To build and deploy all flavors use:

```
fastlane android beta
```

To build a single APK and deploy use:

```
fastlane android beta app:flavorName
```

Using a single Fastlane file, you can manage iOS, Android, and Mac apps. If you are using this file just for one app `platform` is not required.

How It Works

1. `android` argument tells fastlane that we will use `:android` platform.
2. Inside `:android` platform you can have multiple lanes. Currently, I have only `:beta` lane. The second argument from the command above specifies the lane we want to use.
3. `options[:app]`
4. There are two **Gradle** tasks. First, it runs `gradle clean`. If you provided a flavor with `app` key, fastfile runs `gradle assembleReleaseFlavor`. Otherwise, it runs `gradle assembleRelease` to build all build flavors.
5. If we are building for all flavors, an array of generated APK file names is stored inside `SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS`. We use this to loop through generated files and deploy them to **Beta by Crashlytics**. `notifications` and `groups` fields are optional. They are used to notify testers registered for the app on **Beta by Crashlytics**.
6. If you are familiar with Crashlytics, you might know that to activate an app in the portal, you have to run it on a device and use it first. Otherwise, Crashlytics will assume the app inactive and throw an error. In this scenario, I capture it and report to **Slack** as a failure, so you will know which app is inactive.
7. If deployment is successful, **fastlane** will send a success message to **Slack**.
8. `#{/([^\/]*)$/.match(apk)}` this regex is used to get flavor name from APK path. You can remove it if it does not work for you.
9. `get_version_name` and `get_version_code` are two **Fastlane** plugins to retrieve app version name and code. You have to install these gems if you want to use, or you can remove them. Read more about Plugins here.
10. The **`else`** statement will be executed if you are building and deploying a single APK. We don't have to provide `apk_path` to Crashlytics since we have only one app.
11. `error` **`do`** block at the end is used to get notified if anything else goes wrong during execution.

Note

Don't forget to replace `SLACK_URL`, `API_TOKEN`, `GROUP_NAME` and `BUILD_SECRET` with your own credentials.

```
fastlane_version "1.46.1"

default_platform :android

platform :android do

    before_all do
        ENV["SLACK_URL"] = "https://hooks.slack.com/servic...."
    end

    lane :beta do |options|
        # Clean and build the Release version of the app.
        # Usage `fastlane android beta app:flavorName`

        gradle(task: "clean")

        gradle(task: "assemble",
               build_type: "Release",
               flavor: options[:app])

        # If user calls `fastlane android beta` command, it will build all projects and push them
to Crashlytics
        if options[:app].nil?
            lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].each do | apk |

                puts "Uploading APK to Crashlytics: " + apk

                begin
                    crashlytics(
                      api_token: "[API_TOKEN]",
                      build_secret: "[BUILD_SECRET]",
```

```
                    groups: "[GROUP_NAME]",
                    apk_path: apk,
                    notifications: "true"
                )

                slack(
                  message: "Successfully deployed new build for #{/([^\/]*)$/.match(apk)}
#{get_version_name} - #{get_version_code}",
                  success: true,
                  default_payloads: [:git_branch, :lane, :test_result]
                )
            rescue => ex
                # If the app is inactive in Crashlytics, deployment will fail. Handle it here
and report to slack
                slack(
                    message: "Error uploading => #{/([^\/]*)$/.match(apk)} #{get_version_name}
- #{get_version_code}: #{ex}",
                    success: false,
                    default_payloads: [:git_branch, :lane, :test_result]
                )
            end
        end

        after_all do |lane|
            # This block is called, only if the executed lane was successful
            slack(
                message: "Operation completed for
#{lane_context[SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS].size} app(s) for #{get_version_name} -
#{get_version_code}",
                default_payloads: [:git_branch, :lane, :test_result],
                success: true
            )
        end
    else
        # Single APK upload to Beta by Crashlytics
        crashlytics(
            api_token: "[API_TOKEN]",
            build_secret: "[BUILD_SECRET]",
            groups: "[GROUP_NAME]",
            notifications: "true"
        )

        after_all do |lane|
            # This block is called, only if the executed lane was successful
            slack(
                message: "Successfully deployed new build for #{options[:app]}
#{get_version_name} - #{get_version_code}",
                default_payloads: [:git_branch, :lane, :test_result],
                success: true
            )
        end
    end

    error do |lane, exception|
        slack(
            message: exception.message,
            success: false,
            default_payloads: [:git_branch, :lane, :test_result]
        )
    end
  end
end
```

Chapter 181: Define step value (increment) for custom RangeSeekBar

A customization of the Android RangeSeekBar proposed by Alex Florescu at https://github.com/anothem/android-range-seek-bar

It allows to define a step value (increment), when moving the seek bar

Section 181.1: Define a step value of 7

```
<RangeSeekBar
        android:id="@+id/barPrice"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        app:barHeight="0.2dp"
        app:barHeight2="4dp"
        app:increment="7"
        app:showLabels="false" />
```

Chapter 182: Getting started with OpenGL ES 2.0+

This topic is about setting up and using **OpenGL ES 2.0+** on Android. OpenGL ES is the standard for 2D and 3D accelerated graphics on embedded systems - including consoles, smartphones, appliances and vehicles.

Section 182.1: Setting up GLSurfaceView and OpenGL ES 2.0+

To use OpenGL ES in your application you must add this to the manifest:

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

Create your extended GLSurfaceView:

```
import static android.opengl.GLES20.*; // To use all OpenGL ES 2.0 methods and constants statically

public class MyGLSurfaceView extends GLSurfaceView {

    public MyGLSurfaceView(Context context, AttributeSet attrs) {
        super(context, attrs);

        setEGLContextClientVersion(2); // OpenGL ES version 2.0
        setRenderer(new MyRenderer());
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
    }

    public final class MyRenderer implements GLSurfaceView.Renderer{
        public final void onSurfaceCreated(GL10 unused, EGLConfig config) {
            // Your OpenGL ES init methods
            glClearColor(1f, 0f, 0f, 1f);
        }
        public final void onSurfaceChanged(GL10 unused, int width, int height) {
            glViewport(0, 0, width, height);
        }

        public final void onDrawFrame(GL10 unused) {
            // Your OpenGL ES draw methods
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        }
    }
}
```

Add `MyGLSurfaceView` to your layout:

```
<com.example.app.MyGLSurfaceView
   android:id="@+id/gles_renderer"
   android:layout_width="match_parent"
   android:layout_height="match_parent"/>
```

To use newer version of OpenGL ES just change the version number in your manifest, in the static import and change `setEGLContextClientVersion`.

Section 182.2: Compiling and Linking GLSL-ES Shaders from

asset file

The Assets folder is the most common place to store your GLSL-ES shader files. To use them in your OpenGL ES application you need to load them to a string in the first place. This functions creates a string from the asset file:

```
private String loadStringFromAssetFile(Context myContext, String filePath){
    StringBuilder shaderSource = new StringBuilder();
    try {
        BufferedReader reader = new BufferedReader(new
InputStreamReader(myContext.getAssets().open(filePath)));
        String line;
        while((line = reader.readLine()) != null){
            shaderSource.append(line).append("\n");
        }
        reader.close();
        return shaderSource.toString();
    } catch (IOException e) {
        e.printStackTrace();
        Log.e(TAG, "Could not load shader file");
        return null;
    }
}
```

Now you need to create a function that compiles a shader stored in a sting:

```
private int compileShader(int shader_type, String shaderString){

    // This compiles the shader from the string
    int shader = glCreateShader(shader_type);
    glShaderSource(shader, shaderString);
    glCompileShader(shader);

    // This checks for for compilation errors
    int[] compiled = new int[1];
    glGetShaderiv(shader, GL_COMPILE_STATUS, compiled, 0);
    if (compiled[0] == 0) {
        String log = glGetShaderInfoLog(shader);

        Log.e(TAG, "Shader compilation error: ");
        Log.e(TAG, log);
    }
    return shader;
}
```

Now you can load, compile and link your shaders:

```
// Load shaders from file
String vertexShaderString = loadStringFromAssetFile(context, "your_vertex_shader.glsl");
String fragmentShaderString = loadStringFromAssetFile(context, "your_fragment_shader.glsl");

// Compile shaders
int vertexShader = compileShader(GL_VERTEX_SHADER, vertexShaderString);
int fragmentShader = compileShader(GL_FRAGMENT_SHADER, fragmentShaderString);

// Link shaders and create shader program
int shaderProgram = glCreateProgram();
glAttachShader(shaderProgram , vertexShader);
glAttachShader(shaderProgram , fragmentShader);
glLinkProgram(shaderProgram);
```

```
// Check for linking errors:
int linkStatus[] = new int[1];
glGetProgramiv(shaderProgram, GL_LINK_STATUS, linkStatus, 0);
if (linkStatus[0] != GL_TRUE) {
    String log = glGetProgramInfoLog(shaderProgram);

    Log.e(TAG,"Could not link shader program: ");
    Log.e(TAG, log);
}
```

If there are no errors, your shader program is ready to use:

```
glUseProgram(shaderProgram);
```

Chapter 183: Check Data Connection

Section 183.1: Check data connection

This method is to check data connection by ping certain IP or Domain name.

```
public Boolean isDataConnected() {
    try {
        Process p1 = java.lang.Runtime.getRuntime().exec("ping -c 1 8.8.8.8");
        int returnVal = p1.waitFor();
        boolean reachable = (returnVal==0);
        return reachable;
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return false;
}
```

Section 183.2: Check connection using ConnectivityManager

```
    public static boolean isConnectedNetwork (Context context) {

        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
().isConnectedOrConnecting ();

    }
```

Section 183.3: Use network intents to perform tasks while data is allowed

When your device connects to a network, an intent is sent. Many apps don't check for these intents, but to make your application work properly, you can listen to network change intents that will tell you when communication is possible. To check for network connectivity you can, for example, use the following clause:

> ```
> if (intent.getAction().equals(android.net.ConnectivityManager.CONNECTIVITY_ACTION)){
>  NetworkInfo info = intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWORK_INFO);
> //perform your action when connected to a network
> }
> ```

Chapter 184: Java on Android

Android supports all Java 7 language features and a subset of Java 8 language features that vary by platform version. This page describes the new language features you can use, how to properly configure your project to use them and any known issues you may encounter.

Section 184.1: Java 8 features subset with Retrolambda

Retrolambda lets you run Java 8 code with lambda expressions, method references and try-with-resources statements on Java 7, 6 or 5. It does this by transforming your Java 8 compiled bytecode so that it can run on an older Java runtime.

Backported Language Features:

- Lambda expressions are backported by converting them to anonymous inner classes. This includes the optimisation of using a singleton instance for stateless lambda expressions to avoid repeated object allocation. Method references are basically just syntax sugar for lambda expressions and they are backported in the same way.
- Try-with-resources statements are backported by removing calls to `Throwable.addSuppressed` if the target bytecode version is below Java 7. If you would like the suppressed exceptions to be logged instead of swallowed, please create a feature request and we'll make it configurable.
- `Objects.requireNonNull` calls are replaced with calls to `Object.getClass` if the target bytecode version is below Java 7. The synthetic null checks generated by JDK 9 use `Objects.requireNonNull`, whereas earlier JDK versions used `Object.getClass`.
- Optionally also:

    1. Default methods are backported by copying the default methods to a companion class (interface name + "$") as static methods, replacing the default methods in the interface with abstract methods, and by adding the necessary method implementations to all classes which implement that interface.
    2. Static methods on interfaces are backported by moving the static methods to a companion class (interface name + "$"), and by changing all methods calls to call the new method location.

Known Limitations:

- Does not backport Java 8 APIs
- Backporting default methods and static methods on interfaces requires all backported interfaces and all classes which implement them or call their static methods to be backported together, with one execution of Retrolambda. In other words, you must always do a clean build. Also, backporting default methods won't work across module or dependency boundaries.
- May break if a future JDK 8 build stops generating a new class for each `invokedynamic` call. Retrolambda works so that it captures the bytecode that `java.lang.invoke.LambdaMetafactory` generates dynamically, so optimisations to that mechanism may break Retrolambda.

Retrolambda gradle plugin will automatically build your android project with Retrolambda. The latest version can be found on the releases page.

Usage:

1. Download and install jdk8
2. Add the following to your `build.gradle`

```
buildscript {
  repositories {
     mavenCentral()
  }

  dependencies {
     classpath 'me.tatarka:gradle-retrolambda:<latest version>'
  }
}

// Required because retrolambda is on maven central
repositories {
  mavenCentral()
}

apply plugin: 'com.android.application' //or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'

android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Known Issues:

- Lint fails on java files that have lambdas. Android's lint doesn't understand java 8 syntax and will fail silently or loudly. There is now an experimental fork that fixes the issue.
- Using Google Play Services causes Retrolambda to fail. Version 5.0.77 contains bytecode that is incompatible with Retrolambda. This should be fixed in newer versions of play services, if you can update, that should be the preferred solution. To work around this issue, you can either use an earlier version like 4.4.52 or add `-noverify` to the jvm args.

```
retrolambda {
  jvmArgs '-noverify'
}
```

Chapter 185: Android Java Native Interface (JNI)

JNI (Java Native Interface) is a powerful tool that enables Android developers to utilize the NDK and use C++ native code in their applications. This topic describes the usage of Java <-> C++ interface.

Section 185.1: How to call functions in a native library via the JNI interface

The Java Native Interface (JNI) allows you to call native functions from Java code, and vice versa. This example shows how to load and call a native function via JNI, it does not go into accessing Java methods and fields from native code using JNI functions.

Suppose you have a native library named `libjniexample.so` in the `project/libs/<architecture>` folder, and you want to call a function from the `JNITest`Java class inside the `com.example.jniexample` package.

In the JNITest class, declare the function like this:

```
public native int testJNIfunction(int a, int b);
```

In your native code, define the function like this:

```
#include <jni.h>

JNIEXPORT jint JNICALL Java_com_example_jniexample_JNITest_testJNIfunction(JNIEnv *pEnv, jobject
thiz, jint a, jint b)
{
    return a + b;
}
```

The `pEnv` argument is a pointer to the JNI environment that you can pass to JNI functions to access methods and fields of Java objects and classes. The `thiz` pointer is a `jobject` reference to the Java object that the native method was called on (or the class if it is a static method).

In your Java code, in `JNITest`, load the library like this:

```
static{
    System.loadLibrary("jniexample");
}
```

Note the `lib` at the start, and the `.so` at the end of the filename are omitted.

Call the native function from Java like this:

```
JNITest test = new JNITest();
int c = test.testJNIfunction(3, 4);
```

Section 185.2: How to call a Java method from native code

The Java Native Interface (JNI) allows you to call Java functions from native code. Here is a simple example of how to do it:

Java code:

```
package com.example.jniexample;
public class JNITest {
    public static int getAnswer(bool) {
        return 42;
    }
}
```

Native code:

```
int getTheAnswer()
{
    // Get JNI environment
    JNIEnv *env = JniGetEnv();

    // Find the Java class - provide package ('.' replaced to '/') and class name
    jclass jniTestClass = env->FindClass("com/example/jniexample/JNITest");

    // Find the Java method - provide parameters inside () and return value (see table below for an
explanation of how to encode them)
    jmethodID getAnswerMethod = env->GetStaticMethodID(jniTestClass, "getAnswer", "(Z)I;");

    // Calling the method
    return (int)env->CallStaticObjectMethod(jniTestClass, getAnswerMethod, (jboolean)true);
}
```

JNI method signature to Java type:

JNI Signature	Java Type
Z	boolean
B	byte
C	char
S	short
I	int
J	long
F	float
D	double
L fully-qualified-class ;	fully-qualified-class
[type	type[]

So for our example we used (Z)I - which means the function gets a boolean and returns an int.

Section 185.3: Utility method in JNI layer

This method will help to get the Java string from C++ string.

```
jstring getJavaStringFromCPPString(JNIEnv *global_env, const char* cstring) {

        jstring nullString = global_env->NewStringUTF(NULL);

        if (!cstring) {
            return nullString;
        }

        jclass strClass = global_env->FindClass("java/lang/String");
        jmethodID ctorID = global_env->GetMethodID(strClass, "<init>",
                "([BLjava/lang/String;)V");
```

```
        jstring encoding = global_env->NewStringUTF("UTF-8");

        jbyteArray bytes = global_env->NewByteArray(strlen(cstring));
        global_env->SetByteArrayRegion(bytes, 0, strlen(cstring), (jbyte*) cstring);
        jstring str = (jstring) global_env->NewObject(strClass, ctorID, bytes,
                encoding);

        global_env->DeleteLocalRef(strClass);
        global_env->DeleteLocalRef(encoding);
        global_env->DeleteLocalRef(bytes);

        return str;
    }
```

This method will help you to convert jbyteArray to char

```
char* as_unsigned_char_array(JNIEnv *env, jbyteArray array) {
    jsize length = env->GetArrayLength(array);
    jbyte* buffer = new jbyte[length + 1];

    env->GetByteArrayRegion(array, 0, length, buffer);
    buffer[length] = '\0';

    return (char*) buffer;
}
```

Chapter 186: Notification Channel Android O

Method	Description
IMPORTANCE_MAX	unused
IMPORTANCE_HIGH	shows everywhere, makes noise and peeks
IMPORTANCE_DEFAULT	shows everywhere, makes noise, but does not visually intrude
IMPORTANCE_LOW	shows everywhere, but is not intrusive
IMPORTANCE_MIN	only shows in the shade, below the fold
IMPORTANCE_NONE	a notification with no importance; does not show in the shade

Notification channels enable us app developers to group our notifications into groups—channels—with the user having the ability to modify notification settings for the entire channel at once.In Android O this feature is introduced.Right now it is available developers preview.

Section 186.1: Notification Channel

What Are Notification Channels?

Notification channels enable us app developers to group our notifications into groups—channels—with the user having the ability to modify notification settings for the entire channel at once. For example, for each channel, users can completely block all notifications, override the importance level, or allow a notification badge to be shown. This new feature helps in greatly improving the user experience of an app.

Create Notification Channels

```
 import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.content.Context;
import android.content.ContextWrapper;
import android.graphics.Color;

public class NotificationUtils extends ContextWrapper {


private NotificationManager mManager;
public static final String ANDROID_CHANNEL_ID = "com.sai.ANDROID";
public static final String IOS_CHANNEL_ID = "com.sai.IOS";
public static final String ANDROID_CHANNEL_NAME = "ANDROID CHANNEL";
public static final String IOS_CHANNEL_NAME = "IOS CHANNEL";

public NotificationUtils(Context base) {
    super(base);
    createChannels();
}

public void createChannels() {

    // create android channel
    NotificationChannel androidChannel = new NotificationChannel(ANDROID_CHANNEL_ID,
            ANDROID_CHANNEL_NAME, NotificationManager.IMPORTANCE_DEFAULT);
    // Sets whether notifications posted to this channel should display notification lights
    androidChannel.enableLights(true);
    // Sets whether notification posted to this channel should vibrate.
```

```
    androidChannel.enableVibration(true);
    // Sets the notification light color for notifications posted to this channel
    androidChannel.setLightColor(Color.BLUE);
    // Sets whether notifications posted to this channel appear on the lockscreen or not
    androidChannel.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);

    getManager().createNotificationChannel(androidChannel);

    // create ios channel
    NotificationChannel iosChannel = new NotificationChannel(IOS_CHANNEL_ID,
            IOS_CHANNEL_NAME, NotificationManager.IMPORTANCE_HIGH);
    iosChannel.enableLights(true);
    iosChannel.enableVibration(true);
    iosChannel.setLightColor(Color.GRAY);
    iosChannel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);
    getManager().createNotificationChannel(iosChannel);


  }


private NotificationManager getManager() {
    if (mManager == null) {
        mManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    }
    return mManager;
}}
```

In the code above, we created two instances of the NotificationChannel, passing uniqueid a channel name, and also an importance level in its constructor. For each notification channel, we applied following characteristics.

1. Sound
2. Lights
3. Vibration
4. Notification to show on lock screen.

Finally, we got the NotificationManager from the system and then registered the channel by calling the method createNotificationChannel(), passing the channel we have created.

We can create multiple notification channels all at once with createNotificationChannels(), passing a Java list of NotificationChannel instances. You can get all notification channels for an app with getNotificationChannels() and get a specific channel with getNotificationChannel(), passing only the channel id as an argument.

Importance Level in Notification Channels

Method	Description
IMPORTANCE_MAX	unused
IMPORTANCE_HIGH	shows everywhere, makes noise and peeks
IMPORTANCE_DEFAULT	shows everywhere, makes noise, but does not visually intrude
IMPORTANCE_LOW	shows everywhere, but is not intrusive,value is 0
IMPORTANCE_MIN	only shows in the shade, below the fold
IMPORTANCE_NONE	a notification with no importance; does not show in the shade

Create Notification and Post to channel

We have created two notification one using NotificationUtils another using NotificationBuilder.

```
public Notification.Builder getAndroidChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), ANDROID_CHANNEL_ID)
            .setContentTitle(title)
            .setContentText(body)
            .setSmallIcon(android.R.drawable.stat_notify_more)
            .setAutoCancel(true);
}

public Notification.Builder getIosChannelNotification(String title, String body) {
    return new Notification.Builder(getApplicationContext(), IOS_CHANNEL_ID)
            .setContentTitle(title)
            .setContentText(body)
            .setSmallIcon(android.R.drawable.stat_notify_more)
            .setAutoCancel(true);
}
```

We can also set NotificationChannel Using Notification.Builder().For that we can use ***setChannel(String channelId)***.

Update Notification Channel Settings

Once you create a notification channel, the user is in charge of its settings and behavior. You can call createNotificationChannel() again to rename an existing notification channel, or update its description. The following sample code describes how you can redirect a user to the settings for a notification channel by creating an intent to start an activity. In this case, the intent requires extended data including the ID of the notification channel and the package name of your app.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    //...
    Button buttonAndroidNotifSettings = (Button) findViewById(R.id.btn_android_notif_settings);
    buttonAndroidNotifSettings.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View view) {
            Intent i = new Intent(Settings.ACTION_CHANNEL_NOTIFICATION_SETTINGS);
            i.putExtra(Settings.EXTRA_APP_PACKAGE, getPackageName());
            i.putExtra(Settings.EXTRA_CHANNEL_ID, NotificationUtils.ANDROID_CHANNEL_ID);
            startActivity(i);
        }
    });
}
```

XML file:

```
<!--...-->
<Button
   android:id="@+id/btn_android_notif_settings"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="Notification Settings"/>
<!--...-->
```

Deleting Notification Channel

You can delete notification channels by calling deleteNotificationChannel().

```
NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
```

```
// The id of the channel.
String id = "my_channel_01";
NotificationChannel mChannel = mNotificationManager.getNotificationChannel(id);
mNotificationManager.deleteNotificationChannel(mChannel);
```

Now Create MainActivity and xml

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:app="http://schemas.android.com/apk/res-auto"
       xmlns:tools="http://schemas.android.com/tools"
       android:id="@+id/activity_main"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:orientation="vertical"
       android:layout_margin="16dp"
       tools:context="com.chikeandroid.tutsplusalerts.MainActivity">

    <LinearLayout
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:orientation="vertical">

        <TextView
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="Tuts+ Android Channel"
               android:layout_gravity="center_horizontal"
               android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
               android:id="@+id/et_android_title"
               android:layout_width="match_parent"
               android:layout_height="wrap_content"
               android:hint="Title"/>

        <EditText
               android:id="@+id/et_android_author"
               android:layout_width="match_parent"
               android:layout_height="wrap_content"
               android:hint="Author"/>

        <Button
               android:id="@+id/btn_send_android"
               android:layout_width="match_parent"
               android:layout_height="wrap_content"
               android:text="Send"/>
    </LinearLayout>

    <LinearLayout
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:orientation="vertical"
           android:layout_marginTop="20dp">

        <TextView
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
```

```
            android:text="Tuts+ IOS Channel"
            android:layout_gravity="center_horizontal"
            android:textAppearance="@style/TextAppearance.AppCompat.Title"/>

        <EditText
            android:id="@+id/et_ios_title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Title"
            />

        <EditText
            android:id="@+id/et_ios_author"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Author"/>
        <Button
            android:id="@+id/btn_send_ios"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Send"/>
    </LinearLayout>

</LinearLayout>
```

MainActivity.java

we are going to edit our MainActivity so that we can get the title and author from the EditText components and then send these to the Android channel. We get the Notification.Builder for the Android channel we created in our NotificationUtils, and then notify the NotificationManager.

import android.app.Notification; import android.os.Bundle; import android.support.v7.app.AppCompatActivity; import android.text.TextUtils; import android.view.View; import android.widget.Button; import android.widget.EditText;

```
public class MainActivity extends AppCompatActivity {

    private NotificationUtils mNotificationUtils;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mNotificationUtils = new NotificationUtils(this);

        final EditText editTextTitleAndroid = (EditText) findViewById(R.id.et_android_title);
        final EditText editTextAuthorAndroid = (EditText) findViewById(R.id.et_android_author);
        Button buttonAndroid = (Button) findViewById(R.id.btn_send_android);

        buttonAndroid.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String title = editTextTitleAndroid.getText().toString();
                String author = editTextAuthorAndroid.getText().toString();

                if(!TextUtils.isEmpty(title) && !TextUtils.isEmpty(author)) {
                    Notification.Builder nb = mNotificationUtils.
                            getAndroidChannelNotification(title, "By " + author);
```

```
                    mNotificationUtils.getManager().notify(107, nb.build());
                }
            }
        });
    }
}
```

Chapter 187: Robolectric

Unit testing is taking a piece of code and testing it independently without any other dependencies or parts of the system running (for example the database).

Robolectric is a unit test framework that de-fangs the Android SDK jar so you can test-drive the development of your Android app. Tests run inside the JVM on your workstation in seconds.

Combing them both allows you to run fast tests on the JVN still using the Android API's.

Section 187.1: Robolectric test

```
@RunWith(RobolectricTestRunner.class)
public class MyActivityTest {

  @Test
  public void clickingButton_shouldChangeResultsViewText() throws Exception {
    MyActivity activity = Robolectric.setupActivity(MyActivity.class);

    Button button = (Button) activity.findViewById(R.id.button);
    TextView results = (TextView) activity.findViewById(R.id.results);

    button.performClick();
    assertThat(results.getText().toString()).isEqualTo("Robolectric Rocks!");
  }
}
```

Section 187.2: Configuration

To configure robolectric add `@Config` annotation to test class or method.

Run with custom Application class

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
}
```

Set target SDK

```
@RunWith(RobolectricTestRunner.class)
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
}
```

Run with custom manifest

When specified, robolectric will look relative to the current directory. Default value is `AndroidManifest.xml`

Resources and assets will be loaded relative to the manifest.

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
}
```

Use qualifiers

Possible qualifiers can be found in android docs.

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

Chapter 188: Moshi

Moshi is a modern JSON library for Android and Java. It makes it easy to parse JSON into Java objects and Java back into JSON.

Section 188.1: JSON into Java

```
String json = ...;

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

BlackjackHand blackjackHand = jsonAdapter.fromJson(json);
System.out.println(blackjackHand);
```

Section 188.2: serialize Java objects as JSON

```
BlackjackHand blackjackHand = new BlackjackHand(
    new Card('6', SPADES),
    Arrays.asList(new Card('4', CLUBS), new Card('A', HEARTS)));

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

String json = jsonAdapter.toJson(blackjackHand);
System.out.println(json);
```

Section 188.3: Built in Type Adapters

Moshi has built-in support for reading and writing Java's core data types:

- Primitives (int, float, char...) and their boxed counterparts (Integer, Float, Character...).
- Arrays
- Collections
- Lists
- Sets
- Maps Strings Enums

It supports your model classes by writing them out field-by-field. In the example above Moshi uses these classes:

```
class BlackjackHand {
  public final Card hidden_card;
  public final List<Card> visible_cards;
  ...
}

class Card {
  public final char rank;
  public final Suit suit;
  ...
}

enum Suit {
  CLUBS, DIAMONDS, HEARTS, SPADES;
}
to read and write this JSON:
```

```
{
  "hidden_card": {
    "rank": "6",
    "suit": "SPADES"
  },
  "visible_cards": [
    {
      "rank": "4",
      "suit": "CLUBS"
    },
    {
      "rank": "A",
      "suit": "HEARTS"
    }
  ]
}
```

Chapter 189: Strict Mode Policy : A tool to catch the bug in the Compile Time.

Strict Mode is a special class introduced in Android 2.3 for debugging. This developer tools detect things done accidentally and bring them to our attention so that we can fix them. It is most commonly used to catch the accidental disk or network access on the applications' main thread, where UI operations are received and animations takes place. StrictMode is basically a tool to catch the bug in the Compile Time mode.

Section 189.1: The below Code Snippet is to setup the StrictMode for Thread Policies. This Code is to be set at the entry points to our application

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //Logs a message to LogCat
    .build())
```

Section 189.2: The below code deals with leaks of memory, like it detects when in SQLLite finalize is called or not

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

Chapter 190: Internationalization and localization (I18N and L10N)

Internationalization (i18n) and Localization (L10n) are used to adapt software according to differences in languages, regional differences and target audience.

Internationalization : the process of planning for future localization i.e. making the software design flexible to an extent that it can adjust and adapt to future localization efforts.

Localization : the process of adapting the software to a particular region/country/market (locale).

Section 190.1: Planning for localization : enable RTL support in Manifest

RTL (Right-to-left) support is an essential part in planning for i18n and L10n. Unlike English language which is written from left to right, many languages like Arabic, Japanese, Hebrew, etc. are written from right to left. To appeal to a more global audience, it is a good idea to plan your layouts to support these language from the very beginning of the project, so that adding localization is easier later on.

RTL support can be enabled in an Android app by adding the `supportsRtl` tag in the `AndroidManifest`, like so :

```
<application
    ...
    android:supportsRtl="true"
    ...>
...
</application>
```

Section 190.2: Planning for localization : Add RTL support in Layouts

Starting SDK 17 (Android 4.2), RTL support was added in Android layouts and is an essential part of localization. Going forward, the `left/right` notation in layouts should be replaced by `start/end` notation. If, however, your project has a `minSdk` value less than `17`, then both `left/right` and `start/end` notation should be used in layouts.

For relative layouts, `alignParentStart` and `alignParentEnd` should be used, like so:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
     <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"/>
     <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"/>
</RelativeLayout>
```

For specifying gravity and layout gravity, similar notation should be used, like so :

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left|start"
        android:gravity="left|start"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|end"
        android:gravity="right|end"/>
```

Paddings and margins should also be specified accordingly, like so :

```
<include layout="@layout/notification"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="12dp"
    android:layout_marginStart="12dp"
    android:paddingLeft="128dp"
    android:paddingStart="128dp"
    android:layout_toLeftOf="@id/cancel_action"
    android:layout_toStartOf="@id/cancel_action"/>
<include layout="@layout/notification2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginRight="12dp"
    android:layout_marginEnd="12dp"
    android:paddingRight="128dp"
    android:paddingEnd="128dp"
    android:layout_toRightOf="@id/cancel_action"
    android:layout_toEndOf="@id/cancel_action"/>
```

Section 190.3: Planning for localization : Test layouts for RTL

To test if the layouts that have been created are RTL compatible, do the following :

> Go to Settings -> Developer options -> Drawing -> Force RTL layout direction

Enabling this option would force the device to use RTL locales and you can easily verify all parts of the app for RTL support. Note that you don't need to actually add any new locales/ language support up till this point.

Section 190.4: Coding for Localization : Creating default strings and resources

The first step for coding for localization is to create default resources. This step is so implicit that many developers do not even think about it. However, creating default resources is important because if the device runs on an unsupported locale, it would load all of its resources from the default folders. If even one of the resources is missing from the default folders, the app would simply crash.

The default set of strings should be put in the following folder at the specified location:

```
res/values/strings.xml
```

This file should contain the strings in the language that majority users of the app are expected to speak.

Also, default resources for the app should be placed at the following folders and locations :

```
res/drawable/
res/layout/
```

If your app requires folders like `anim`, or `xml`, the default resources should be added to the following folders and locations:

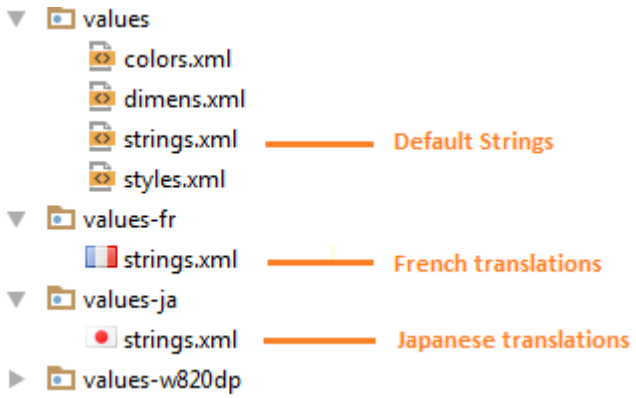
```
res/anim/
res/xml/
res/raw/
```

Section 190.5: Coding for localization : Providing alternative strings

To provide translations in other languages (locales), we need to create a `strings.xml` in a separate folder by the following convention :

```
res/values-<locale>/strings.xml
```

An example for the same is given below:



In this example, we have default English strings in the file `res/values/strings.xml`, French translations are provided in the folder `res/values-fr/strings.xml` and Japanese translations are provided in the folder `res/values-ja/strings.xml`

Other translations for other locales can similarly be added to the app.

> A complete list of locale codes can be found here : ISO 639 codes

Non-translatable Strings:

Your project may have certain strings that are not to be translated. Strings which are used as keys for SharedPreferences or strings which are used as symbols, fall in this category. These strings should be stored only in the default `strings.xml` and should be marked with a `translatable="false"` attribute. e.g.

```
<string name="pref_widget_display_label_hot">Hot News</string>
<string name="pref_widget_display_key" translatable="false">widget_display</string>
<string name="pref_widget_display_hot" translatable="false">0</string>
```

This attribute is important because translations are often carried out by professionals who are bilingual. This would allow these persons involved in translations to identify strings which are not to be translated, thus saving time and money.

Section 190.6: Coding for localization : Providing alternate layouts

Creating language specific layouts is often unnecessary if you have specified the correct `start/end` notation, as described in the earlier example. However, there may be situations where the defaults layouts may not work correctly for certain languages. Sometimes, left-to-right layouts may not translate for RTL languages. It is necessary to provide the correct layouts in such cases.

To provide complete optimization for RTL layouts, we can use entirely separate layout files using the `ldrtl` resource qualifier (`ldrtl` stands for layout-direction-right-to-left}). For example, we can save your default layout files in `res/layout/` and our RTL optimized layouts in `res/layout-ldrtl/`.

The `ldrtl` qualifier is great for drawable resources, so that you can provide graphics that are oriented in the direction corresponding to the reading direction.

Here is a great post which describes the precedence of the `ldrtl` layouts : Language specific layouts

Chapter 191: Fast way to setup Retrolambda on an android project.

Retrolambda is a library which allows to use Java 8 lambda expressions, method references and try-with-resources statements on Java 7, 6 or 5.

The Gradle Retrolambda Plug-in allows to integrate Retrolambda into a Gradle based build. This allows for example to use these constructs in an Android application, as standard Android development currently does not yet support Java 8.

Section 191.1: Setup and example how to use:

Setup Steps:

1. Download and install jdk8.
2. Add the following to your project's main build.gradle

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'
    }
}
```

3. Now add this to your application module's build.gradle

```
apply plugin: 'com.android.application' // or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'
```

4. Add these lines to your application module's build.gradle to inform the IDE of the language level:

```
android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Example:

So things like this:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        log("Clicked");
    }
});
```

Become this:

```
button.setOnClickListener(v -> log("Clicked"));
```

Chapter 192: How to use SparseArray

A `SparseArray` is an alternative for a `Map`. A `Map` requires its keys to be objects. The phenomenon of autoboxing occurs when we want to use a primitive **`int`** value as key. The compiler automatically converts primitive values to their boxed types (e.g. **`int`** to `Integer`). The difference in memory footprint is noticeable: **`int`** uses 4 bytes, `Integer` uses 16 bytes. A `SparseArray` uses **`int`** as key value.

Section 192.1: Basic example using SparseArray

```
class Person {
    String name;

    public Person(String name) {
        this.name = name;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Person person = (Person) o;

        return name != null ? name.equals(person.name) : person.name == null;
    }

    @Override
    public int hashCode() {
        return name != null ? name.hashCode() : 0;
    }

    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' +
                '}';
    }
}

final Person steve = new Person("Steve");
Person[] persons = new Person[] { new Person("John"), new Person("Gwen"), steve, new Person("Rob")
};
int[] identifiers = new int[] {1234, 2345, 3456, 4567};

final SparseArray<Person> demo = new SparseArray<>();

// Mapping persons to identifiers.
for (int i = 0; i < persons.length; i++) {
    demo.put(identifiers[i], persons[i]);
}

// Find the person with identifier 1234.
Person id1234 = demo.get(1234); // Returns John.

// Find the person with identifier 6410.
Person id6410 = demo.get(6410); // Returns null.

// Find the 3rd person.
Person third = demo.valueAt(3); // Returns Rob.
```

```
// Find the 42th person.
//Person fortysecond = demo.valueAt(42); // Throws ArrayIndexOutOfBoundsException.

// Remove the last person.
demo.removeAt(demo.size() - 1); // Rob removed.

// Remove the person with identifier 1234.
demo.delete(1234); // John removed.

// Find the index of Steve.
int indexOfSteve = demo.indexOfValue(steve);

// Find the identifier of Steve.
int identifierOfSteve = demo.keyAt(indexOfSteve);
```

Tutorial on YouTube

Chapter 193: Shared Element Transitions

Here you find examples for transition between `Activities` or `Fragments` using a shared element. An example for this behaviour is the Google Play Store App which translates an App's icon from the list to the App's details view.

Section 193.1: Shared Element Transition between two Fragments

In this example, one of two different `ImageViews` should be translated from the `ChooserFragment` to the `DetailFragment`.

In the `ChooserFragment` layout we need the unique `transitionName` attributes:

```
<ImageView
   android:id="@+id/image_first"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/ic_first"
   android:transitionName="fistImage" />

<ImageView
   android:id="@+id/image_second"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/ic_second"
   android:transitionName="secondImage" />
```

In the `ChooserFragments` class, we need to pass the `View` which was clicked and an ID to the parent `Activity` wich is handling the replacement of the fragments (we need the ID to know which image resource to show in the `DetailFragment`). How to pass information to a parent activity in detail is surely covered in another documentation.

```
view.findViewById(R.id.image_first).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (mCallback != null) {
            mCallback.showDetailFragment(view, 1);
        }
    }
});

view.findViewById(R.id.image_second).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (mCallback != null) {
            mCallback.showDetailFragment(view, 2);
        }
     }
});
```

In the `DetailFragment`, the `ImageView` of the shared element also needs the unique `transitionName` attribute.

```
<ImageView
   android:id="@+id/image_shared"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center"
```

```
    android:transitionName="sharedImage" />
```

In the `onCreateView()` method of the `DetailFragment`, we have to decide which image resource should be shown (if we don't do that, the shared element will disappear after the transition).

```
public static DetailFragment newInstance(Bundle args) {
    DetailFragment fragment = new DetailFragment();
    fragment.setArguments(args);
    return fragment;
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View view = inflater.inflate(R.layout.fragment_detail, container, false);

    ImageView sharedImage = (ImageView) view.findViewById(R.id.image_shared);

    // Check which resource should be shown.
    int type = getArguments().getInt("type");

    // Show image based on the type.
    switch (type) {
        case 1:
            sharedImage.setBackgroundResource(R.drawable.ic_first);
            break;

        case 2:
            sharedImage.setBackgroundResource(R.drawable.ic_second);
            break;
    }

    return view;
}
```

The parent `Activity` is receiving the callbacks and handles the replacement of the fragments.

```
@Override
public void showDetailFragment(View sharedElement, int type) {
    // Get the chooser fragment, which is shown in the moment.
    Fragment chooserFragment = getFragmentManager().findFragmentById(R.id.fragment_container);

    // Set up the DetailFragment and put the type as argument.
    Bundle args = new Bundle();
    args.putInt("type", type);
    Fragment fragment = DetailFragment.newInstance(args);

    // Set up the transaction.
    FragmentTransaction transaction = getFragmentManager().beginTransaction();

    // Define the shared element transition.
    fragment.setSharedElementEnterTransition(new DetailsTransition());
    fragment.setSharedElementReturnTransition(new DetailsTransition());

    // The rest of the views are just fading in/out.
    fragment.setEnterTransition(new Fade());
    chooserFragment.setExitTransition(new Fade());

    // Now use the image's view and the target transitionName to define the shared element.
    transaction.addSharedElement(sharedElement, "sharedImage");
```

```
    // Replace the fragment.
    transaction.replace(R.id.fragment_container, fragment, fragment.getClass().getSimpleName());

    // Enable back navigation with shared element transitions.
    transaction.addToBackStack(fragment.getClass().getSimpleName());

    // Finally press play.
    transaction.commit();
}
```

Not to forget - the `Transition` itself. This example moves and scales the shared element.

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class DetailsTransition extends TransitionSet {

    public DetailsTransition() {
        setOrdering(ORDERING_TOGETHER);
        addTransition(new ChangeBounds()).
            addTransition(new ChangeTransform()).
            addTransition(new ChangeImageTransform());
    }

}
```

Chapter 194: Android Things

Section 194.1: Controlling a Servo Motor

This example assumes you have a servo with the following characteristics, which happen to be typical:

- movement between 0 and 180 degrees
- pulse period of 20 ms
- minimum pulse length of 0.5 ms
- maximum pulse length of 2.5 ms

You need to check if those values match your hardware, since forcing it to go outside its specified operating range can damage the servo. A damaged servo in turn has the potential to damage your Android Things device. The example `ServoController` class consists of two methods, `setup()` and `setPosition()`:

```
public class ServoController {
    private double periodMs, maxTimeMs, minTimeMs;
    private Pwm pin;

    public void setup(String pinName) throws IOException {
        periodMs  = 20;
        maxTimeMs = 2.5;
        minTimeMs = 0.5;

        PeripheralManagerService service = new PeripheralManagerService();
        pin = service.openPwm(pinName);

        pin.setPwmFrequencyHz(1000.0d / periodMs);
        setPosition(90);
        pin.setEnabled(true);
    }

    public void setPosition(double degrees) {
        double pulseLengthMs = (degrees / 180.0 * (maxTimeMs - minTimeMs)) + minTimeMs;

        if (pulseLengthMs < minTimeMs) {
            pulseLengthMs = minTimeMs;
        } else if (pulseLengthMs > maxTimeMs) {
            pulseLengthMs = maxTimeMs;
        }

        double dutyCycle = pulseLengthMs / periodMs * 100.0;

        Log.i(TAG, "Duty cycle = " + dutyCycle + " pulse length = " + pulseLengthMs);

        try {
            pin.setPwmDutyCycle(dutyCycle);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

You can discover pin names that support PWM on your device as follows:

```
PeripheralManagerService service = new PeripheralManagerService();

for (String pinName : service.getPwmList() ) {
```

```
    Log.i("ServoControlled","Pwm pin found: " + pinName);
}
```

In order to make your servo swinging forever between 80 degrees and 100 degrees, you can simply use the following code:

```
final ServoController servoController = new ServoController(pinName);

Thread th = new Thread(new Runnable() {
    @Override
    public void run() {
        while (true) {
            try {
                servoController.setPosition(80);
                Thread.sleep(500);
                servoController.setPosition(100);
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
});
th.start();
```

You can compile and deploy all of the above code without actually hooking any servo motors to the computing device. For the wiring, refer to your computing device pinout chart (e.g. a Raspberry Pi 3 pinout chart is available here).

Then you need to hook your servo to Vcc, Gnd, and signal.

Chapter 195: Library Dagger 2: Dependency Injection in Applications

Dagger 2, as explained on GitHub, is a compile-time evolution approach to dependency injection. Taking the approach started in Dagger 1.x to its ultimate conclusion, Dagger 2.x eliminates all reflection, and improves code clarity by removing the traditional `ObjectGraph/Injector` in favor of user-specified `@Component` interfaces.

Section 195.1: Create @Module Class and @Singleton annotation for Object

```
import javax.inject.Singleton;
import dagger.Module;
import dagger.Provides;

@Module
public class VehicleModule {

    @Provides @Singleton
    Motor provideMotor(){
        return new Motor();
    }

    @Provides @Singleton
    Vehicle provideVehicle(){
        return new Vehicle(new Motor());
    }
}
```

Every provider (or method) must have the `@Provides` annotation and the class must have the `@Module` annotation. The `@Singleton` annotation indicates that there will be only one instance of the object.

Section 195.2: Request Dependencies in Dependent Objects

Now that you have the providers for your different models, you need to request them. Just as `Vehicle` needs `Motor`, you have to add the `@Inject` annotation in the `Vehicle` constructor as follows:

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

You can use the `@Inject` annotation to request dependencies in the constructor, fields, or methods. In this example, I'm keeping the injection in the constructor.

Section 195.3: Connecting @Modules with @Inject

The connection between the provider of dependencies, `@Module`, and the classes requesting them through `@Inject` is made using `@Component`, which is an interface:

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
```

```
public interface VehicleComponent {
    Vehicle provideVehicle();
}
```

For the `@Component` annotation, you have to specify which modules are going to be used. In this example `VehicleModule` is used, which is defined in this example. If you need to use more modules, then just add them using a comma as a separator.

Section 195.4: Using @Component Interface to Obtain Objects

Now that you have every connection ready, you have to obtain an instance of this interface and invoke its methods to obtain the object you need:

```
VehicleComponent component = Dagger_VehicleComponent.builder().vehicleModule(new
VehicleModule()).build();
vehicle = component.provideVehicle();
Toast.makeText(this, String.valueOf(vehicle.getSpeed()), Toast.LENGTH_SHORT).show();
```

When you try to create a new object of the interface with the `@Component` annotation, you have to do it using the prefix `Dagger_<NameOfTheComponentInterface>`, in this case `Dagger_VehicleComponent`, and then use the builder method to call every module within.

Chapter 196: JCodec

Section 196.1: Getting Started

You can get JCodec automatically with maven. For this just add below snippet to your pom.xml .

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

Section 196.2: Getting frame from movie

Getting a single frame from a movie (supports only AVC, H.264 in MP4, ISO BMF, Quicktime container):

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

Getting a sequence of frames from a movie (supports only AVC, H.264 in MP4, ISO BMF, Quicktime container):

```
double startSec = 51.632;
FileChannelWrapper ch = null;
try {
    ch = NIOUtils.readableFileChannel(new File("filename.mp4"));
    FrameGrab fg = new FrameGrab(ch);
    grab.seek(startSec);
    for (int i = 0; i < 100; i++) {
        ImageIO.write(grab.getFrame(), "png",
            new File(System.getProperty("user.home"), String.format("Desktop/frame_%08d.png", i)));
    }
} finally {
    NIOUtils.closeQuietly(ch);
}
```

Chapter 197: Formatting phone numbers with pattern.

This example show you how to format phone numbers with a patter

You will need the following library in your gradle.

compile 'com.googlecode.libphonenumber:libphonenumber:7.2.2'

Section 197.1: Patterns + 1 (786) 1234 5678

Given a normalized phone number like +178612345678 we will get a formatted number with the provided pattern.

```
private String getFormattedNumber(String phoneNumber) {

    PhoneNumberUtil phoneNumberUtil = PhoneNumberUtil.getInstance();

    Phonemetadata.NumberFormat numberFormat = new Phonemetadata.NumberFormat();

    numberFormat.pattern = "(\\d{3})(\\d{3})(\\d{4})";

    numberFormat.format = "($1) $2-$3";

    List<Phonemetadata.NumberFormat> newNumberFormats = new ArrayList<>();

    newNumberFormats.add(numberFormat);

    Phonenumber.PhoneNumber phoneNumberPN = null;

    try {
        phoneNumberPN = phoneNumberUtil.parse(phoneNumber, Locale.US.getCountry());
        phoneNumber = phoneNumberUtil.formatByPattern(phoneNumberPN,
PhoneNumberUtil.PhoneNumberFormat.INTERNATIONAL, newNumberFormats);

    } catch (NumberParseException e) {
        e.printStackTrace();
    }

    return phoneNumber;
}
```

Chapter 198: Paint

A paint is one of the four objects needed to draw, along with a Canvas (holds drawing calls), a Bitmap (holds the pixels), and a drawing primitive (Rect, Path, Bitmap...)

Section 198.1: Creating a Paint

You can create a new paint with one of these 3 constructors:

- `new Paint()` Create with default settings
- `new Paint(int flags)` Create with flags
- `new Paint(Paint from)` Copy settings from another paint

It is generally suggested to never create a paint object, or any other object in onDraw() as it can lead to performance issues. (Android Studio will probably warn you) Instead, make it global and initialize it in your class constructor like so:

```
public class CustomView extends View {

    private Paint paint;

    public CustomView(Context context) {
        super(context);
        paint = new Paint();
        //...
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        paint.setColor(0xFF000000);
        // ...
    }
}
```

Section 198.2: Setting up Paint for text

Text drawing settings

- `setTypeface(Typeface typeface)` Set the font face. See Typeface
- `setTextSize(int size)` Set the font size, in pixels.
- `setColor(int color)` Set the paint drawing color, including the text color. You can also use `setARGB(int a, int r, int g, int b` and `setAlpha(int alpha)`
- `setLetterSpacing(float size)` Set the spacing between characters, in ems. Default value is 0, a negative value will tighten the text, while a positive one will expand it.
- `setTextAlign(Paint.Align align)` Set text alignment relative to its origin. `Paint.Align.LEFT` will draw it to the right of the origin, `RIGHT` will draw it to the left, and `CENTER` will draw it centered on the origin (horizontally)
- `setTextSkewX(float skewX)` This could be considered as fake italic. SkewX represents the horizontal offset of the text bottom. (use -0.25 for italic)
- `setStyle(Paint.Style style)` Fill text `FILL`, Stroke text `STROKE`, or both `FILL_AND_STROKE`

Note that you can use `TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())` to convert from SP or DP to pixels.

Measuring text

- `float width = paint.measureText(String text)` Measure the width of text
- `float height = paint.ascent()` Measure the height of text
- `paint.getTextBounds(String text, int start, int end, Rect bounds` Stores the text dimensions. You have allocate the Rect, it cannot be null:

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

There are other methods for measuring, however these three should fit most purposes.

Section 198.3: Setting up Paint for drawing shapes

- `setStyle(Paint.Style style)` Filled shape `FILL`, Stroke shape `STROKE`, or both `FILL_AND_STROKE`
- `setColor(int color)` Set the paint drawing color. You can also use `setARGB(int a, int r, int g, int b` and `setAlpha(int alpha)`
- `setStrokeCap(Paint.Cap cap)` Set line caps, either `ROUND`, `SQUARE`, or `BUTT` (none) See this.
- `setStrokeJoin(Paint.Join join)` Set line joins, either `MITER` (pointy), `ROUND`, or `BEVEL`. See this.
- `setStrokeMiter(float miter)` Set miter join limit. This can prevent miter join from going on indefinitively, turning it into a bevel join after x pixels. See this.
- `setStrokeWidth(float width)` Set stroke width. `0` will draw in hairline mode, independant of the canvas matrix. (always 1 pixel)

Section 198.4: Setting flags

You can set the following flags in the constructor, or with `setFlags(int flags)`

- `Paint.ANTI_ALIAS_FLAG` Enable antialiasing, smooths the drawing.
- `Paint.DITHER_FLAG` Enable dithering. If color precision is higher than the device's, this will happen.
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG` Enables the use of bitmap fonts.
- `Paint.FAKE_BOLD_TEXT_FLAG` will draw text with a fake bold effect, can be used instead of using a bold typeface. Some fonts have styled bold, fake bold won't
- `Paint.FILTER_BITMAP_FLAG` Affects the sampling of bitmaps when transformed.
- `Paint.HINTING_OFF`, `Paint.HINTING_ON` Toggles font hinting, see this
- `Paint.LINEAR_TEXT_FLAG` Disables font scaling, draw operations are scaled instead
- `Paint.SUBPIXEL_TEXT_FLAG` Text will be computed using subpixel accuracy.
- `Paint.STRIKE_THRU_TEXT_FLAG` Text drawn will be striked
- `Paint.UNDERLINE_TEXT_FLAG` Text drawn will be underlined

You can add a flag and remove flags like this:

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG);   // Add flag
paint.setFlags(paint.getFlags() & ~Paint.FLAG);  // Remove flag
```

Trying to remove a flag that isn't there or adding a flag that is already there won't change anything. Also note that most flags can also be set using `set<Flag>(boolean enabled)`, for example `setAntialias(true)`.

You can use `paint.reset()` to reset the paint to its default settings. The only default flag is `EMBEDDED_BITMAP_TEXT_FLAG`. It will be set even if you use `new Paint(0)`, you will have

Chapter 199: What is ProGuard? What is use in Android?

Proguard is free Java class file shrinker, optimizer, obfuscator, and preverifier. It detects and removes unused classes, fields, methods, and attributes. It optimizes bytecode and removes unused instructions. It renames the remaining classes, fields, and methods using short meaningless names.

Section 199.1: Shrink your code and resources with proguard

To make your APK file as small as possible, you should enable shrinking to remove unused code and resources in your release build. This page describes how to do that and how to specify what code and resources to keep or discard during the build.

Code shrinking is available with ProGuard, which detects and removes unused classes, fields, methods, and attributes from your packaged app, including those from included code libraries (making it a valuable tool for working around the 64k reference limit). ProGuard also optimizes the bytecode, removes unused code instructions, and obfuscates the remaining classes, fields, and methods with short names. The obfuscated code makes your APK difficult to reverse engineer, which is especially valuable when your app uses security-sensitive features, such as licensing verification.

Resource shrinking is available with the Android plugin for Gradle, which removes unused resources from your packaged app, including unused resources in code libraries. It works in conjunction with code shrinking such that once unused code has been removed, any resources no longer referenced can be safely removed as well.

Shrink Your Code

To enable code shrinking with `ProGuard`, add `minifyEnabled` true to the appropriate build type in your `build.gradle` file.

Be aware that code shrinking slows down the build time, so you should avoid using it on your debug build if possible. However, it's important that you do enable code shrinking on your final APK used for testing, because it might introduce bugs if you do not sufficiently customize which code to keep.

For example, the following snippet from a `build.gradle` file enables code shrinking for the release build:

```
android {
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                    'proguard-rules.pro'
        }
    }
    ...
}
```

In addition to the `minifyEnabled` property, the `proguardFiles` property defines the `ProGuard rules`:

The getDefaultProguardFile('proguard-android.txt') method gets the default ProGuard settings from the Android SDK `tools/proguard/ folder`. Tip: For even more code shrinking, try the `proguard-android-optimize.txt` file that's in the same location. It includes the same ProGuard rules, but with other optimizations that perform analysis at the bytecode level—inside and across methods—to reduce your APK size further and help it run faster. The `proguard-rules.pro` file is where you can add custom ProGuard rules. By default, this file is located at the root of

the module (next to the build.gradle file). To add more ProGuard rules that are specific to each build variant, add another proguardFiles property in the corresponding `productFlavor` block. For example, the following Gradle file adds flavor2-rules.pro to the flavor2 product flavor. Now flavor2 uses all three ProGuard rules because those from the release block are also applied.

```
android {
    ...
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                    'proguard-rules.pro'
        }
    }
    productFlavors {
        flavor1 {
        }
        flavor2 {
            proguardFile 'flavor2-rules.pro'
        }
    }
}
```

Chapter 200: Create Android Custom ROMs

Section 200.1: Making Your Machine Ready for Building!

Before you can build anything, you are required to make your machine ready for building. For this you need to install a lot of libraries and modules. The most recommended Linux distribution is Ubuntu, so this example will focus on installing everything that is needed on Ubuntu.

Installing Java

First, add the following Personal Package Archive (PPA): `sudo apt-add-repository ppa:openjdk-r/ppa`.

Then, update the sources by executing: `sudo apt-get update`.

Installing Additional Dependencies

All required additional dependencies can be installed by the following command:

```
sudo apt-get install git-core python gnupg flex bison gperf libsdl1.2-dev libesd0-dev libwxgtk2.8-
dev squashfs-tools build-essential zip curl libncurses5-dev zlib1g-dev openjdk-8-jre openjdk-8-jdk
pngcrush schedtool libxml2 libxml2-utils xsltproc lzop libc6-dev schedtool g++-multilib lib32z1-dev
lib32ncurses5-dev gcc-multilib liblz4-* pngquant ncurses-dev texinfo gcc gperf patch libtool
automake g++ gawk subversion expat libexpat1-dev python-all-dev binutils-static bc libcloog-isl-dev
libcap-dev autoconf libgmp-dev build-essential gcc-multilib g++-multilib pkg-config libmpc-dev
libmpfr-dev lzma* liblzma* w3m android-tools-adb maven ncftp figlet
```

Preparing the system for development

Now that all the dependencies are installed, let us prepare the system for development by executing:

```
sudo curl --create-dirs -L -o /etc/udev/rules.d/51-android.rules -O -L
https://raw.githubusercontent.com/snowdream/51-android/master/51-android.rules
sudo chmod 644   /etc/udev/rules.d/51-android.rules
sudo chown root /etc/udev/rules.d/51-android.rules
sudo service udev restart
adb kill-server
sudo killall adb
```

Finally, let us set up the cache and the repo by the following commands:

```
sudo install utils/repo /usr/bin/
sudo install utils/ccache /usr/bin/
```

Please note: We can also achieve this setup by running the automated scripts made by Akhil Narang (*akhilnarang*), one of the maintainers of Resurrection Remix OS. These scripts can be found on GitHub.

Chapter 201: Genymotion for android

Genymotion is a fast third-party emulator that can be used instead of the default Android emulator. In some cases it's as good as or better than developing on actual devices!

Section 201.1: Installing Genymotion, the free version

Step 1 - installing `VirtualBox`

Download and install VirtualBox according to your operating system. , it is required to run `Genymotion`.

Step 2 - downloading `Genymotion`

Go to the Genymotion download page and download `Genymotion` according to your operating system.

Note: you will need to create a new account OR log-in with your account.

Step 3 - Installing `Genymotion`

if on `Linux` then refer to this answer, to install and run a `.bin` file.

Step 4 - Installing `Genymotion`'s emulators

- run `Genymotion`
- Press on the Add button (in top bar).
- Log-In with your account and you will be able to browse the available emulators.
- select and Install what you need.

Step 5 - Integrating `genymotion` with `Android Studio`

`Genymotion`, can be integrated with `Android Studio` via a plugin, here the steps to install it in `Android Studio`

- go to File/Settings (for Windows and Linux) or to Android Studio/Preferences (for Mac OS X)
- Select Plugins and click Browse Repositories.
- Right-click on Genymotion and click Download and install.

You should now be able to see the plugin icon, see this image

Note, you might want to display the toolbar by clicking View > Toolbar.

Step 6 - Running `Genymotion` from `Android Studio`

- go to File/Settings (for Windows and Linux) or to Android Studio/Preferences (for Mac OS X)
- go to Other Settings/Genymotion and add the path of `Genymotion's` folder and apply your changes.

Now you should be able to run `Genymotion's` emulator by pressing the plugin icon and selecting an installed emulator and then press start button!

Section 201.2: Google framework on Genymotion

If developers want to test Google Maps or any other Google service like Gmail,Youtube, Google drive etc. then they first need to install Google framework on Genymotion. Here are the steps:

4.4 Kitkat
5.0 Lollipop
5.1 Lollipop
6.0 Marshmallow
7.0 Nougat
7.1 Nougat (webview patch)

1. Download from above link
2. Just drag & drop downloaded zip file to genymotion and restart
3. Add google account and download "Google Play Music" and Run.

Reference:
Stack overflow question on this topic

Chapter 202: ConstraintSet

This class allows you to define programmatically a set of constraints to be used with `ConstraintLayout`. It lets you create and save constraints, and apply them to an existing `ConstraintLayout`.

Section 202.1: ConstraintSet with ContraintLayout Programmatically

```
import android.content.Context;
import android.os.Bundle;
import android.support.constraint.ConstraintLayout;
import android.support.constraint.ConstraintSet;
import android.support.transition.TransitionManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    ConstraintSet mConstraintSet1 = new ConstraintSet(); // create a Constraint Set
    ConstraintSet mConstraintSet2 = new ConstraintSet(); // create a Constraint Set
    ConstraintLayout mConstraintLayout; // cache the ConstraintLayout
    boolean mOld = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Context context = this;
        mConstraintSet2.clone(context, R.layout.state2); // get constraints from layout
        setContentView(R.layout.state1);
        mConstraintLayout = (ConstraintLayout) findViewById(R.id.activity_main);
        mConstraintSet1.clone(mConstraintLayout); // get constraints from ConstraintSet
    }

    public void foo(View view) {
        TransitionManager.beginDelayedTransition(mConstraintLayout);
        if (mOld = !mOld) {
            mConstraintSet1.applyTo(mConstraintLayout); // set new constraints
        }  else {
            mConstraintSet2.applyTo(mConstraintLayout); // set new constraints
        }
    }
}
```

Chapter 203: CleverTap

Quick hacks for the analytics and engagement SDK provided by CleverTap - Android

Section 203.1: Setting the debug level

In your custom application class, override the `onCreate()` method, add the line below:

```
CleverTapAPI.setDebugLevel(1);
```

Section 203.2: Get an instance of the SDK to record events

```
CleverTapAPI cleverTap;
try {
  cleverTap = CleverTapAPI.getInstance(getApplicationContext());
} catch (CleverTapMetaDataNotFoundException e) {
  // thrown if you haven't specified your CleverTap Account ID or Token in your AndroidManifest.xml
} catch (CleverTapPermissionsNotSatisfied e) {
  // thrown if you haven't requested the required permissions in your AndroidManifest.xml
}
```

Chapter 204: Publish a library to Maven Repositories

Section 204.1: Publish .aar file to Maven

In order to publish to a repository in Maven format ,“maven-publish” plugin for gradle can be used.

The plugin should be added to `build.gradle` file in library module.

```
apply plugin: 'maven-publish'
```

You should define the publication and its identity attributes in `build.gradle` file too. This identity attributes will be shown in the generated pom file and in future for importing this publication you will use them.You also need to define which artifacts you want to publish,for example i just want to publish generated .aar file after building the library.

```
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
}
```

You will also need to define your repository url

```
publishing{
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

Here is full library `build.gradle` file

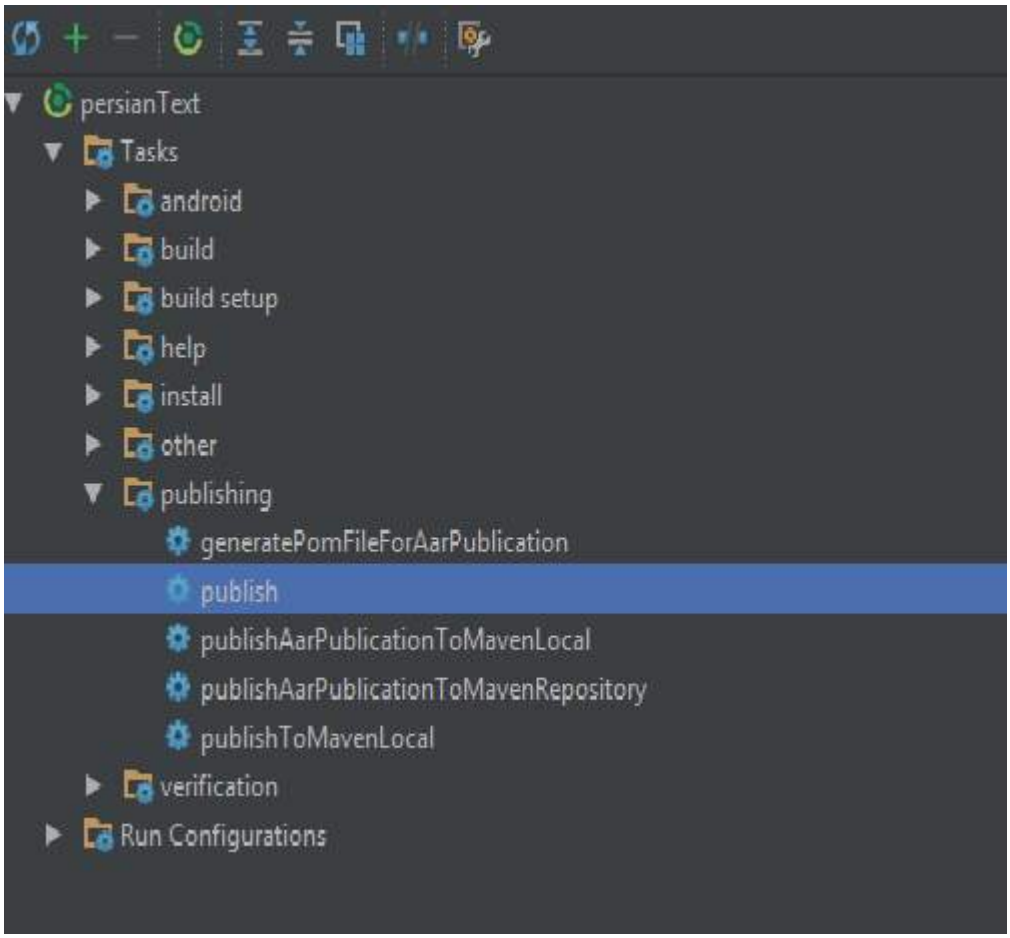
```
apply plugin: 'com.android.library'
apply plugin: 'maven-publish'

buildscript {
   ...
}
android {
    ...
}
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
```

```
    }
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

For publishing you can run gradle console command

> gradle publish

or you can run from gradle tasks panel



Chapter 205: adb shell

Parameter	Details
-e	choose escape character, or "none"; default '~'
-n	don't read from stdin
-T	disable PTY allocation
-t	force PTY allocation
-x	disable remote exit codes and stdout/stderr separation

`adb shell` opens a Linux shell in a target device or emulator. It is the most powerful and versatile way to control an Android device via `adb`.

This topic was split from ADB (Android Debug Bridge) due to reaching the limit of examples, many of which were involving `adb shell` command.

Section 205.1: Granting & revoking API 23+ permissions

A one-liner that helps granting or revoking vulnerable permissions.

- **granting**

```
adb shell pm grant <sample.package.id> android.permission.<PERMISSION_NAME>
```

- **revoking**

```
adb shell pm revoke <sample.package.id> android.permission.<PERMISSION_NAME>
```

- **Granting all run-time permissions at a time on installation (-g)**

```
adb install -g /path/to/sample_package.apk
```

Section 205.2: Send text, key pressed and touch events to Android Device via ADB

execute the following command to insert the text into a view with a focus (if it supports text input)

`Version ≥ 6.0`

Send text on SDK 23+

```
adb shell "input keyboard text 'Paste text on Android Device'"
```

If already connected to your device via `adb`:

```
input text 'Paste text on Android Device'
```

`Version < 6.0`

Send text prior to SDK 23

```
adb shell "input keyboard text 'Paste%stext%son%sAndroid%sDevice'"
```

Spaces are not accepted as the input, replace them with %s.

Send events

To simulate pressing the hardware power key

```
adb shell input keyevent 26
```

or alternatively

```
adb shell input keyevent POWER
```

Even if you don't have a hardware key you still can use a `keyevent` to perform the equivalent action

```
adb shell input keyevent CAMERA
```

Send touch event as input

```
adb shell input tap Xpoint Ypoint
```

Send swipe event as input

```
adb shell input swipe Xpoint1 Ypoint1 Xpoint2 Ypoint2 [DURATION*]
```

*DURATION is optional, default=300ms. source

Get X and Y points by enabling pointer location in developer option.

ADB sample shell script

> To run a script in Ubuntu, Create script.sh right click the file and add read/write permission and tick **allow executing file as program**.

> Open terminal emulator and run the command ./script.sh

Script.sh

```
for (( c=1; c<=5; c++ ))
   do
      adb shell input tap X Y
      echo "Clicked $c times"
      sleep 5s
   done
```

For a comprehensive list of event numbers

- shortlist of several interesting events ADB Shell Input Events
- reference documentation https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER.

Section 205.3: List packages

Prints all packages, optionally only those whose package name contains the text in <FILTER>.

```
adb shell pm list packages [options] <FILTER>

All <FILTER>

adb shell pm list packages
```

Attributes:

`-f` to see their associated file.

`-i` See the installer for the packages.

`-u` to also include uninstalled packages.

`-u` Also include uninstalled packages.

Attributes that filter:

`-d` for disabled packages.

`-e` for enabled packages.

`-s` for system packages.

`-3` for third party packages.

`--user <USER_ID>` for a specific user space to query.

Section 205.4: Recording the display

`Version ≥ 4.4`

Recording the display of devices running Android 4.4 (API level 19) and higher:

```
adb shell screenrecord [options] <filename>
adb shell screenrecord /sdcard/demo.mp4
```

(press Ctrl-C to stop recording)

Download the file from the device:

```
adb pull /sdcard/demo.mp4
```

> Note: Stop the screen recording by pressing Ctrl-C, otherwise the recording stops automatically at three minutes or the time limit set by `--time-limit`.

```
adb shell screenrecord --size <WIDTHxHEIGHT>
```

Sets the video size: 1280x720. The default value is the device's native display resolution (if supported), 1280x720 if not. For best results, use a size supported by your device's Advanced Video Coding (AVC) encoder.

```
adb shell screenrecord --bit-rate <RATE>
```

Sets the video bit rate for the video, in megabits per second. The default value is 4Mbps. You can increase the bit rate to improve video quality, but doing so results in larger movie files. The following example sets the recording bit rate to 5Mbps:

```
adb shell screenrecord --bit-rate 5000000 /sdcard/demo.mp4
```

```
adb shell screenrecord --time-limit <TIME>
```

Sets the maximum recording time, in seconds. The default and maximum value is 180 (3 minutes).

```
adb shell screenrecord --rotate
```

Rotates the output 90 degrees. This feature is experimental.

```
adb shell screenrecord --verbose
```

Displays log information on the command-line screen. If you do not set this option, the utility does not display any information while running.

> Note: This might not work on some devices.

`Version < 4.4`

The screen recording command isn't compatible with android versions pre 4.4

> The screenrecord command is a shell utility for recording the display of devices running Android 4.4 (API level 19) and higher. The utility records screen activity to an MPEG-4 file.

Section 205.5: Open Developer Options

```
adb shell am start -n com.android.settings/.DevelopmentSettings
```

Will navigate your device/emulator to the `Developer Options` section.

Section 205.6: Set Date/Time via adb

`Version ≥ 6.0`

Default SET format is `MMDDhhmm[[CC]YY][.ss]`, that's (2 digits each)

For example, to set July 17'th 10:10am, without changing the current year, type:

```
adb shell 'date 07171010.00'
```

Tip 1: the date change will not be reflected immediately, and a noticable change will happen only after the system clock advances to the next minute.

You can force an update by attaching a `TIME_SET` intent broadcast to your call, like that:

```
adb shell 'date 07171010.00 ; am broadcast -a android.intent.action.TIME_SET'
```

Tip 2: to synchronize Android's clock with your local machine:

Linux:

```
adb shell date `date +%m%d%H%M%G.%S`
```

Windows (PowerShell):

```
$currentDate = Get-Date -Format "MMddHHmmyyyy.ss" # Android's preferred format
adb shell "date $currentDate"
```

Both tips together:

```
adb shell 'date `date +%m%d%H%M%G.%S` ; am broadcast -a android.intent.action.TIME_SET'
```

`Version < 6.0`

Default SET format is 'YYYYMMDD.HHmmss'

```
adb shell 'date -s 20160117.095930'
```

Tip: to synchronize Android's clock with your local (linux based) machine:

```
adb shell date -s `date +%G%m%d.%H%M%S`
```

Section 205.7: Generating a "Boot Complete" broadcast

This is relevant for apps that implement a `BootListener`. Test your app by killing your app and then test with:

```
adb shell am broadcast -a android.intent.action.BOOT_COMPLETED -c android.intent.category.HOME -n
your.app/your.app.BootListener
```

(replace `your.package/your.app.BootListener` with proper values).

Section 205.8: Print application data

This command print all relevant application data:

- version code
- version name
- granted permissions (Android API 23+)
- etc..

```
adb shell dumpsys package <your.package.id>
```

Section 205.9: Changing file permissions using chmod command

Notice, that in order to change file prmissions, your device need to be rooted, *`su` binary doesn't come with factory shipped devices!*

Convention:

```
adb shell su -c "chmod <numeric-permisson> <file>"
```

Numeric permission constructed from user, group and world sections.

For example, if you want to change file to be readable, writable and executable by everyone, this will be your command:

```
adb shell su -c "chmod 777 <file-path>"
```

Or

```
adb shell su -c "chmod 000 <file-path>"
```

if you intent to deny any permissions to it.

1st digit-specifies user permission, **2nd digit**- specifies group permission, **3rd digit** - specifies world (others) permission.

Access permissions:

```
--- :   binary value:   000,  octal value: 0 (none)
--x :   binary value:   001,  octal value: 1 (execute)
-w- :   binary value:   010,  octal value: 2 (write)
-wx :   binary value:   011,  octal value: 3 (write, execute)
r-- :   binary value:   100,  octal value: 4 (read)
r-x :   binary value:   101,  octal value: 5 (read, execute)
rw- :   binary value:   110,  octal value: 6 (read, write)
rwx :   binary value:   111,  octal value: 7 (read, write, execute)
```

Section 205.10: View external/secondary storage content

View content:

```
adb shell ls \$EXTERNAL_STORAGE
adb shell ls \$SECONDARY_STORAGE
```

View path:

```
adb shell echo \$EXTERNAL_STORAGE
adb shell echo \$SECONDARY_STORAGE
```

Section 205.11: kill a process inside an Android device

Sometimes Android's logcat is running infinitely with errors coming from some process not own by you, draining battery or just making it hard to debug your code.

A convenient way to fix the problem without restarting the device is to locate and kill the process causing the problem.

From Logcat

```
03-10 11:41:40.010 1550-1627/? E/SomeProcess: ....
```

notice the process number: 1550

Now we can open a shell and kill the process. Note that we cannot kill `root` process.

```
adb shell
```

inside the shell we can check more about the process using

```
ps -x | grep 1550
```

and kill it if we want:

```
kill -9 1550
```

Chapter 206: Ping ICMP

The ICMP Ping request can be performed in Android by creating a new process to run the ping request. The outcome of the request can be evaluated upon the completion of the ping request from within its process.

Section 206.1: Performs a single Ping

This example attempts a single Ping request. The ping command inside the `runtime.exec` method call can be modified to any valid ping command you might perform yourself in the command line.

```
try {
    Process ipProcess = runtime.exec("/system/bin/ping -c 1 8.8.8.8");
    int exitValue = ipProcess.waitFor();
    ipProcess.destroy();

    if(exitValue == 0){
        // Success
    } else {
        // Failure
    }
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}
```

Chapter 207: AIDL

AIDL is Android interface definition language.

What? Why? How ?

What? It is a bounded services. This AIDL service will be active till atleast one of the client is exist. It works based on marshaling and unmarshaling concept.

Why? Remote applications can access your service + Multi Threading.(Remote application request).

How? Create the .aidl file Implement the interface Expose the interface to clients

Section 207.1: AIDL Service

> ICalculator.aidl

```
// Declare any non-default types here with import statements

    interface ICalculator {
        int add(int x,int y);
        int sub(int x,int y);
    }
```

> AidlService.java

```
public class AidlService extends Service {

    private static final String TAG = "AIDLServiceLogs";
    private static final String className = " AidlService";

    public AidlService() {
        Log.i(TAG, className+" Constructor");
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        Log.i(TAG, className+" onBind");
        return iCalculator.asBinder();
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.i(TAG, className+" onCreate");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.i(TAG, className+" onDestroy");
    }
```

```
    ICalculator.Stub iCalculator = new ICalculator.Stub() {
        @Override
        public int add(int x, int y) throws RemoteException {
            Log.i(TAG, className+" add Thread Name: "+Thread.currentThread().getName());
            int z = x+y;
            return z;
        }

        @Override
        public int sub(int x, int y) throws RemoteException {
            Log.i(TAG, className+" add Thread Name: "+Thread.currentThread().getName());
            int z = x-y;
            return z;
        }
    };

}
```

> Service Connection

```
 // Return the stub as interface
ServiceConnection serviceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            Log.i(TAG, className + " onServiceConnected");
            iCalculator = ICalculator.Stub.asInterface(service);
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {

            unbindService(serviceConnection);
        }
    };
```

Chapter 208: Android game development

A short introduction to creating a game on the Android platform using Java

Section 208.1: Game using Canvas and SurfaceView

This covers how you can create a basic 2D game using SurfaceView.

First, we need an activity:

```
public class GameLauncher extends AppCompatActivity {

    private Game game;
    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        game = new Game(GameLauncher.this);//Initialize the game instance
        setContentView(game);//setContentView to the game surfaceview
        //Custom XML files can also be used, and then retrieve the game instance using findViewById.
    }

}
```

The activity also has to be declared in the Android Manifest.

Now for the game itself. First, we start by implementing a game thread:

```
public class Game extends SurfaceView implements SurfaceHolder.Callback, Runnable{

    /**
     * Holds the surface frame
     */
    private SurfaceHolder holder;

    /**
     * Draw thread
     */
    private Thread drawThread;

    /**
     * True when the surface is ready to draw
     */
    private boolean surfaceReady = false;


    /**
     * Drawing thread flag
     */

    private boolean drawingActive = false;

    /**
     * Time per frame for 60 FPS
     */
    private static final int MAX_FRAME_TIME = (int) (1000.0 / 60.0);

    private static final String LOGTAG = "surface";

    /*
```

```
     * All the constructors are overridden to ensure functionality if one of the different
constructors are used through an XML file or programmatically
     */
    public Game(Context context) {
        super(context);
        init();
    }
    public Game(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }
    public Game(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }
    @TargetApi(21)
    public Game(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
        super(context, attrs, defStyleAttr, defStyleRes);
        init();
    }

    public void init(Context c) {
        this.c = c;

        SurfaceHolder holder = getHolder();
        holder.addCallback(this);
        setFocusable(true);
        //Initialize other stuff here later
    }

    public void render(Canvas c){
        //Game rendering here
    }

    public void tick(){
        //Game logic here
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
    {
        if (width == 0 || height == 0){
            return;
        }

        // resize your UI
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder){
        this.holder = holder;

        if (drawThread != null){
            Log.d(LOGTAG, "draw thread still active..");
            drawingActive = false;
            try{
                drawThread.join();
            } catch (InterruptedException e){}
        }

        surfaceReady = true;
        startDrawThread();
```

```
        Log.d(LOGTAG, "Created");
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder){
        // Surface is not used anymore - stop the drawing thread
        stopDrawThread();
        // and release the surface
        holder.getSurface().release();

        this.holder = null;
        surfaceReady = false;
        Log.d(LOGTAG, "Destroyed");
    }

    @Override
    public boolean onTouchEvent(MotionEvent event){
        // Handle touch events
        return true;
    }

    /**
     * Stops the drawing thread
     */
    public void stopDrawThread(){
        if (drawThread == null){
            Log.d(LOGTAG, "DrawThread is null");
            return;
        }
        drawingActive = false;
        while (true){
            try{
                Log.d(LOGTAG, "Request last frame");
                drawThread.join(5000);
                break;
            } catch (Exception e) {
                Log.e(LOGTAG, "Could not join with draw thread");
            }
        }
        drawThread = null;
    }

    /**
     * Creates a new draw thread and starts it.
     */
    public void startDrawThread(){
        if (surfaceReady && drawThread == null){
            drawThread = new Thread(this, "Draw thread");
            drawingActive = true;
            drawThread.start();
        }
    }

    @Override
    public void run() {
        Log.d(LOGTAG, "Draw thread started");
        long frameStartTime;
        long frameTime;

        /*
         * In order to work reliable on Nexus 7, we place ~500ms delay at the start of drawing thread
         * (AOSP - Issue 58385)
```

```
         */
        if (android.os.Build.BRAND.equalsIgnoreCase("google") &&
android.os.Build.MANUFACTURER.equalsIgnoreCase("asus") &&
android.os.Build.MODEL.equalsIgnoreCase("Nexus 7")) {
            Log.w(LOGTAG, "Sleep 500ms (Device: Asus Nexus 7)");
            try {
                Thread.sleep(500);
            } catch (InterruptedException ignored) {}
        }

        while (drawing) {
            if (sf == null) {
                return;
            }

            frameStartTime = System.nanoTime();
            Canvas canvas = sf.lockCanvas();
            if (canvas != null) {
                try {
                    synchronized (sf) {
                        tick();
                        render(canvas);
                    }
                } finally {

                    sf.unlockCanvasAndPost(canvas);
                }
            }

            // calculate the time required to draw the frame in ms
            frameTime = (System.nanoTime() - frameStartTime) / 1000000;

            if (frameTime < MAX_FRAME_TIME){
                try {
                    Thread.sleep(MAX_FRAME_TIME - frameTime);
                } catch (InterruptedException e) {
                    // ignore
                }
            }

        }
        Log.d(LOGTAG, "Draw thread finished");
    }
}
```

That is the basic part. Now you have the ability to draw onto the screen.

Now, let's start by adding to integers:

```
public final int x = 100;//The reason for this being static will be shown when the game is runnable
public int y;
public int velY;
```

For this next part, you are going to need an image. It should be about 100x100 but it can be bigger or smaller. For learning, a Rect can also be used(but that requires change in code a little bit down)

Now, we declare a Bitmap:

```
private Bitmap PLAYER_BMP = BitmapFactory.decodeResource(getResources(),
R.drawable.my_player_drawable);
```

In render, we need to draw this bitmap.

```
...
c.drawBitmap(PLAYER_BMP, x, y, null);
...
```

BEFORE LAUNCHING there are still some things to be done

We need a boolean first:

```
boolean up = false;
```

in onTouchEvent, we add:

```
if(ev.getAction() == MotionEvent.ACTION_DOWN){
    up = true;
}else if(ev.getAction() == MotionEvent.ACTION_UP){
    up = false;
}
```

And in tick we need this to move the player:

```
if(up){
    velY -=1;
}
else{
    velY +=1;
}
if(velY >14)velY = 14;
if(velY <-14)velY = -14;
y += velY *2;
```

and now we need this in init:

```
WindowManager wm = (WindowManager) c.getSystemService(Context.WINDOW_SERVICE);
Display display = wm.getDefaultDisplay();
Point size = new Point();
display.getSize(size);
WIDTH = size.x;
HEIGHT = size.y;
y = HEIGHT/ 2 - PLAYER_BMP.getHeight();
```

And we need these to variables:

```
public static int WIDTH, HEIGHT;
```

At this point, the game is runnable. Meaning you can launch it and test it.

Now you should have a player image or rect going up and down the screen. The player can be created as a custom class if needed. Then all the player-related things can be moved into that class, and use an instance of that class to move, render and do other logic.

Now, as you probably saw under testing it flies off the screen. So we need to limit it.

First, we need to declare the Rect:

```
private Rect screen;
```

In init, after initializing width and height, we create a new rect that is the screen.

```
screen = new Rect(0,0,WIDTH,HEIGHT);
```

Now we need another rect in the form of a method:

```
private Rect getPlayerBound(){
    return new Rect(x, y, x + PLAYER_BMP.getWidth(), y + PLAYER_BMP.getHeight();
}
```

and in tick:

```
if(!getPlayerBound().intersects(screen){
    gameOver = true;
}
```

The implementation of gameOVer can also be used to show the start of a game.

Other aspects of a game worth noting:

Saving(currently missing in documentation)

Chapter 209: Android programming with Kotlin

Using Kotlin with Android Studio is an easy task as Kotlin is developed by JetBrains. It is the same company that stands behind IntelliJ IDEA - a base IDE for Android Studio. That is why there are almost none problems with the compatibility.

Section 209.1: Installing the Kotlin plugin

First, you'll need to install the Kotlin plugin.

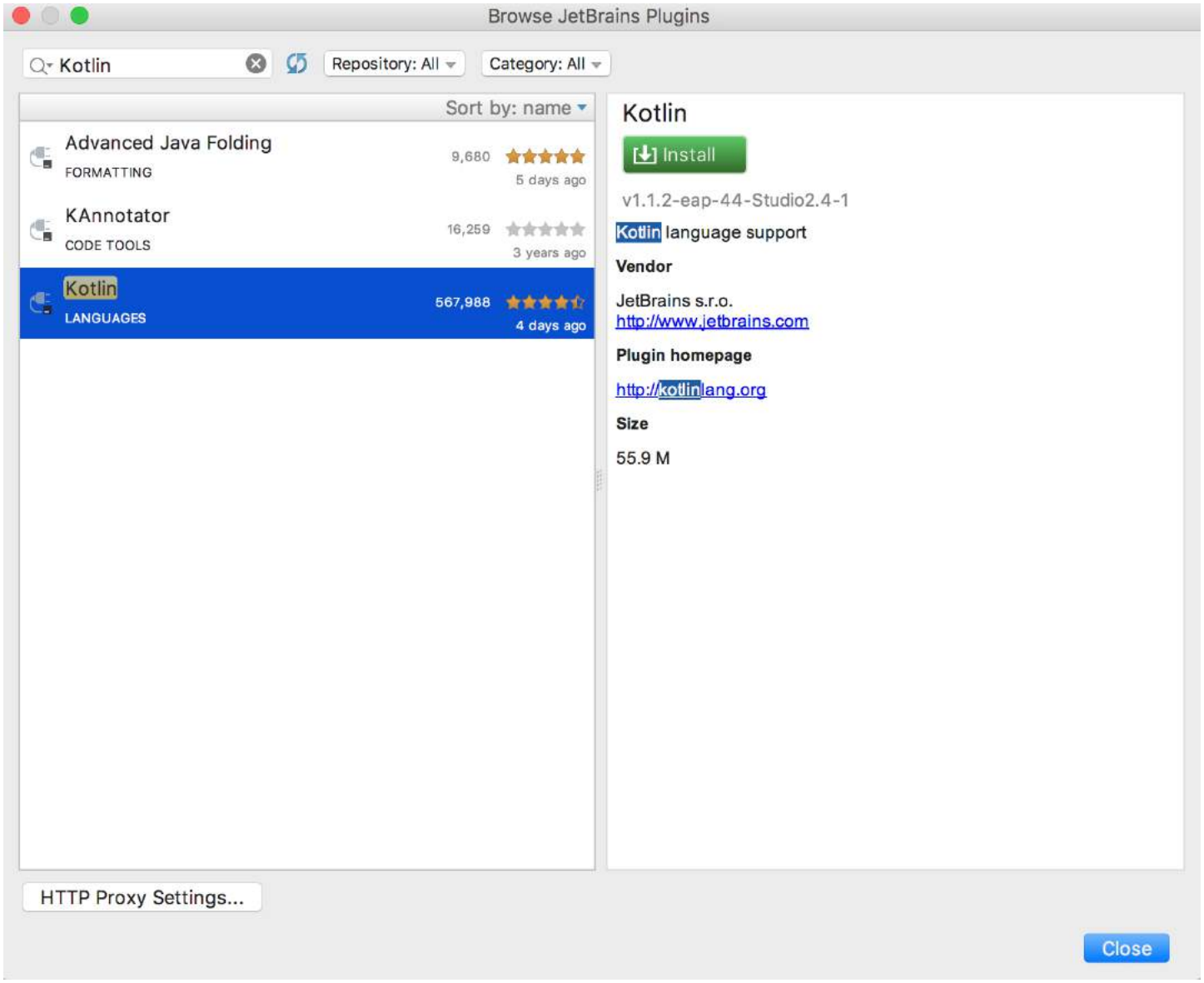
For Windows:

- Navigate to `File` → `Settings` → `Plugins` → `Install JetBrains plugin`

For Mac:

- Navigate to `Android Studio` → `Preferences` → `Plugins` → `Install JetBrains plugin`

And then search for and install Kotlin. You'll need to restart the IDE after this completes.



Section 209.2: Configuring an existing Gradle project with Kotlin

You can create a New Project in Android Studio and then add Kotlin support to it or modify your existing project. To do it, you have to:

1. **Add dependency to a root gradle file** - you have to add the dependency for `kotlin-android` plugin to a root `build.gradle` file.

```
buildscript {

    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.1'
        classpath 'org.jetbrains.kotlin:kotlin-gradle-plugin:1.1.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

2. **Apply Kotlin Android Plugin** - simply add `apply plugin: 'kotlin-android'` to a module `build.gradle` file.
3. **Add dependency to Kotlin stdlib** - add the dependency to `'org.jetbrains.kotlin:kotlin-stdlib:1.1.2'` to the dependency section in a module `build.gradle` file.

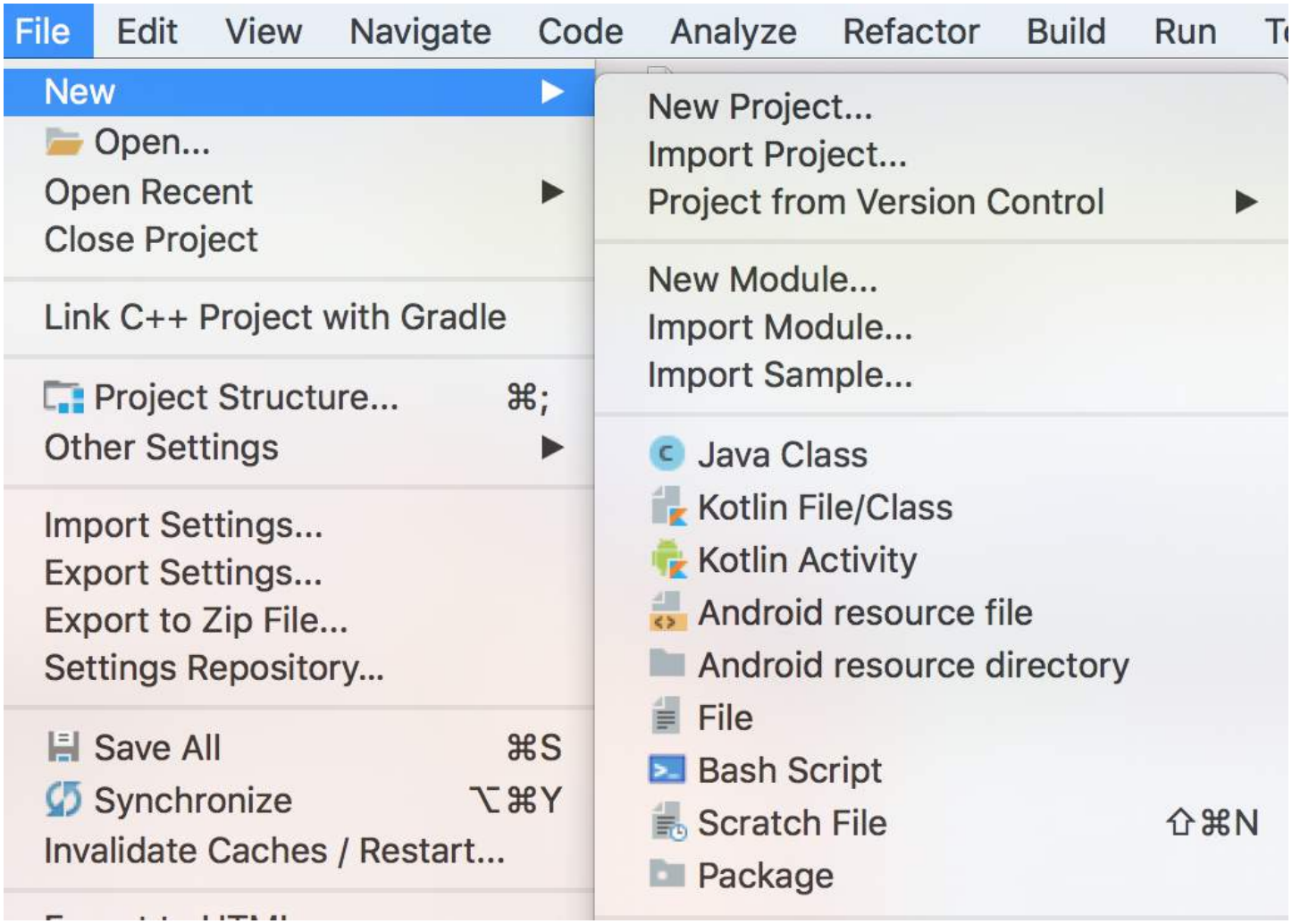
For a new project, `build.gradle` file could looks like this:

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "org.example.example"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
```

```
    compile 'org.jetbrains.kotlin:kotlin-stdlib:1.1.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.android.support:appcompat-v7:25.3.1'

    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })

    testCompile 'junit:junit:4.12'
}
```

Section 209.3: Creating a new Kotlin Activity

1. Click to `File` → **`New`** → `Kotlin Activity`.
2. Choose a type of the Activity.
3. Select name and other parameter for the Activity.
4. Finish.



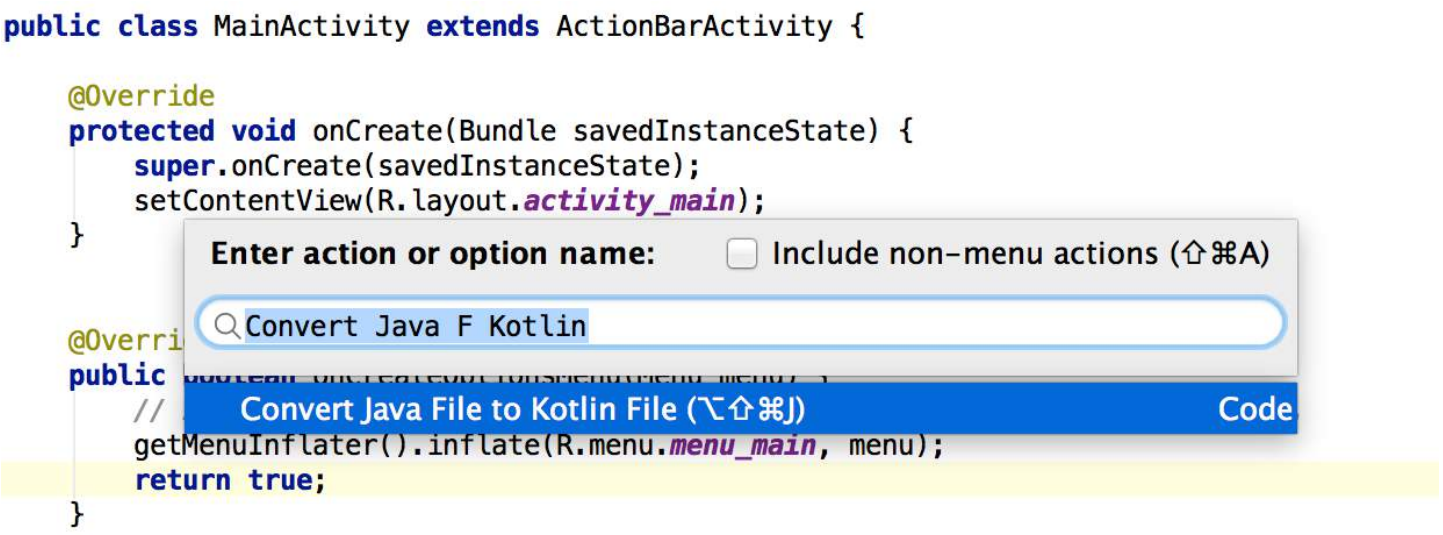
Final class could look like this:

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
    }
}
```

Section 209.4: Converting existing Java code to Kotlin

Kotlin Plugin for Android Studio support converting existing Java files to Kotlin files. Choose a Java file and invoke action Convert Java File to Kotlin File:



Section 209.5: Starting a new Activity

```
fun startNewActivity(){
    val intent: Intent = Intent(context, Activity::class.java)
    startActivity(intent)
}
```

You can add extras to the intent just like in Java.

```
fun startNewActivityWithIntents(){
    val intent: Intent = Intent(context, Activity::class.java)
    intent.putExtra(KEY_NAME, KEY_VALUE)
    startActivity(intent)
}
```

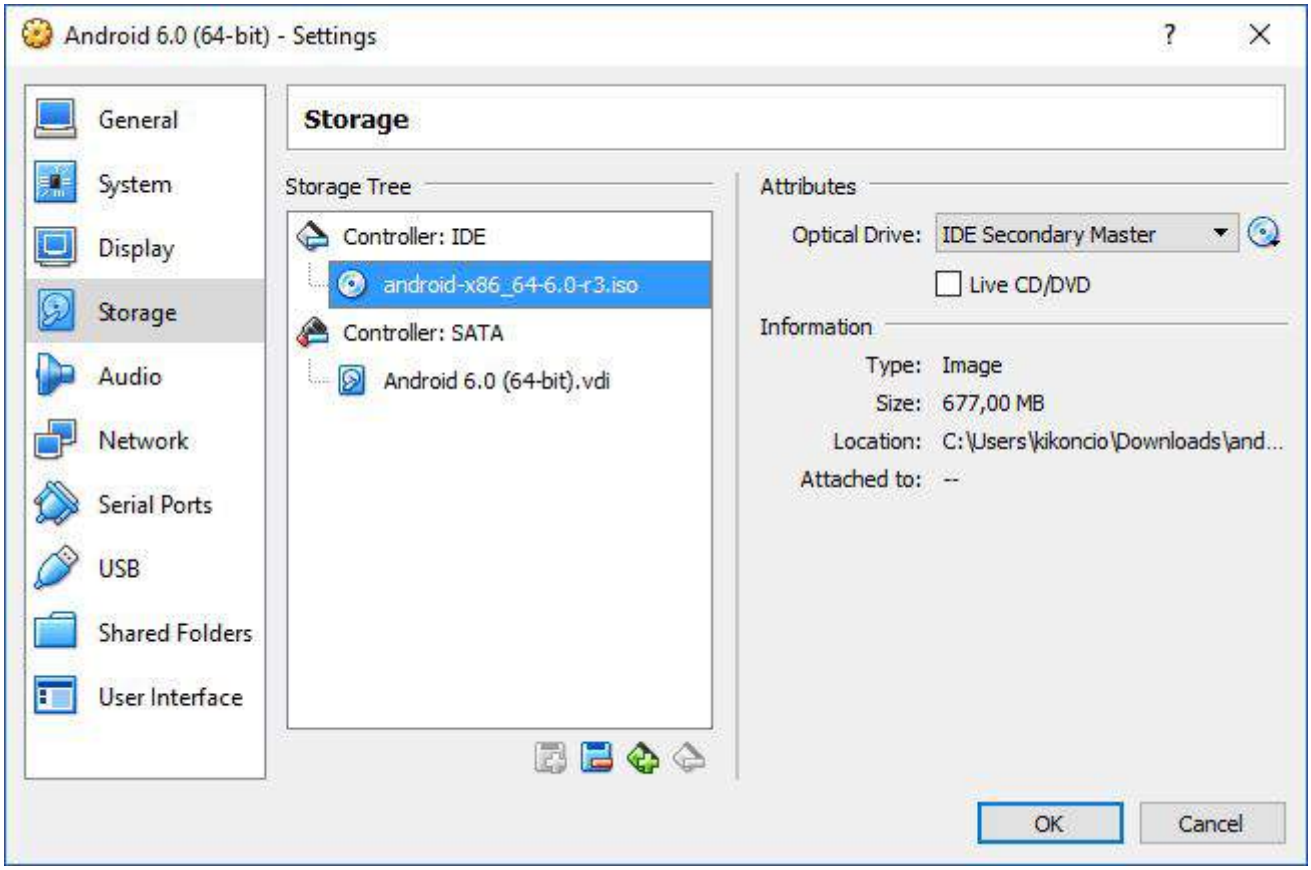
Chapter 210: Android-x86 in VirtualBox

The idea of this section is to cover how to install and use the VirtualBox with Android-x86 for debugging purposes. This is a difficult task because there are differences between versions. For the moment I´m going to cover 6.0 which is the one that I had to work with and then we'll have to find similarities.

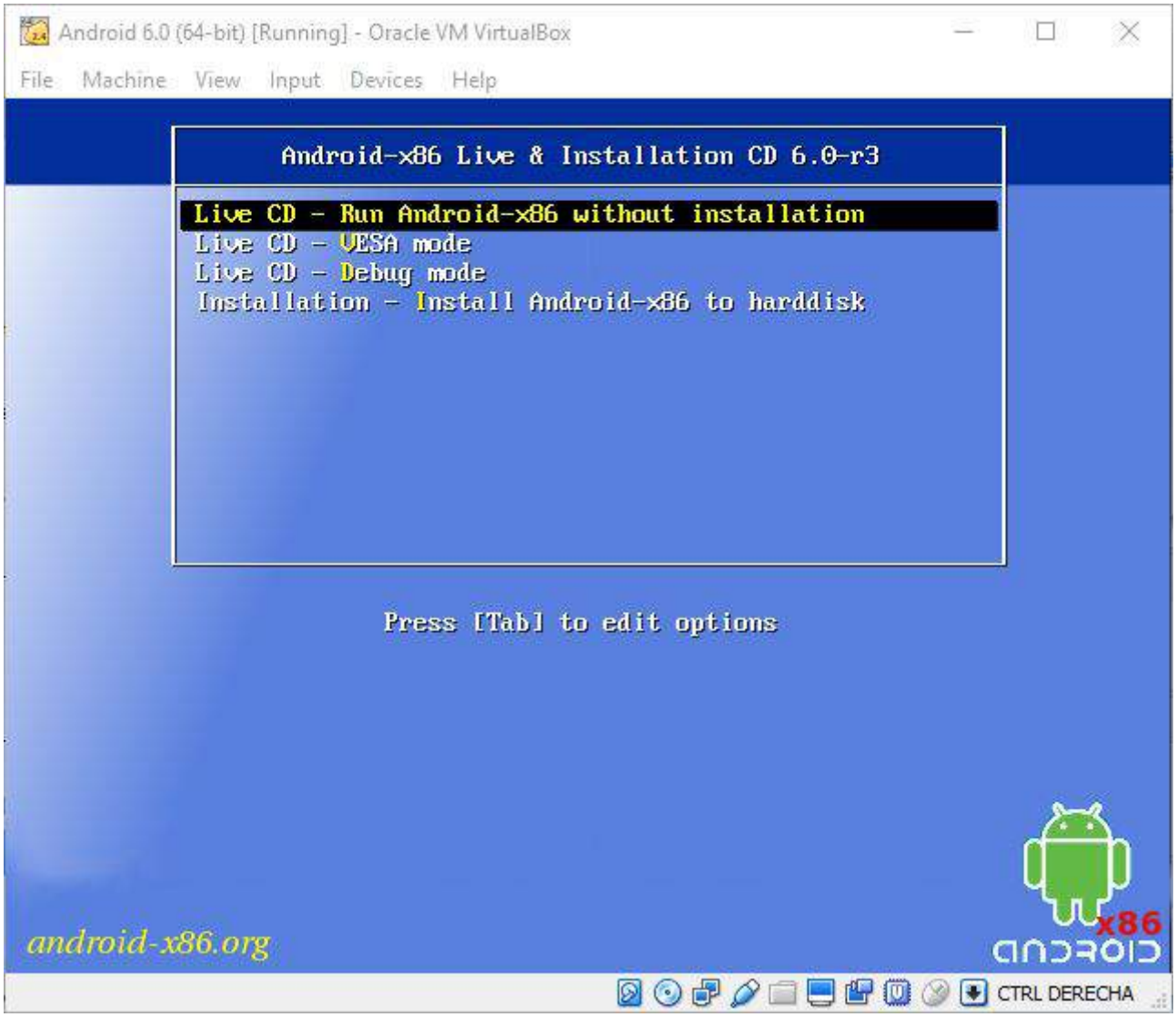
It doesn't cover VirtualBox or a Linux in detail but it shows the commands I've used to make it work.

Section 210.1: Virtual hard drive Setup for SDCARD Support

With the virtual hard drive just created, boot the virtual machine with the android-x86 image in the optical drive.



Once you boot, you can see the grub menu of the Live CD



Choose the Debug Mode Option, then you should see the shell prompt. This is a busybox shell. You can get more shell by switching between virtual console Alt-F1/F2/F3.

Create two partitions by fdisk (some other versions would use cfdisk). Format them to ext3. Then reboot:

```
# fdisk /dev/sda
```

Then type:

"n" (new partition)

"p" (primary partition)

"1" (1st partition)

"1" (first cylinder)

"261" (choose a cylinder, we'll leave 50% of the disk for a 2nd partition)

"2" (2nd partition)

"262" (262nd cylinder)

"522" (choose the last cylinder)

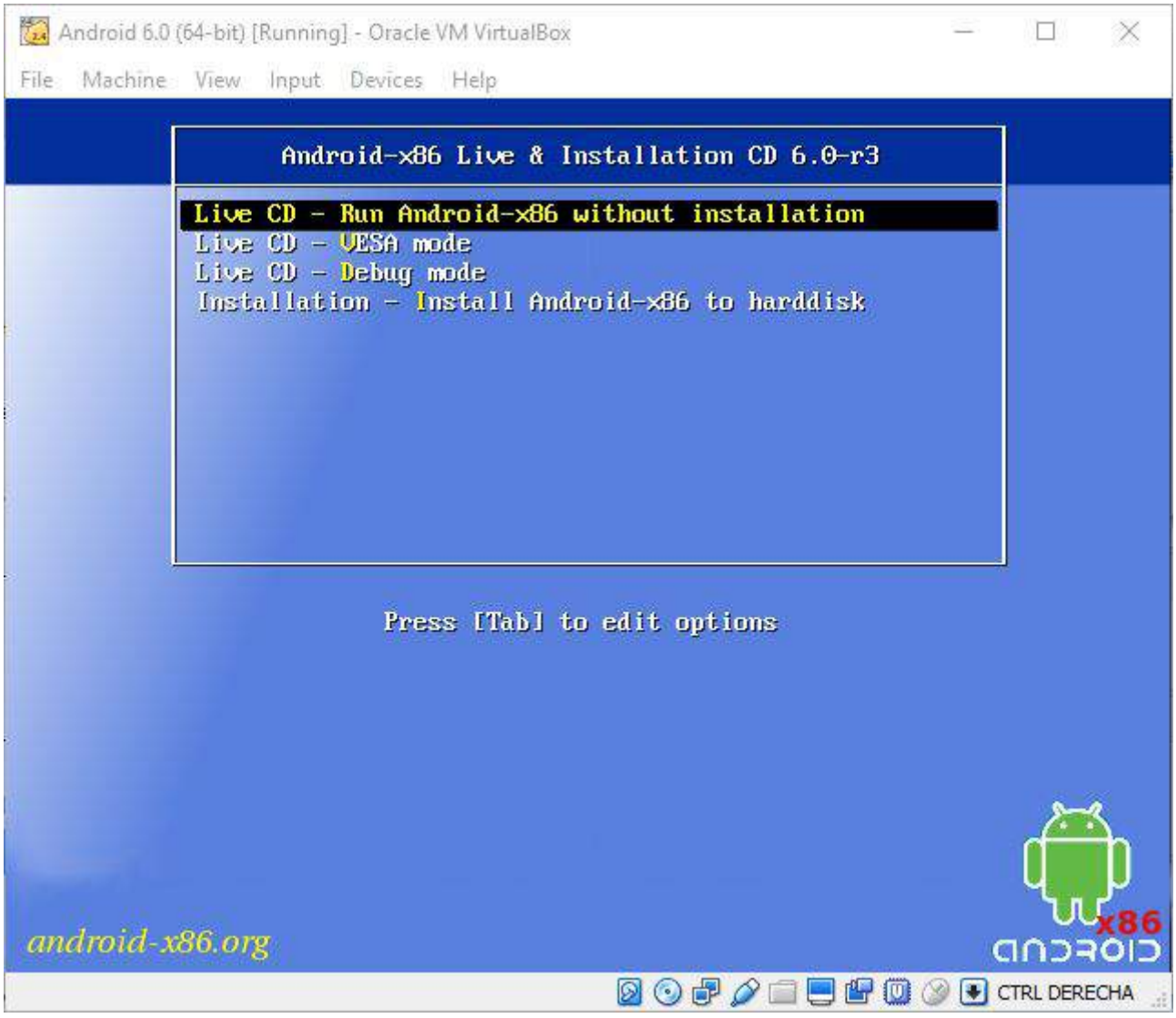
"w" (write the partition)

```
#mdev -s
#mke2fs -j -L DATA /dev/sda1
#mke2fs -j -L SDCARD /dev/sda2
#reboot -f
```

When you restart the virtual machine and the grub menu appears and you will be able edit the kernel boot line so you can add `DATA=sda1 SDCARD=sda2` options to point to the sdcard or the data partition.

Section 210.2: Installation in partition

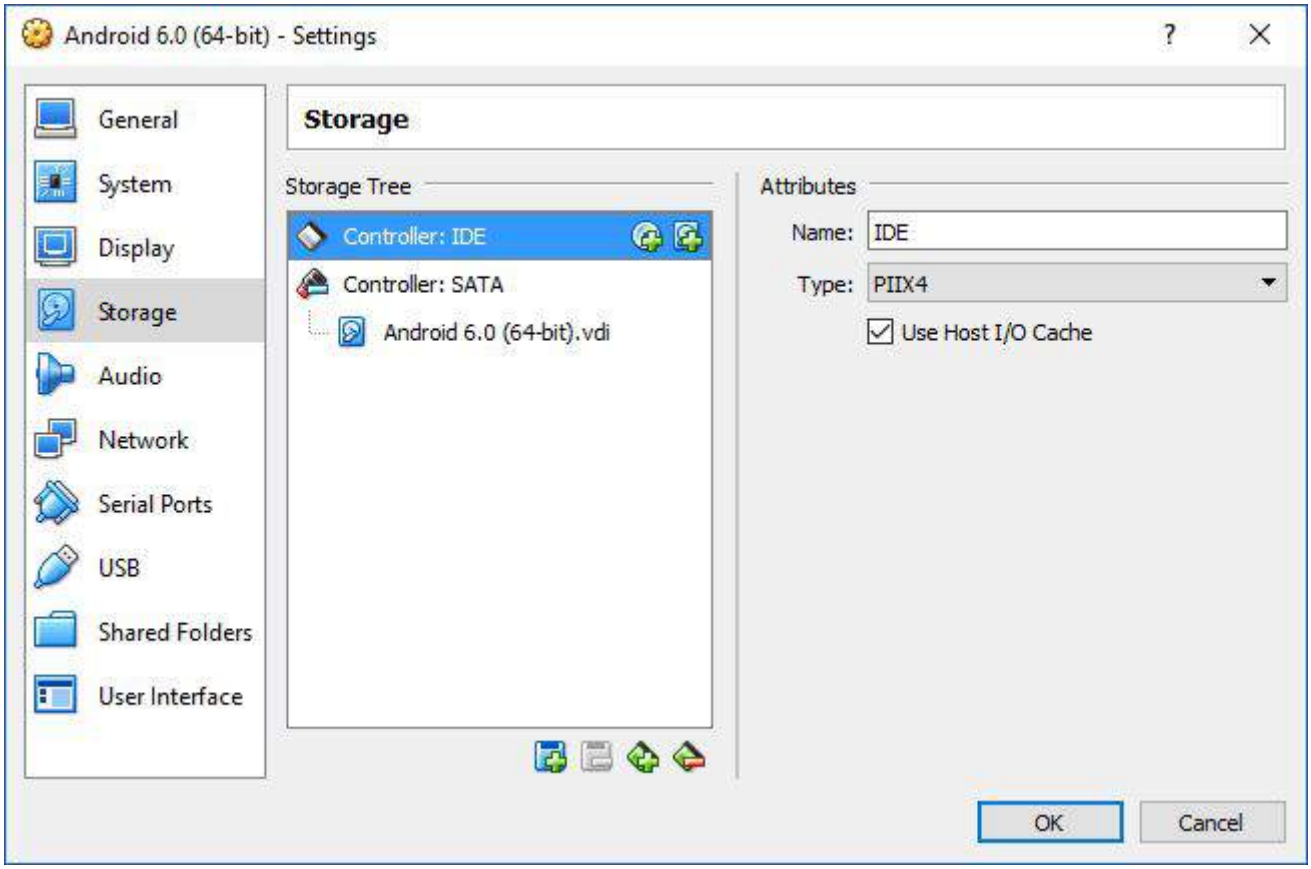
With the virtual hard drive just created, boot the virtual machine with the android-x86 image as the optical drive.



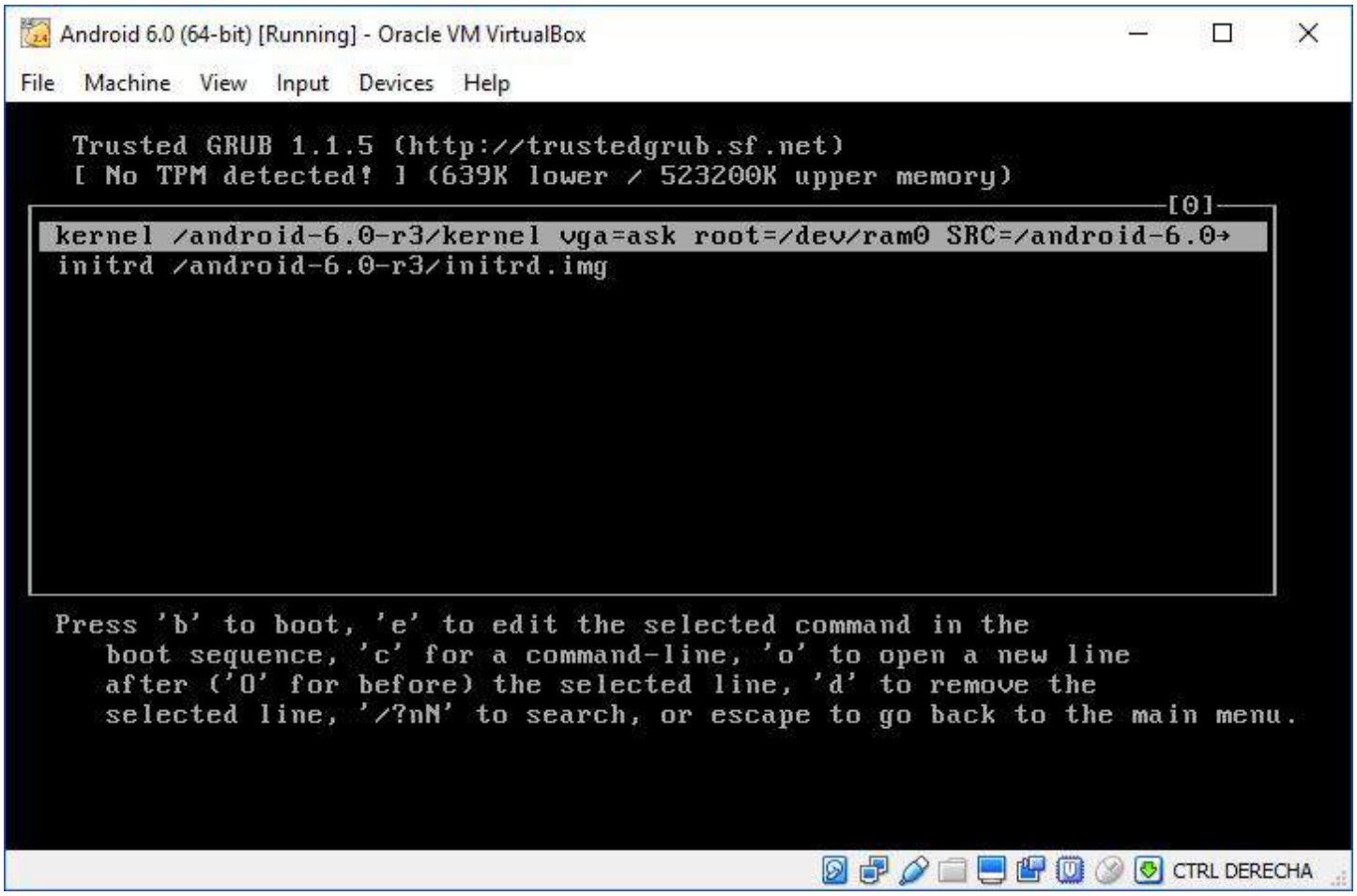
In the booting options of the Live CD choose "Installation - Install Android to hard disk"

Choose the sda1 partition and install android and we'll install grub.

Reboot the virtual machine but make sure that the image is not in the optical drive so it can restart from the virtual hard drive.



In the grub menu we need to edit kernel like in the "Android-x86 6.0-r3" option so press e.

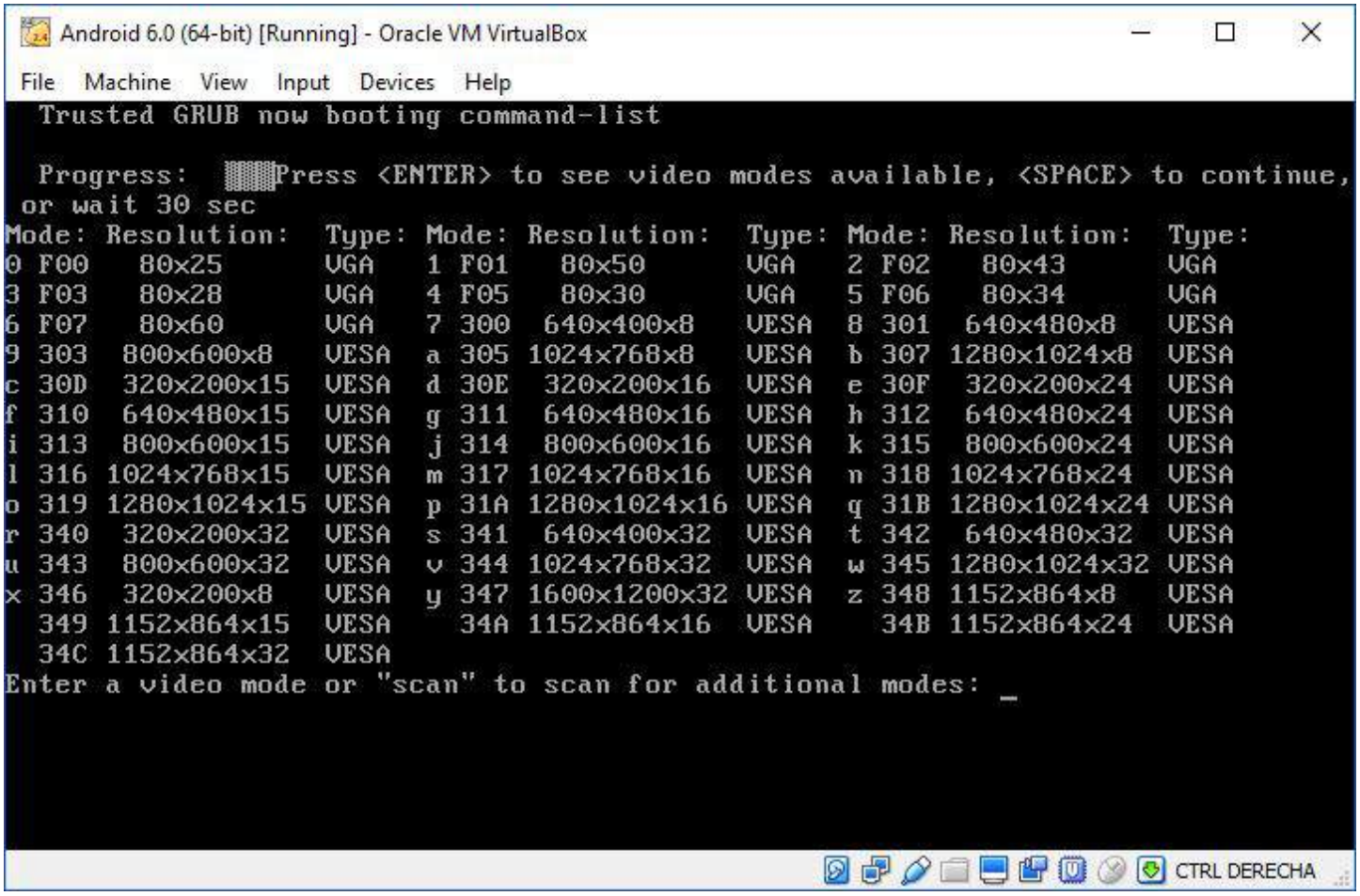


Then we substitute "quiet" with "vga=ask" and add the option "SDCARD=sda2"

In my case, the kernel line looks like this after modified:

```
kenel /android-6.0-r3/kernel vga=ask root=ram0 SRC=/android-6/android-6.0-r3 SDCARD=sda2
```

Press b to boot, then you'll be able to choose the screen size pressing ENTER (the `vga=ask` option)



Once the installation wizard has started choose the language. I could choose English (United States) and Spanish (United States) and I had trouble choosing any other.

Section 210.3: Virtual Machine setup

These are my VirtualBox settings:

- OS Type: Linux 2.6 (I've user 64bit because my computer can support it)
- Virtual hard drive size: 4Gb
- Ram Memory: 2048
- Video Memory: 8M
- Sound device: Sound Blaster 16.
- Network device: PCnet-Fast III, attached to NAT. You can also use bridged adapter, but you need a DHCP server in your environment.

The image used with this configuration has been android-x86_64-6.0-r3.iso (it is 64bit) downloaded from http://www.android-x86.org/download. I suppose that it also works with 32bit version.

Chapter 211: Leakcanary

Leak Canary is an Android and Java library used to detect leak in the application

Section 211.1: Implementing a Leak Canary in Android Application

In your *build.gradle* you need to add the below dependencies:

```
debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'
releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
```

In your `Application` class you need to add the below code inside your `onCreate()`:

```
LeakCanary.install(this);
```

That's all you need to do for *LeakCanary*, it will automatically show notifications when there is a leak in your build.

Chapter 212: Okio

Section 212.1: Download / Implement

Download the latest JAR or grab via Maven:

```
<dependency>
    <groupId>com.squareup.okio</groupId>
    <artifactId>okio</artifactId>
    <version>1.12.0</version>
</dependency>
```

or Gradle:

```
compile 'com.squareup.okio:okio:1.12.0'
```

Section 212.2: PNG decoder

Decoding the chunks of a PNG file demonstrates Okio in practice.

```
private static final ByteString PNG_HEADER = ByteString.decodeHex("89504e470d0a1a0a");

public void decodePng(InputStream in) throws IOException {
  try (BufferedSource pngSource = Okio.buffer(Okio.source(in))) {
    ByteString header = pngSource.readByteString(PNG_HEADER.size());
    if (!header.equals(PNG_HEADER)) {
      throw new IOException("Not a PNG.");
    }

    while (true) {
      Buffer chunk = new Buffer();

      // Each chunk is a length, type, data, and CRC offset.
      int length = pngSource.readInt();
      String type = pngSource.readUtf8(4);
      pngSource.readFully(chunk, length);
      int crc = pngSource.readInt();

      decodeChunk(type, chunk);
      if (type.equals("IEND")) break;
    }
  }
}

private void decodeChunk(String type, Buffer chunk) {
  if (type.equals("IHDR")) {
    int width = chunk.readInt();
    int height = chunk.readInt();
    System.out.printf("%08x: %s %d x %d%n", chunk.size(), type, width, height);
  } else {
    System.out.printf("%08x: %s%n", chunk.size(), type);
  }
}
```

Section 212.3: ByteStrings and Buffers

ByteStrings and Buffers

Okio is built around two types that pack a lot of capability into a straightforward API:

ByteString is an immutable sequence of bytes. For character data, String is fundamental. ByteString is String's long-lost brother, making it easy to treat binary data as a value. This class is ergonomic: it knows how to encode and decode itself as hex, base64, and UTF-8.

Buffer is a mutable sequence of bytes. Like ArrayList, you don't need to size your buffer in advance. You read and write buffers as a queue: write data to the end and read it from the front. There's no obligation to manage positions, limits, or capacities.

Internally, `ByteString` and `Buffer` do some clever things to save CPU and memory. If you encode a UTF-8 string as a ByteString, it caches a reference to that string so that if you decode it later, there's no work to do.

`Buffer` is implemented as a linked list of segments. When you move data from one buffer to another, it reassigns ownership of the segments rather than copying the data across. This approach is particularly helpful for multithreaded programs: a thread that talks to the network can exchange data with a worker thread without any copying or ceremony.

Chapter 213: Bluetooth Low Energy

This documentation is meant as an enhancement over the original documentation and it will focus on the latest Bluetooth LE API introduced in Android 5.0 (API 21). Both Central and Peripheral roles will be covered as well as how to start scanning and advertising operations.

Section 213.1: Finding BLE Devices

The following permissions are required to use the Bluetooth APIs:

android.permission.BLUETOOTH android.permission.BLUETOOTH_ADMIN

If you're targeting devices with Android 6.0 (**API Level 23**) or higher and want to perform scanning/advertising operations you will require a Location permission:

android.permission.ACCESS_FINE_LOCATION

or

android.permission.ACCESS_COARSE_LOCATION

Note.- Devices with Android 6.0 (API Level 23) or higher also need to have Location Services enabled.

A BluetoothAdapter object is required to start scanning/advertising operations:

```
BluetoothManager bluetoothManager = (BluetoothManager)
context.getSystemService(Context.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

The `startScan (ScanCallback callback)`method of the BluetoothLeScanner class is the most basic way to start a scanning operation. A `ScanCallback` object is required to receive results:

```
bluetoothAdapter.getBluetoothLeScanner().startScan(new ScanCallback() {
     @Override
     public void onScanResult(int callbackType, ScanResult result) {
     super.onScanResult(callbackType, result);
     Log.i(TAG, "Remote device name: " + result.getDevice().getName());
       }
    });
```

Section 213.2: Connecting to a GATT Server

Once you have discovered a desired BluetoothDevice object, you can connect to it by using its `connectGatt()` method which takes as parameters a Context object, a boolean indicating whether to automatically connect to the BLE device and a BluetoothGattCallback reference where connection events and client operations results will be delivered:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        device.connectGatt(context, false, bluetoothGattCallback, BluetoothDevice.TRANSPORT_AUTO);
    } else {
        device.connectGatt(context, false, bluetoothGattCallback);
    }
```

Override `onConnectionStateChange` in BluetoothGattCallback to receive connection an disconnection events:

```
    BluetoothGattCallback bluetoothGattCallback =
        new BluetoothGattCallback() {
    @Override
    public void onConnectionStateChange(BluetoothGatt gatt, int status,
            int newState) {
        if (newState == BluetoothProfile.STATE_CONNECTED) {
            Log.i(TAG, "Connected to GATT server.");

        } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {

            Log.i(TAG, "Disconnected from GATT server.");
        }
    }
   };
```

Section 213.3: Writing and Reading from Characteristics

Once you are connected to a Gatt Server, you're going to be interacting with it by writing and reading from the server's characteristics. To do this, first you have to discover what services are available on this server and which characteristics are available in each service:

```
 @Override
 public void onConnectionStateChange(BluetoothGatt gatt, int status,
        int newState) {
    if (newState == BluetoothProfile.STATE_CONNECTED) {
        Log.i(TAG, "Connected to GATT server.");
        gatt.discoverServices();

    }
. . .

 @Override
    public void onServicesDiscovered(BluetoothGatt gatt, int status) {
        if (status == BluetoothGatt.GATT_SUCCESS) {
             List<BluetoothGattService> services = gatt.getServices();
                for (BluetoothGattService service : services) {
                    List<BluetoothGattCharacteristic> characteristics =
service.getCharacteristics();
                    for (BluetoothGattCharacteristic characteristic : characteristics) {
                        ///Once you have a characteristic object, you can perform read/write
                        //operations with it
                    }
                }
            }
           }
```

A basic write operation goes like this:

```
characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);
```

When the write process has finished, the `onCharacteristicWrite` method of your `BluetoothGattCallback` will be called:

```
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
```

```
        Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);
    }
```

A basic write operation goes like this:

```
gatt.readCharacteristic(characteristic);
```

When the write process has finished, the `onCharacteristicRead` method of your `BluetoothGattCallback` will be called:

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic,
int status) {
  super.onCharacteristicRead(gatt, characteristic, status);
  byte[] value = characteristic.getValue();
  }
```

Section 213.4: Subscribing to Notifications from the Gatt Server

You can request to be notified from the Gatt Server when the value of a characteristic has been changed:

```
gatt.setCharacteristicNotification(characteristic, true);
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
mBluetoothGatt.writeDescriptor(descriptor);
```

All notifications from the server will be received in the `onCharacteristicChanged` method of your BluetoothGattCallback:

```
    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic) {
        super.onCharacteristicChanged(gatt, characteristic);
        byte[] newValue = characteristic.getValue();
}
```

Section 213.5: Advertising a BLE Device

You can use Bluetooth LE Advertising to broadcast data packages to all nearby devices without having to establish a connection first. Bear in mind that there's a strict limit of 31 bytes of advertisement data. Advertising your device is also the first step towards letting other users connect to you.

Since not all devices support Bluetooth LE Advertising, the first step is to check that your device has all the necessary requirements to support it. Afterwards, you can initialize a `BluetoothLeAdvertiser` object and with it, you can start advertising operations:

```
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP &&
bluetoothAdapter.isMultipleAdvertisementSupported())
        {
            BluetoothLeAdvertiser advertiser = bluetoothAdapter.getBluetoothLeAdvertiser();

            AdvertiseData.Builder dataBuilder = new AdvertiseData.Builder();
           //Define a service UUID according to your needs
           dataBuilder.addServiceUuid(SERVICE_UUID);
```

```
            dataBuilder.setIncludeDeviceName(true);


             AdvertiseSettings.Builder settingsBuilder = new AdvertiseSettings.Builder();
             settingsBuilder.setAdvertiseMode(AdvertiseSettings.ADVERTISE_MODE_LOW_POWER);
             settingsBuilder.setTimeout(0);

             //Use the connectable flag if you intend on opening a Gatt Server
             //to allow remote connections to your device.
             settingsBuilder.setConnectable(true);

            AdvertiseCallback advertiseCallback=new AdvertiseCallback() {
            @Override
            public void onStartSuccess(AdvertiseSettings settingsInEffect) {
                super.onStartSuccess(settingsInEffect);
                Log.i(TAG, "onStartSuccess: ");
            }

            @Override
            public void onStartFailure(int errorCode) {
                super.onStartFailure(errorCode);
                Log.e(TAG, "onStartFailure: "+errorCode );
            }
          };
advertising.startAdvertising(settingsBuilder.build(),dataBuilder.build(),advertiseCallback);
        }
```

Section 213.6: Using a Gatt Server

In order for your device to act as a peripheral, first you need to open a `BluetoothGattServer` and populate it with at least one `BluetoothGattService` and one `BluetoothGattCharacteristic`:

```
BluetoothGattServer server=bluetoothManager.openGattServer(context, bluetoothGattServerCallback);

BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
BluetoothGattService.SERVICE_TYPE_PRIMARY);
```

This is an example of a BluetoothGattCharacteristic with full write,read and notify permissions. According to your needs, you might want to fine tune the permissions that you grant this characteristic:

```
BluetoothGattCharacteristic characteristic = new BluetoothGattCharacteristic(CHARACTERISTIC_UUID,
                    BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE |
                            BluetoothGattCharacteristic.PROPERTY_NOTIFY,
                    BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

characteristic.addDescriptor(new
BluetoothGattDescriptor(UUID.fromString("00002902-0000-1000-8000-00805f9b34fb"),
BluetoothGattCharacteristic.PERMISSION_WRITE));

service.addCharacteristic(characteristic);

server.addService(service);
```

The `BluetoothGattServerCallback` is responsible for receiving all events related to your `BluetoothGattServer`:

```
BluetoothGattServerCallback bluetoothGattServerCallback= new BluetoothGattServerCallback() {
                @Override
```

```
                public void onConnectionStateChange(BluetoothDevice device, int status, int
newState) {
                    super.onConnectionStateChange(device, status, newState);
                }

                @Override
                public void onCharacteristicReadRequest(BluetoothDevice device, int requestId, int
offset, BluetoothGattCharacteristic characteristic) {
                    super.onCharacteristicReadRequest(device, requestId, offset, characteristic);
                }

                @Override
                public void onCharacteristicWriteRequest(BluetoothDevice device, int requestId,
BluetoothGattCharacteristic characteristic, boolean preparedWrite, boolean responseNeeded, int
offset, byte[] value) {
                    super.onCharacteristicWriteRequest(device, requestId, characteristic,
preparedWrite, responseNeeded, offset, value);
                }

                @Override
                public void onDescriptorReadRequest(BluetoothDevice device, int requestId, int
offset, BluetoothGattDescriptor descriptor) {
                    super.onDescriptorReadRequest(device, requestId, offset, descriptor);
                }

                @Override
                public void onDescriptorWriteRequest(BluetoothDevice device, int requestId,
BluetoothGattDescriptor descriptor, boolean preparedWrite, boolean responseNeeded, int offset,
byte[] value) {
                    super.onDescriptorWriteRequest(device, requestId, descriptor, preparedWrite,
responseNeeded, offset, value);
                }
```

Whenever you receive a request for a write/read to a characteristic or descriptor you must send a response to it in order for the request to be completed succesfully :

```
@Override
 public void onCharacteristicReadRequest(BluetoothDevice device, int requestId, int offset,
BluetoothGattCharacteristic characteristic) {
    super.onCharacteristicReadRequest(device, requestId, offset, characteristic);
    server.sendResponse(device, requestId, BluetoothGatt.GATT_SUCCESS, offset, YOUR_RESPONSE);
}
```

Chapter 214: Looper

A `Looper` is an Android class used to run a message loop for a thread, which usually do not have one associated with them.

The most common `Looper` in Android is the main-loop, also commonly known as the main-thread. This instance is unique for an application and can be accessed statically with `Looper.getMainLooper()`.

If a `Looper` is associated with the current thread, it can be retrieved with `Looper.myLooper()`.

Section 214.1: Create a simple LooperThread

A typical example of the implementation of a `Looper` thread given by the official documentation uses `Looper.prepare()` and `Looper.loop()` and associates a `Handler` with the loop between these calls.

```
class LooperThread extends Thread {
    public Handler mHandler;

    public void run() {
        Looper.prepare();

        mHandler = new Handler() {
            public void handleMessage(Message msg) {
                // process incoming messages here
            }
        };

        Looper.loop();
    }
}
```

Section 214.2: Run a loop with a HandlerThread

A `HandlerThread` can be used to start a thread with a `Looper`. This looper then can be used to create a `Handler` for communications with it.

```
HandlerThread thread = new HandlerThread("thread-name");
thread.start();
Handler handler = new Handler(thread.getLooper());
```

Chapter 215: Annotation Processor

Annotation processor is a tool build in javac for scanning and processing annotations at compile time.

Annotations are a class of metadata that can be associated with classes, methods, fields, and even other annotations.There are two ways to access these annotations at runtime via reflection and at compile time via annotation processors.

Section 215.1: @NonNull Annotation

```
public class Foo {
    private String name;
    public Foo(@NonNull String name){...};
    ...
}
```

Here @NonNull is annotation which is processed compile time by the android studio to warn you that the particular function needs non null parameter.

Section 215.2: Types of Annotations

There are three types of annotations.

1. **Marker Annotation** - annotation that has no method

   ```
   @interface CustomAnnotation {}
   ```

2. **Single-Value Annotation** - annotation that has one method

   ```
   @interface CustomAnnotation {
       int value();
   }
   ```

3. **Multi-Value Annotation** - annotation that has more than one method

   ```
   @interface CustomAnnotation{
       int value1();
       String value2();
       String value3();
   }
   ```

Section 215.3: Creating and Using Custom Annotations

For creating custom annotations we need to decide

- Target - on which these annotations will work on like field level, method level, type level etc.
- Retention - to what level annotation will be available.

For this, we have built in custom annotations. Check out these mostly used ones:

@Target

Element Types	Where the annotation can be applied
TYPE	class, interface or enumeration
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variables
ANNOTATION_TYPE	annotation type
PARAMETER	parameter

@Retention

RetentionPolicy	Availability
RetentionPolicy.SOURCE	refers to the source code, discarded during compilation. It will not be available in the compiled class.
RetentionPolicy.CLASS	refers to the .class file, available to java compiler but not to JVM . It is included in the class file.
RetentionPolicy.RUNTIME	refers to the runtime, available to java compiler and JVM .

Creating Custom Annotation

```
@Retention(RetentionPolicy.SOURCE) // will not be available in compiled class
@Target(ElementType.METHOD) // can be applied to methods only
@interface CustomAnnotation{
      int value();
}
```

Using Custom Annotation

```
class Foo{
  @CustomAnnotation(value = 1)  // will be used by an annotation processor
  public void foo(){..}
}
```

the value provided inside `@CustomAnnotation` will be consumed by an Annotationprocessor may be to generate code at compile time etc.

Chapter 216: SyncAdapter with periodically do sync of data

The sync adapter component in your app encapsulates the code for the tasks that transfer data between the device and a server. Based on the scheduling and triggers you provide in your app, the sync adapter framework runs the code in the sync adapter component.

Recently i worked on SyncAdapter i want share my knowledge with others,it may help others.

Section 216.1: Sync adapter with every min requesting value from server

```
<provider
       android:name=".DummyContentProvider"
       android:authorities="sample.map.com.ipsyncadapter"
       android:exported="false" />

    <!-- This service implements our SyncAdapter. It needs to be exported, so that the system
   sync framework can access it. -->
    <service android:name=".SyncService"
       android:exported="true">
        <!-- This intent filter is required. It allows the system to launch our sync service
       as needed. -->
        <intent-filter>
            <action android:name="android.content.SyncAdapter" />
        </intent-filter>
        <!-- This points to a required XML file which describes our SyncAdapter. -->
        <meta-data android:name="android.content.SyncAdapter"
           android:resource="@xml/syncadapter" />
    </service>

    <!-- This implements the account we'll use as an attachment point for our SyncAdapter. Since
   our SyncAdapter doesn't need to authenticate the current user (it just fetches a public RSS
   feed), this account's implementation is largely empty.

   It's also possible to attach a SyncAdapter to an existing account provided by another
   package. In that case, this element could be omitted here. -->
    <service android:name=".AuthenticatorService"
       >
        <!-- Required filter used by the system to launch our account service. -->
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator" />
        </intent-filter>
        <!-- This points to an XMLf ile which describes our account service. -->
        <meta-data android:name="android.accounts.AccountAuthenticator"
           android:resource="@xml/authenticator" />
    </service>
```

This code need to be add in manifest file

In above code we have the syncservice and conteprovider and authenticatorservice.

In app we need to create the xml package to add syncadpter and authenticator xml files. **authenticator.xml**

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="@string/R.String.accountType"
    android:icon="@mipmap/ic_launcher"
```

```
    android:smallIcon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    />
```

syncadapter

```
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"
    android:contentAuthority="@string/R.String.contentAuthority"
    android:accountType="@string/R.String.accountType"
    android:userVisible="true"
    android:allowParallelSyncs="true"
    android:isAlwaysSyncable="true"
    android:supportsUploading="false"/>
```

Authenticator

```
import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.NetworkErrorException;
import android.content.Context;
import android.os.Bundle;

public class Authenticator extends AbstractAccountAuthenticator {
    private Context mContext;
    public Authenticator(Context context) {
        super(context);
         this.mContext=context;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse accountAuthenticatorResponse, String
s) {
        return null;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse accountAuthenticatorResponse, String s,
String s1, String[] strings, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse accountAuthenticatorResponse,
Account account, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse accountAuthenticatorResponse, Account
account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public String getAuthTokenLabel(String s) {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse accountAuthenticatorResponse,
```

```
Account account, String s, Bundle bundle) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle hasFeatures(AccountAuthenticatorResponse accountAuthenticatorResponse, Account
account, String[] strings) throws NetworkErrorException {
        return null;
    }
}
```

AuthenticatorService

```
public class AuthenticatorService extends Service {

    private Authenticator authenticator;

    public AuthenticatorService() {
        super();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        IBinder ret = null;
        if (intent.getAction().equals(AccountManager.ACTION_AUTHENTICATOR_INTENT)) ;
        ret = getAuthenticator().getIBinder();
        return ret;
    }

    public Authenticator getAuthenticator() {
        if (authenticator == null)
            authenticator = new Authenticator(this);
        return authenticator;
    }
}
```

IpDataDBHelper

```
 public class IpDataDBHelper extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION=1;
    private static final String DATABASE_NAME="ip.db";
    public static final String TABLE_IP_DATA="ip";

    public static final String COLUMN_ID="_id";
    public static final String COLUMN_IP="ip";
    public static final String COLUMN_COUNTRY_CODE="country_code";
    public static final String COLUMN_COUNTRY_NAME="country_name";
    public static final String COLUMN_CITY="city";
    public static final String COLUMN_LATITUDE="latitude";
    public static final String COLUMN_LONGITUDE="longitude";

    public IpDataDBHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int
version) {
        super(context, DATABASE_NAME, factory, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        String CREATE_TABLE="CREATE TABLE " + TABLE_IP_DATA + "( " + COLUMN_ID + " INTEGER PRIMARY
KEY ,"
```

```
                + COLUMN_IP + " INTEGER ," + COLUMN_COUNTRY_CODE + " INTEGER ," +
COLUMN_COUNTRY_NAME +
                " TEXT ," + COLUMN_CITY + " TEXT ," + COLUMN_LATITUDE + " INTEGER ," +
COLUMN_LONGITUDE + " INTEGER)";
        sqLiteDatabase.execSQL(CREATE_TABLE);
        Log.d("SQL",CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_IP_DATA);
        onCreate(sqLiteDatabase);
    }

    public long AddIPData(ContentValues values)
    {
        SQLiteDatabase sqLiteDatabase =getWritableDatabase();
        long insertedRow=sqLiteDatabase.insert(TABLE_IP_DATA,null,values);
        return insertedRow;
    }

    public Cursor getAllIpData()
    {
        String[]
projection={COLUMN_ID,COLUMN_IP,COLUMN_COUNTRY_CODE,COLUMN_COUNTRY_NAME,COLUMN_CITY,COLUMN_LATITUDE
,COLUMN_LONGITUDE};
        SQLiteDatabase sqLiteDatabase =getReadableDatabase();
        Cursor cursor = sqLiteDatabase.query(TABLE_IP_DATA,projection,null,null,null,null,null);
        return cursor;
    }

    public int deleteAllIpData()
    {
        SQLiteDatabase sqLiteDatabase=getWritableDatabase();
        int rowDeleted=sqLiteDatabase.delete(TABLE_IP_DATA,null,null);
        return rowDeleted;
    }
}
```

MainActivity

```
     public class MainActivity extends AppCompatActivity {

        private static final String ACCOUNT_TYPE="sample.map.com.ipsyncadapter";
        private static final String AUTHORITY="sample.map.com.ipsyncadapter";
        private static final String ACCOUNT_NAME="Sync";

        public TextView mIp,mCountryCod,mCountryName,mCity,mLatitude,mLongitude;
        CursorAdapter cursorAdapter;
        Account mAccount;
        private String TAG=this.getClass().getCanonicalName();
        ListView mListView;
        public SharedPreferences mSharedPreferences;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            mListView = (ListView) findViewById(R.id.list);
            mIp=(TextView)findViewById(R.id.txt_ip);
            mCountryCod=(TextView)findViewById(R.id.txt_country_code);
            mCountryName=(TextView)findViewById(R.id.txt_country_name);
```

```
            mCity=(TextView)findViewById(R.id.txt_city);
            mLatitude=(TextView)findViewById(R.id.txt_latitude);
            mLongitude=(TextView)findViewById(R.id.txt_longitude);
            mSharedPreferences=getSharedPreferences("MyIp",0);

//Using shared preference iam displaying values in text view.
            String txtIp=mSharedPreferences.getString("ipAdr","");
            String txtCC=mSharedPreferences.getString("CCode","");
            String txtCN=mSharedPreferences.getString("CName","");
            String txtC=mSharedPreferences.getString("City","");
            String txtLP=mSharedPreferences.getString("Latitude","");
            String txtLN=mSharedPreferences.getString("Longitude","");

            mIp.setText(txtIp);
            mCountryCod.setText(txtCC);
            mCountryName.setText(txtCN);
            mCity.setText(txtC);
            mLatitude.setText(txtLP);
            mLongitude.setText(txtLN);

            mAccount=createSyncAccount(this);
//In this code i am using content provider to save data.
           /* Cursor
cursor=getContentResolver().query(MyIPContentProvider.CONTENT_URI,null,null,null,null);
            cursorAdapter=new SimpleCursorAdapter(this,R.layout.list_item,cursor,new String
[]{"ip","country_code","country_name","city","latitude","longitude"},
                                                                new int[]
{R.id.txt_ip,R.id.txt_country_code,R.id.txt_country_name,R.id.txt_city,R.id.txt_latitude,R.id.txt_lon
gitude},0);

            mListView.setAdapter(cursorAdapter);
            getContentResolver().registerContentObserver(MyIPContentProvider.CONTENT_URI,true,new
StockContentObserver(new Handler()));
    */
            Bundle settingBundle=new Bundle();
            settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL,true);
            settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED,true);
            ContentResolver.requestSync(mAccount,AUTHORITY,settingBundle);
            ContentResolver.setSyncAutomatically(mAccount,AUTHORITY,true);
            ContentResolver.addPeriodicSync(mAccount,AUTHORITY,Bundle.EMPTY,60);
        }

        private Account createSyncAccount(MainActivity mainActivity) {
            Account account=new Account(ACCOUNT_NAME,ACCOUNT_TYPE);
            AccountManager
accountManager=(AccountManager)mainActivity.getSystemService(ACCOUNT_SERVICE);
            if(accountManager.addAccountExplicitly(account,null,null))
            {

            }else
            {

            }
            return account;
        }


        private class StockContentObserver extends ContentObserver {
            @Override
            public void onChange(boolean selfChange, Uri uri) {
                Log.d(TAG, "CHANGE OBSERVED AT URI: " + uri);
```

```
cursorAdapter.swapCursor(getContentResolver().query(MyIPContentProvider.CONTENT_URI, null, null,
null, null));
            }

            public StockContentObserver(Handler handler) {
                super(handler);

            }
        }
        @Override
        protected void onResume() {
            super.onResume();
            registerReceiver(syncStaredReceiver, new IntentFilter(SyncAdapter.SYNC_STARTED));
            registerReceiver(syncFinishedReceiver, new IntentFilter(SyncAdapter.SYNC_FINISHED));
        }

        @Override
        protected void onPause() {
            super.onPause();
            unregisterReceiver(syncStaredReceiver);
            unregisterReceiver(syncFinishedReceiver);
        }
        private BroadcastReceiver syncFinishedReceiver = new BroadcastReceiver() {

            @Override
            public void onReceive(Context context, Intent intent) {
                Log.d(TAG, "Sync finished!");
                Toast.makeText(getApplicationContext(), "Sync Finished",
Toast.LENGTH_SHORT).show();
            }
        };
        private BroadcastReceiver syncStaredReceiver = new BroadcastReceiver() {

            @Override
            public void onReceive(Context context, Intent intent) {
                Log.d(TAG, "Sync started!");
                Toast.makeText(getApplicationContext(), "Sync started...",
Toast.LENGTH_SHORT).show();
            }
        };
    }
```

MyIPContentProvider

```
public class MyIPContentProvider extends ContentProvider {

public static final int IP_DATA=1;
private static final String AUTHORITY="sample.map.com.ipsyncadapter";
private static final String TABLE_IP_DATA="ip_data";
public static final Uri CONTENT_URI=Uri.parse("content://" + AUTHORITY + '/' + TABLE_IP_DATA);
private static final UriMatcher URI_MATCHER= new UriMatcher(UriMatcher.NO_MATCH);

static
{
    URI_MATCHER.addURI(AUTHORITY,TABLE_IP_DATA,IP_DATA);
}

private IpDataDBHelper myDB;

@Override
public boolean onCreate() {
```

```
    myDB=new IpDataDBHelper(getContext(),null,null,1);
    return false;
}

@Nullable
@Override
public Cursor query(Uri uri, String[] strings, String s, String[] strings1, String s1) {
    int uriType=URI_MATCHER.match(uri);
    Cursor cursor=null;
    switch (uriType)
    {
        case IP_DATA:
            cursor=myDB.getAllIpData();
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URL");
    }
    cursor.setNotificationUri(getContext().getContentResolver(), uri);
    return cursor;
}

@Nullable
@Override
public String getType(Uri uri) {
    return null;
}

@Nullable
@Override
public Uri insert(Uri uri, ContentValues contentValues) {
    int uriType=URI_MATCHER.match(uri);
    long id=0;
    switch (uriType)
    {
        case IP_DATA:
            id=myDB.AddIPData(contentValues);
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
    getContext().getContentResolver().notifyChange(uri,null);
    return Uri.parse(contentValues + "/" + id);
}

@Override
public int delete(Uri uri, String s, String[] strings) {
    int uriType=URI_MATCHER.match(uri);
    int rowsDeleted=0;

    switch (uriType)
    {
        case IP_DATA:
            rowsDeleted=myDB.deleteAllIpData();
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
    getContext().getContentResolver().notifyChange(uri,null);
    return rowsDeleted;
}

@Override
```

```
public int update(Uri uri, ContentValues contentValues, String s, String[] strings) {
    return 0;
}
```

}

SyncAdapter

```
public class SyncAdapter extends AbstractThreadedSyncAdapter {
ContentResolver mContentResolver;
Context mContext;
public static final String SYNC_STARTED="Sync Started";
public static final String SYNC_FINISHED="Sync Finished";
private static final String TAG=SyncAdapter.class.getCanonicalName();
public SharedPreferences mSharedPreferences;

public SyncAdapter(Context context, boolean autoInitialize) {
    super(context, autoInitialize);
    this.mContext=context;
    mContentResolver=context.getContentResolver();
    Log.i("SyncAdapter","SyncAdapter");
}

@Override
public void onPerformSync(Account account, Bundle bundle, String s, ContentProviderClient
contentProviderClient, SyncResult syncResult) {

    Intent intent = new Intent(SYNC_STARTED);
    mContext.sendBroadcast(intent);

    Log.i(TAG, "onPerformSync");

    intent = new Intent(SYNC_FINISHED);
    mContext.sendBroadcast(intent);
    mSharedPreferences =mContext.getSharedPreferences("MyIp",0);
    SharedPreferences.Editor editor=mSharedPreferences.edit();

    mContentResolver.delete(MyIPContentProvider.CONTENT_URI,null,null);

    String data="";

    try {
        URL url =new URL("https://freegeoip.net/json/");
        Log.d(TAG, "URL :"+url);
        HttpURLConnection connection=(HttpURLConnection)url.openConnection();
        Log.d(TAG,"Connection :"+connection);
        connection.connect();
        Log.d(TAG,"Connection 1:"+connection);
        InputStream inputStream=connection.getInputStream();
        data=getInputData(inputStream);
        Log.d(TAG,"Data :"+data);

        if (data != null || !data.equals("null")) {
            JSONObject jsonObject = new JSONObject(data);


            String ipa = jsonObject.getString("ip");
            String country_code = jsonObject.getString("country_code");
            String country_name = jsonObject.getString("country_name");
            String region_code=jsonObject.getString("region_code");
            String region_name=jsonObject.getString("region_name");
```

```
            String zip_code=jsonObject.getString("zip_code");
            String time_zone=jsonObject.getString("time_zone");
            String metro_code=jsonObject.getString("metro_code");

            String city = jsonObject.getString("city");
            String latitude = jsonObject.getString("latitude");
            String longitude = jsonObject.getString("longitude");
           /* ContentValues values = new ContentValues();
            values.put("ip", ipa);
            values.put("country_code", country_code);
            values.put("country_name", country_name);
            values.put("city", city);
            values.put("latitude", latitude);
            values.put("longitude", longitude);*/
            //Using cursor adapter for results.
            //mContentResolver.insert(MyIPContentProvider.CONTENT_URI, values);

            //Using Shared preference for results.
            editor.putString("ipAdr",ipa);
            editor.putString("CCode",country_code);
            editor.putString("CName",country_name);
            editor.putString("City",city);
            editor.putString("Latitude",latitude);
            editor.putString("Longitude",longitude);
            editor.commit();

        }
        }catch(Exception e){
            e.printStackTrace();
        }

}

private String getInputData(InputStream inputStream) throws IOException {
    StringBuilder builder=new StringBuilder();
    BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(inputStream));
    //String data=null;
    /*Log.d(TAG,"Builder 2:"+ bufferedReader.readLine());
    while ((data=bufferedReader.readLine())!= null);
    {
        builder.append(data);
        Log.d(TAG,"Builder :"+data);
    }
    Log.d(TAG,"Builder 1 :"+data);
    bufferedReader.close();*/
    String data=bufferedReader.readLine();
    bufferedReader.close();
    return data.toString();
}
```

}

SyncService

```
public class SyncService extends Service {
private static SyncAdapter syncAdapter=null;
private static final Object syncAdapterLock=new Object();

@Override
public void onCreate() {
    synchronized (syncAdapterLock)
```

```
    {
        if(syncAdapter==null)
        {
            syncAdapter =new SyncAdapter(getApplicationContext(),true);
        }
    }
}

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return syncAdapter.getSyncAdapterBinder();
}
```

}

Chapter 217: Fastjson

Fastjson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object.

Fastjson Features:

Provide best performance in server side and android client

Provide simple `toJSONString()` and `parseObject()` methods to convert Java objects to JSON and vice-versa

Allow pre-existing unmodifiable objects to be converted to and from JSON

Extensive support of Java Generics

Section 217.1: Parsing JSON with Fastjson

You can look at example in Fastjson library

Encode

```
import com.alibaba.fastjson.JSON;

Group group = new Group();
group.setId(0L);
group.setName("admin");

User guestUser = new User();
guestUser.setId(2L);
guestUser.setName("guest");

User rootUser = new User();
rootUser.setId(3L);
rootUser.setName("root");

group.addUser(guestUser);
group.addUser(rootUser);

String jsonString = JSON.toJSONString(group);

System.out.println(jsonString);
```

Output

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"},{"id":3,"name":"root"}]}
```

Decode

```
String jsonString = ...;
Group group = JSON.parseObject(jsonString, Group.class);
```

Group.java

```
public class Group {

    private Long id;
```

```
    private String name;
    private List<User> users = new ArrayList<User>();

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
        this.users = users;
    }

    public void addUser(User user) {
        users.add(user);
    }
}
```

User.java

```
public class User {

    private Long   id;
    private String name;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Section 217.2: Convert the data of type Map to JSON String

Code

```
    Group group = new Group();
    group.setId(1);
    group.setName("Ke");

    User user1 = new User();
    user1.setId(2);
    user1.setName("Liu");

    User user2 = new User();
    user2.setId(3);
    user2.setName("Yue");
    group.getList().add(user1);
    group.getList().add(user2);

    Map<Integer, Object> map = new HashMap<Integer,Object>();
    map.put(1, "No.1");
    map.put(2, "No.2");
    map.put(3, group.getList());

    String jsonString = JSON.toJSONString(map);
    System.out.println(jsonString);
```

Output

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"},{"id":3,"name":"Yue"}]}
```

Chapter 218: JSON in Android with org.json

Section 218.1: Creating a simple JSON object

Create the `JSONObject` using the empty constructor and add fields using the `put()` method, which is overloaded so that it can be used with different types:

```
try {
    // Create a new instance of a JSONObject
    final JSONObject object = new JSONObject();

    // With put you can add a name/value pair to the JSONObject
    object.put("name", "test");
    object.put("content", "Hello World!!!1");
    object.put("year", 2016);
    object.put("value", 3.23);
    object.put("member", true);
    object.put("null_value", JSONObject.NULL);

    // Calling toString() on the JSONObject returns the JSON in string format.
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "Failed to create JSONObject", e);
}
```

The resulting `JSON` string looks like this:

```
{
   "name":"test",
   "content":"Hello World!!!1",
   "year":2016,
   "value":3.23,
   "member":true,
   "null_value":null
}
```

Section 218.2: Create a JSON String with null value

If you need to produce a JSON string with a value of **`null`** like this:

```
{
   "name":null
}
```

Then you have to use the special constant JSONObject.NULL.

Functioning example:

```
jsonObject.put("name", JSONObject.NULL);
```

Section 218.3: Add JSONArray to JSONObject

```
// Create a new instance of a JSONArray
JSONArray array = new JSONArray();
```

```
// With put() you can add a value to the array.
array.put("ASDF");
array.put("QWERTY");

// Create a new instance of a JSONObject
JSONObject obj = new JSONObject();

try {
    // Add the JSONArray to the JSONObject
    obj.put("the_array", array);
} catch (JSONException e) {
    e.printStackTrace();
}

String json = obj.toString();
```

The resulting JSON string looks like this:

```
{
   "the_array":[
      "ASDF",
      "QWERTY"
   ]
}
```

Section 218.4: Parse simple JSON object

Consider the following JSON string:

```
{
  "title": "test",
  "content": "Hello World!!!",
  "year": 2016,
  "names" : [
        "Hannah",
        "David",
        "Steve"
   ]
}
```

This JSON object can be parsed using the following code:

```
try {
    // create a new instance from a string
    JSONObject jsonObject = new JSONObject(jsonAsString);
    String title = jsonObject.getString("title");
    String content = jsonObject.getString("content");
    int year = jsonObject.getInt("year");
    JSONArray names = jsonObject.getJSONArray("names"); //for an array of String objects
} catch (JSONException e) {
    Log.w(TAG,"Could not parse JSON. Error: " + e.getMessage());
}
```

Here is another example with a JSONArray nested inside JSONObject:

```
{
    "books":[
      {
        "title":"Android JSON Parsing",
```

```
        "times_sold":186
      }
    ]
}
```

This can be parsed with the following code:

```
JSONObject root = new JSONObject(booksJson);
JSONArray booksArray = root.getJSONArray("books");
JSONObject firstBook = booksArray.getJSONObject(0);
String title = firstBook.getString("title");
int timesSold = firstBook.getInt("times_sold");
```

Section 218.5: Check for the existence of fields on JSON

Sometimes it's useful to check if a field is present or absent on your JSON to avoid some `JSONException` on your code.

To achieve that, use the `JSONObject#has(String)` or the method, like on the following example:

Sample JSON

```
{
   "name":"James"
}
```

Java code

```
String jsonStr = " { \"name\":\"James\" }";
JSONObject json = new JSONObject(jsonStr);
// Check if the field "name" is present
String name, surname;

// This will be true, since the field "name" is present on our JSON.
if (json.has("name")) {
    name = json.getString("name");
}
else {
    name = "John";
}
// This will be false, since our JSON doesn't have the field "surname".
if (json.has("surname")) {
    surname = json.getString("surname");
}
else {
    surname = "Doe";
}

// Here name == "James" and surname == "Doe".
```

Section 218.6: Create nested JSON object

To produce nested JSON object, you need to simply add one JSON object to another:

```
JSONObject mainObject = new JSONObject();            // Host object
JSONObject requestObject = new JSONObject();         // Included object

try {
```

```
    requestObject.put("lastname", lastname);
    requestObject.put("phone", phone);
    requestObject.put("latitude", lat);
    requestObject.put("longitude", lon);
    requestObject.put("theme", theme);
    requestObject.put("text", message);

    mainObject.put("claim", requestObject);
} catch (JSONException e) {
    return "JSON Error";
}
```

Now `mainObject` contains a key called `claim` with the whole `requestObject` as a value.

Section 218.7: Updating the elements in the JSON

sample json to update

```
{
"student":{"name":"Rahul", "lastname":"sharma"},
"marks":{"maths":"88"}
}
```

To update the elements value in the json we need to assign the value and update.

```
try {
    // Create a new instance of a JSONObject
    final JSONObject object = new JSONObject(jsonString);

    JSONObject studentJSON = object.getJSONObject("student");
    studentJSON.put("name","Kumar");

     object.remove("student");

     object.put("student",studentJSON);

    // Calling toString() on the JSONObject returns the JSON in string format.
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "Failed to create JSONObject", e);
}
```

updated value

```
{
"student":{"name":"Kumar", "lastname":"sharma"},
"marks":{"maths":"88"}
}
```

Section 218.8: Using JsonReader to read JSON from a stream

`JsonReader` reads a JSON encoded value as a stream of tokens.

```
  public List<Message> readJsonStream(InputStream in) throws IOException {
    JsonReader reader = new JsonReader(new InputStreamReader(in, "UTF-8"));
    try {
      return readMessagesArray(reader);
```

```
    } finally {
      reader.close();
    }
  }

  public List<Message> readMessagesArray(JsonReader reader) throws IOException {
    List<Message> messages = new ArrayList<Message>();

    reader.beginArray();
    while (reader.hasNext()) {
      messages.add(readMessage(reader));
    }
    reader.endArray();
    return messages;
  }

  public Message readMessage(JsonReader reader) throws IOException {
    long id = -1;
    String text = null;
    User user = null;
    List<Double> geo = null;

    reader.beginObject();
    while (reader.hasNext()) {
      String name = reader.nextName();
      if (name.equals("id")) {
        id = reader.nextLong();
      } else if (name.equals("text")) {
        text = reader.nextString();
      } else if (name.equals("geo") && reader.peek() != JsonToken.NULL) {
        geo = readDoublesArray(reader);
      } else if (name.equals("user")) {
        user = readUser(reader);
      } else {
        reader.skipValue();
      }
    }
    reader.endObject();
    return new Message(id, text, user, geo);
  }

  public List<Double> readDoublesArray(JsonReader reader) throws IOException {
    List<Double> doubles = new ArrayList<Double>();

    reader.beginArray();
    while (reader.hasNext()) {
      doubles.add(reader.nextDouble());
    }
    reader.endArray();
    return doubles;
  }

  public User readUser(JsonReader reader) throws IOException {
    String username = null;
    int followersCount = -1;

    reader.beginObject();
    while (reader.hasNext()) {
      String name = reader.nextName();
      if (name.equals("name")) {
        username = reader.nextString();
      } else if (name.equals("followers_count")) {
```

```
          followersCount = reader.nextInt();
        } else {
          reader.skipValue();
        }
      }
      reader.endObject();
      return new User(username, followersCount);
    }
```

Section 218.9: Working with null-string when parsing json

```
{
    "some_string": null,
    "ather_string": "something"
}
```

If we will use this way:

```
JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");
```

We will have output:

```
someString = "null";
```

So we need to provide this workaround:

```
/**
 * According to
http://stackoverflow.com/questions/18226288/json-jsonobject-optstring-returns-string-null
 * we need to provide a workaround to opt string from json that can be null.
 * <strong></strong>
 */
public static String optNullableString(JSONObject jsonObject, String key) {
    return optNullableString(jsonObject, key, "");
}

/**
 * According to
http://stackoverflow.com/questions/18226288/json-jsonobject-optstring-returns-string-null
 * we need to provide a workaround to opt string from json that can be null.
 * <strong></strong>
 */
public static String optNullableString(JSONObject jsonObject, String key, String fallback) {
    if (jsonObject.isNull(key)) {
        return fallback;
    } else {
        return jsonObject.optString(key, fallback);
    }
}
```

And then call:

```
JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");
```

And we will have Output as we expected:

```
someString = null; //not "null"
someString2 = "";
```

Section 218.10: Handling dynamic key for JSON response

This is an example for how to handle dynamic key for response. Here `A` and `B` are dynamic keys it can be anything

Response

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
        },
        {
          "name": "Ping"
        }
      ],
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Java code

```
// ResponseData is raw string of response
JSONObject responseDataObj = new JSONObject(responseData);
JSONArray responseArray = responseDataObj.getJSONArray("response");
for (int i = 0; i < responseArray.length(); i++) {
    // Nodes ArrayList<ArrayList<String>> declared globally
    nodes = new ArrayList<ArrayList<String>>();
    JSONObject obj = responseArray.getJSONObject(i);
    Iterator keys = obj.keys();
    while(keys.hasNext()) {
       // Loop to get the dynamic key
       String currentDynamicKey = (String)keys.next();
       // Get the value of the dynamic key
       JSONArray currentDynamicValue = obj.getJSONArray(currentDynamicKey);
       int jsonArraySize = currentDynamicValue.length();
       if(jsonArraySize > 0) {
           for (int ii = 0; ii < jsonArraySize; ii++) {
                // NameList ArrayList<String> declared globally
                nameList = new ArrayList<String>();
               if(ii == 0) {
                JSONObject nameObj = currentDynamicValue.getJSONObject(ii);
                String name = nameObj.getString("name");
                System.out.print("Name = " + name);
                // Store name in an array list
                nameList.add(name);
              }
           }
```

```
        }
      nodes.add(nameList);
    }
}
```

Chapter 219: Gson

Gson is a Java library that can be used to convert Java Objects into their JSON representation. Gson considers both of these as very important design goals.

Gson Features:

Provide simple `toJson()` and `fromJson()` methods to convert Java objects to JSON and vice-versa

Allow pre-existing unmodifiable objects to be converted to and from JSON

Extensive support of Java Generics

Support arbitrarily complex objects (with deep inheritance hierarchies and extensive use of generic types)

Section 219.1: Parsing JSON with Gson

The example shows parsing a JSON object using the Gson library from Google.

Parsing objects:

```
class Robot {
    //OPTIONAL - this annotation allows for the key to be different from the field name, and can be
omitted if key and field name are same . Also this is good coding practice as it decouple your
variable names with server keys name
    @SerializedName("version")
    private String version;

    @SerializedName("age")
    private int age;

    @SerializedName("robotName")
    private String name;

    // optional : Benefit it allows to set default values and retain them, even if key is missing
from Json response. Not required for primitive data types.

    public Robot{
       version = "";
       name = "";
    }


}
```

Then where parsing needs to occur, use the following:

```
String robotJson = "{
                        \"version\": \"JellyBean\",
                        \"age\": 3,
                        \"robotName\": \"Droid\"
                    }";

Gson gson = new Gson();
Robot robot = gson.fromJson(robotJson, Robot.class);
```

Parsing a list:

When retrieving a list of JSON objects, often you will want to parse them and convert them into Java objects.

The JSON string that we will try to convert is the following:

```
{
  "owned_dogs": [
    {
      "name": "Ron",
      "age": 12,
      "breed": "terrier"
    },
    {
      "name": "Bob",
      "age": 4,
      "breed": "bulldog"
    },
    {
      "name": "Johny",
      "age": 3,
      "breed": "golden retriever"
    }
  ]
}
```

This particular JSON array contains three objects. In our Java code we'll want to map these objects to `Dog` objects. A Dog object would look like this:

```
private class Dog {
    public String name;
    public int age;

    @SerializedName("breed")
    public String breedName;
}
```

To convert the JSON array to a `Dog[]`:

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

Converting a `Dog[]` to a JSON string:

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

To convert the JSON array to an `ArrayList<Dog>` we can do the following:

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

The Type object `typeListOfDogs` defines what a list of `Dog` objects would look like. GSON can use this type object to map the JSON array to the right values.

Alternatively, converting a `List<Dog>` to a JSON array can be done in a similar manner.

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

Section 219.2: Adding a custom Converter to Gson

Sometimes you need to serialize or deserialize some fields in a desired format, for example your backend may use the format "YYYY-MM-dd HH:mm" for dates and you want your POJOS to use the DateTime class in Joda Time.

In order to automatically convert these strings into DateTimes object, you can use a custom converter.

```
/**
 * Gson serialiser/deserialiser for converting Joda {@link DateTime} objects.
 */
public class DateTimeConverter implements JsonSerializer<DateTime>, JsonDeserializer<DateTime> {

    private final DateTimeFormatter dateTimeFormatter;

    @Inject
    public DateTimeConverter() {
        this.dateTimeFormatter = DateTimeFormat.forPattern("YYYY-MM-dd HH:mm");
    }

    @Override
    public JsonElement serialize(DateTime src, Type typeOfSrc, JsonSerializationContext context) {
        return new JsonPrimitive(dateTimeFormatter.print(src));
    }

    @Override
    public DateTime deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context)
            throws JsonParseException {

        if (json.getAsString() == null || json.getAsString().isEmpty()) {
            return null;
        }

        return dateTimeFormatter.parseDateTime(json.getAsString());
    }
}
```

To make Gson use the newly created converter you need to assign it when creating the Gson object:

```
    DateTimeConverter dateTimeConverter = new DateTimeConverter();
    Gson gson = new GsonBuilder().registerTypeAdapter(DateTime.class, dateTimeConverter)
            .create();

    String s = gson.toJson(DateTime.now());
    // this will show the date in the desired format
```

In order to deserialize the date in that format you only have to define a field in the DateTime format:

```
public class SomePojo {
    private DateTime someDate;
}
```

When Gson encounters a field of type DateTime, it will call your converter in order to deserialize the field.

Section 219.3: Parsing a List<String> with Gson

Method 1

```
Gson gson = new Gson();
```

```
String json = "[ \"Adam\", \"John\", \"Mary\" ]";

Type type = new TypeToken<List<String>>(){}.getType();
List<String> members = gson.fromJson(json, type);
Log.v("Members", members.toString());
```

This is useful for most generic container classes, since you can't get the class of a parameterized type (ie: you can't call `List<String>.class`).

Method 2

```
public class StringList extends ArrayList<String> { }

...

List<String> members = gson.fromJson(json, StringList.class);
```

Alternatively, you can always subclass the type you want, and then pass in that class. However this isn't always best practice, since it will return to you an object of type `StringList`;

Section 219.4: Adding Gson to your project

```
dependencies {
    compile 'com.google.code.gson:gson:2.8.1'
}
```

To use latest version of Gson

The below line will compile latest version of gson library every time you compile, you do not have to change version.

Pros: You can use latest features, speed and less bugs.
Cons: It might break compatibility with your code.

```
compile 'com.google.code.gson:gson:+'
```

Section 219.5: Parsing JSON to Generic Class Object with Gson

Suppose we have a JSON string :

```
["first","second","third"]
```

We can parse this JSON string into a `String` array :

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
String[] strings = gson.fromJson(jsonArray, String[].class);
```

But if we want parse it into a `List<String>` object, we must use `TypeToken`.

Here is the sample :

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>() {}.getType());
```

Suppose we have two classes below:

```
public class Outer<T> {
    public int index;
    public T data;
}

public class Person {
    public String firstName;
    public String lastName;
}
```

and we have a JSON string that should be parsed to a `Outer<Person>` object.

This example shows how to parse this JSON string to the related generic class object:

```
String json = "......";
Type userType = new TypeToken<Outer<Person>>(){}.getType();
Result<User> userResult = gson.fromJson(json,userType);
```

If the JSON string should be parsed to a `Outer<List<Person>>` object :

```
Type userListType = new TypeToken<Outer<List<Person>>>(){}.getType();
Result<List<User>> userListResult = gson.fromJson(json,userListType);
```

Section 219.6: Using Gson with inheritance

Gson does not support inheritance out of the box.

Let's say we have the following class hierarchy:

```
public class BaseClass {
    int a;

    public int getInt() {
        return a;
   }
}

public class DerivedClass1 extends BaseClass {
     int b;

     @Override
     public int getInt() {
         return b;
     }
 }

public class DerivedClass2 extends BaseClass {
    int c;

    @Override
    public int getInt() {
        return c;
    }
}
```

And now we want to serialize an instance of `DerivedClass1` to a JSON string

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);
```

Now, in another place, we receive this json string and want to deserialize it - but in compile time we only know it is supposed to be an instance of `BaseClass`:

```
BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

But GSON does not know `derivedClass1Json` was originally an instance of `DerivedClass1`, so this will print out 10.

How to solve this?

You need to build your own `JsonDeserializer`, that handles such cases. The solution is not perfectly clean, but I could not come up with a better one.

First, add the following field to your base class

```
@SerializedName("type")
private String typeName;
```

And initialize it in the base class constructor

```
public BaseClass() {
    typeName = getClass().getName();
}
```

Now add the following class:

```
public class JsonDeserializerWithInheritance<T> implements JsonDeserializer<T> {

 @Override
 public T deserialize(
     JsonElement json, Type typeOfT, JsonDeserializationContext context)
     throws JsonParseException {
     JsonObject jsonObject = json.getAsJsonObject();
     JsonPrimitive classNamePrimitive = (JsonPrimitive) jsonObject.get("type");

     String className = classNamePrimitive.getAsString();

     Class<?> clazz;
     try {
     clazz = Class.forName(className);
     } catch (ClassNotFoundException e) {
     throw new JsonParseException(e.getMessage());
     }
     return context.deserialize(jsonObject, clazz);
 }
}
```

All there is left to do is hook everything up -

```
GsonBuilder builder = new GsonBuilder();
 builder
```

```
.registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
Gson gson = builder.create();
```

And now, running the following code-

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;
String derivedClass1Json = gson.toJson(derivedClass1);

BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

Will print out 5.

Section 219.7: Parsing JSON property to enum with Gson

If you want to parse a String to enum with Gson:

> {"status" : "open"}

```
public enum Status {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

Section 219.8: Using Gson to load a JSON file from disk

This will load a JSON file from disk and convert it to the given type.

```
public static <T> T getFile(String fileName, Class<T> type) throws FileNotFoundException {
    Gson gson = new GsonBuilder()
            .create();
    FileReader json = new FileReader(fileName);
    return gson.fromJson(json, type);
}
```

Section 219.9: Using Gson as serializer with Retrofit

First of all you need to add the `GsonConverterFactory` to your build.gradle file

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Then, you have to add the converter factory when creating the Retrofit Service:

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
        .baseUrl(someUrl)
        .addConverterFactory(GsonConverterFactory.create(gson))
```

```
        .build()
        .create(RetrofitService.class);
```

You can add custom converters when creating the Gson object that you are passing to the factory. Allowing you to create custom type conversions.

Section 219.10: Parsing json array to generic class using Gson

Suppose we have a json :

```
{
  "total_count": 132,
  "page_size": 2,
  "page_index": 1,
  "twitter_posts": [
    {
      "created_on": 1465935152,
      "tweet_id": 210462857140252672,
      "tweet": "Along with our new #Twitterbird, we've also updated our Display Guidelines",
      "url": "https://twitter.com/twitterapi/status/210462857140252672"
    },
    {
      "created_on": 1465995741,
      "tweet_id": 735128881808691200,
      "tweet": "Information on the upcoming changes to Tweets is now on the developer site",
      "url": "https://twitter.com/twitterapi/status/735128881808691200"
    }
  ]
}
```

We can parse this array into a Custom Tweets (tweets list container) object manually, but it is easier to do it with `fromJson` method:

```
Gson gson = new Gson();
String jsonArray = "....";
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

Suppose we have two classes below:

```
class Tweets {
    @SerializedName("total_count")
    int totalCount;
    @SerializedName("page_size")
    int pageSize;
    @SerializedName("page_index")
    int pageIndex;
    // all you need to do it is just define List variable with correct name
    @SerializedName("twitter_posts")
    List<Tweet> tweets;
}

class Tweet {
    @SerializedName("created_on")
    long createdOn;
    @SerializedName("tweet_id")
    String tweetId;
    @SerializedName("tweet")
    String tweetBody;
    @SerializedName("url")
```

```
    String url;
}
```

and if you need just parse a json array you can use this code in your parsing:

```
String tweetsJsonArray = "[{.....},{.....}]"
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>() {}.getType());
```

Section 219.11: Custom JSON Deserializer using Gson

Imagine you have all dates in all responses in some custom format, for instance `/Date(1465935152)/` and you want apply general rule to deserialize all Json dates to java `Date` instances. In this case you need to implement custom Json Deserializer.

Example of json:

```
{
  "id": 1,
  "created_on": "Date(1465935152)",
  "updated_on": "Date(1465968945)",
  "name": "Oleksandr"
}
```

Suppose we have this class below:

```
class User {
    @SerializedName("id")
    long id;
    @SerializedName("created_on")
    Date createdOn;
    @SerializedName("updated_on")
    Date updatedOn;
    @SerializedName("name")
    String name;
}
```

Custom deserializer:

```
class DateDeSerializer implements JsonDeserializer<Date> {
    private static final String DATE_PREFIX = "/Date(";
    private static final String DATE_SUFFIX = ")/";

    @Override
    public Date deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context)
throws JsonParseException {
        String dateString = json.getAsString();
        if (dateString.startsWith(DATE_PREFIX) && dateString.endsWith(DATE_SUFFIX)) {
            dateString = dateString.substring(DATE_PREFIX.length(), dateString.length() -
DATE_SUFFIX.length());
        } else {
            throw new JsonParseException("Wrong date format: " + dateString);
        }
        return new Date(Long.parseLong(dateString) - TimeZone.getDefault().getRawOffset());
    }
}
```

And the usage:

```
Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, new DateDeSerializer())
                .create();
String json = "....";
User user = gson.fromJson(json, User.class);
```

Serialize and deserialize Jackson JSON strings with Date types

This also applies to the case where you want to make Gson Date conversion compatible with Jackson, for example.

Jackson usually serializes Date to "milliseconds since epoch" whereas Gson uses a readable format like `Aug 31, 2016 10:26:17` to represent Date. This leads to JsonSyntaxExceptions in Gson when you try to deserialize a Jackson format Date.

To circumvent this, you can add a custom serializer and a custom deserializer:

```
JsonSerializer<Date> ser = new JsonSerializer<Date>() {
    @Override
    public JsonElement serialize(Date src, Type typeOfSrc, JsonSerializationContext
                context) {
        return src == null ? null : new JsonPrimitive(src.getTime());
    }
};

JsonDeserializer<Date> deser = new JsonDeserializer<Date>() {
    @Override
    public Date deserialize(JsonElement json, Type typeOfT,
                JsonDeserializationContext context) throws JsonParseException {
        return json == null ? null : new Date(json.getAsLong());
    }
};

Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, ser)
                .registerTypeAdapter(Date.class, deser)
                .create();
```

Section 219.12: JSON Serialization/Deserialization with AutoValue and Gson

Import in your gradle root file

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

Import in your gradle app file

```
apt 'com.google.auto.value:auto-value:1.2'
apt 'com.ryanharter.auto.value:auto-value-gson:0.3.1'
provided 'com.jakewharton.auto.value:auto-value-annotations:1.2-update1'
provided 'org.glassfish:javax.annotation:10.0-b28'
```

Create object with autovalue:

```
@AutoValue public abstract class SignIn {
    @SerializedName("signin_token") public abstract String signinToken();
    public abstract String username();
```

```
    public static TypeAdapter<SignIn> typeAdapter(Gson gson) {
        return new AutoValue_SignIn.GsonTypeAdapter(gson);
    }

    public static SignIn create(String signin, String username) {
        return new AutoValue_SignIn(signin, username);
    }
}
```

Create your Gson converter with your GsonBuilder

```
Gson gson = new GsonBuilder()
                .registerTypeAdapterFactory(
                    new AutoValueGsonTypeAdapterFactory())
                .create());
```

Deserialize

```
String myJsonData = "{
    \"signin_token\": \"mySigninToken\",
    \"username\": \"myUsername\" }";
SignIn signInData = gson.fromJson(myJsonData, Signin.class);
```

Serialize

```
Signin myData = SignIn.create("myTokenData", "myUsername");
String myJsonData = gson.toJson(myData);
```

Using Gson is a great way to simplify Serialization and Deserialization code by using POJO objects. The side effect is that reflection is costly performance wise. That's why using AutoValue-Gson to generate CustomTypeAdapter will avoid this reflection cost while staying very simple to update when an api change is happening.

Chapter 220: Android Architecture Components

Android Architecture Components is new collection of libraries that help you design robust, testable, and maintainable apps. Main parts are: Lifecycles, ViewModel, LiveData, Room.

Section 220.1: Using Lifecycle in AppCompatActivity

Extend your activity from this activity

```
public abstract class BaseCompatLifecycleActivity extends AppCompatActivity implements
LifecycleRegistryOwner {
    // We need this class, because LifecycleActivity extends FragmentActivity not AppCompatActivity

    @NonNull
    private final LifecycleRegistry lifecycleRegistry = new LifecycleRegistry(this);

    @NonNull
    @Override
    public LifecycleRegistry getLifecycle() {
        return lifecycleRegistry;
    }
}
```

Section 220.2: Add Architecture Components

Project build.gradle

```
allprojects {
    repositories {
        jcenter()
        // Add this if you use Gradle 4.0+
        google()
        // Add this if you use Gradle < 4.0
        maven { url 'https://maven.google.com' }
    }
}

ext {
    archVersion = '1.0.0-alpha5'
}
```

Application build gradle

```
// For Lifecycles, LiveData, and ViewModel
compile "android.arch.lifecycle:runtime:$archVersion"
compile "android.arch.lifecycle:extensions:$archVersion"
annotationProcessor "android.arch.lifecycle:compiler:$archVersion"

// For Room
compile "android.arch.persistence.room:runtime:$archVersion"
annotationProcessor "android.arch.persistence.room:compiler:$archVersion"

// For testing Room migrations
testCompile "android.arch.persistence.room:testing:$archVersion"

// For Room RxJava support
```

```
compile "android.arch.persistence.room:rxjava2:$archVersion"
```

Section 220.3: ViewModel with LiveData transformations

```
public class BaseViewModel extends ViewModel {
    private static final int TAG_SEGMENT_INDEX = 2;
    private static final int VIDEOS_LIMIT = 100;

    // We save input params here
    private final MutableLiveData<Pair<String, String>> urlWithReferrerLiveData = new
MutableLiveData<>();

    // transform specific uri param to "tag"
    private final LiveData<String> currentTagLiveData =
Transformations.map(urlWithReferrerLiveData, pair -> {
        Uri uri = Uri.parse(pair.first);
        List<String> segments = uri.getPathSegments();
        if (segments.size() > TAG_SEGMENT_INDEX)
            return segments.get(TAG_SEGMENT_INDEX);
        return null;
    });

    // transform "tag" to videos list
    private final LiveData<List<VideoItem>> videoByTagData =
Transformations.switchMap(currentTagLiveData, tag -> contentRepository.getVideoByTag(tag,
VIDEOS_LIMIT));

    ContentRepository contentRepository;

    public BaseViewModel() {
        // some inits
    }

    public void setUrlWithReferrer(String url, String referrer) {
        // set value activates observers and transformations
        urlWithReferrerLiveData.setValue(new Pair<>(url, referrer));
    }

    public LiveData<List<VideoItem>> getVideoByTagData() {
        return videoByTagData;
    }
}
```

Somewhere in UI:

```
public class VideoActivity extends BaseCompatLifecycleActivity {
    private VideoViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Get ViewModel
        viewModel = ViewModelProviders.of(this).get(BaseViewModel.class);
        // Add observer
        viewModel.getVideoByTagData().observe(this, data -> {
            // some checks
            adapter.updateData(data);
        });
```

```
        ...
        if (savedInstanceState == null) {
            // init loading only at first creation
            // you just set params and
            viewModel.setUrlWithReferrer(url, referrer);
        }
}
```

Section 220.4: Room peristence

Room require four parts: Database class, DAO classes, Entity classes and Migration classes (now you may use **only DDL methods**):

Entity classes

```
// Set custom table name, add indexes
@Entity(tableName = "videos",
        indices = {@Index("title")}
)
public final class VideoItem {
    @PrimaryKey // required
    public long articleId;
    public String title;
    public String url;
}

// Use ForeignKey for setup table relation
@Entity(tableName = "tags",
        indices = {@Index("score"), @Index("videoId"), @Index("value")},
        foreignKeys = @ForeignKey(entity = VideoItem.class,
                parentColumns = "articleId",
                childColumns = "videoId",
                onDelete = ForeignKey.CASCADE)
)
public final class VideoTag {
    @PrimaryKey
    public long id;
    public long videoId;
    public String displayName;
    public String value;
    public double score;
}
```

DAO classes

```
@Dao
public interface VideoDao {
    // Create insert with custom conflict strategy
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void saveVideos(List<VideoItem> videos);

    // Simple update
    @Update
    void updateVideos(VideoItem... videos);

    @Query("DELETE FROM tags WHERE videoId = :videoId")
    void deleteTagsByVideoId(long videoId);

    // Custom query, you may use select/delete here
    @Query("SELECT v.* FROM tags t LEFT JOIN videos v ON v.articleId = t.videoId WHERE t.value =
```

```
:tag ORDER BY updatedAt DESC LIMIT :limit")
    LiveData<List<VideoItem>> getVideosByTag(String tag, int limit);
}
```

Database class

```
// register your entities and DAOs
@Database(entities = {VideoItem.class, VideoTag.class}, version = 2)
public abstract class ContentDatabase extends RoomDatabase {
    public abstract VideoDao videoDao();
}
```

Migrations

```
public final class Migrations {
    private static final Migration MIGRATION_1_2 = new Migration(1, 2) {
        @Override
        public void migrate(SupportSQLiteDatabase database) {
            final String[] sqlQueries = {
                    "CREATE TABLE IF NOT EXISTS `tags` (`id` INTEGER PRIMARY KEY AUTOINCREMENT," +
                            " `videoId` INTEGER, `displayName` TEXT, `value` TEXT, `score` REAL," +
                            " FOREIGN KEY(`videoId`) REFERENCES `videos`(`articleId`)" +
                            " ON UPDATE NO ACTION ON DELETE CASCADE )",
                    "CREATE  INDEX `index_tags_score` ON `tags` (`score`)",
                    "CREATE  INDEX `index_tags_videoId` ON `tags` (`videoId`)"};
            for (String query : sqlQueries) {
                database.execSQL(query);
            }
        }
    };

    public static final Migration[] ALL = {MIGRATION_1_2};

    private Migrations() {
    }
}
```

Use in Application class or provide via Dagger

```
ContentDatabase provideContentDatabase() {
    return Room.databaseBuilder(context, ContentDatabase.class, "data.db")
            .addMigrations(Migrations.ALL).build();
}
```

Write your repository:

```
public final class ContentRepository {
    private final ContentDatabase db;
    private final VideoDao videoDao;

    public ContentRepository(ContentDatabase contentDatabase, VideoDao videoDao) {
        this.db = contentDatabase;
        this.videoDao = videoDao;
    }

    public LiveData<List<VideoItem>> getVideoByTag(@Nullable String tag, int limit) {
        // you may fetch from network, save to database
        ....
        return videoDao.getVideosByTag(tag, limit);
    }
```

```
}
```

Use in ViewModel:

```
ContentRepository contentRepository = ...;
contentRepository.getVideoByTag(tag, limit);
```

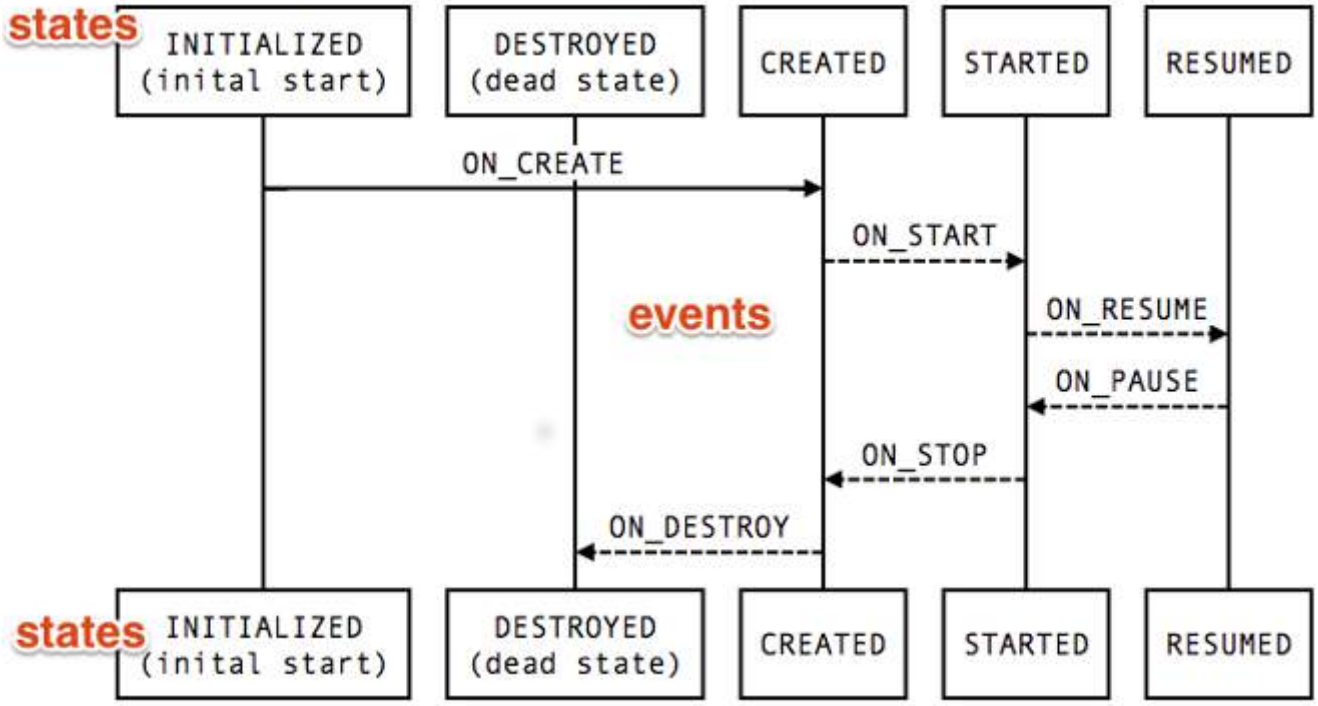
Section 220.5: Custom LiveData

You may write custom LiveData, if you need custom logic.
Don't write custom class, if you only need to transform data (use Transformations class)

```
public class LocationLiveData extends LiveData<Location> {
    private LocationManager locationManager;

    private LocationListener listener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            setValue(location);
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            // Do something
        }

        @Override
        public void onProviderEnabled(String provider) {
            // Do something
        }

        @Override
        public void onProviderDisabled(String provider) {
            // Do something
        }
    };

    public LocationLiveData(Context context) {
        locationManager = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
    }

    @Override
    protected void onActive() {
        // We have observers, start working
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, listener);
    }

    @Override
    protected void onInactive() {
        // We have no observers, stop working
        locationManager.removeUpdates(listener);
    }
}
```

Section 220.6: Custom Lifecycle-aware component

Each UI component lifecycle changed as shown at image.



You may create component, that will be notified on lifecycle state change:

```
public class MyLocationListener implements LifecycleObserver {
    private boolean enabled = false;
    private Lifecycle lifecycle;
    public MyLocationListener(Context context, Lifecycle lifecycle, Callback callback) {
       ...
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_START)
    void start() {
        if (enabled) {
           // connect
        }
    }

    public void enable() {
        enabled = true;
        if (lifecycle.getState().isAtLeast(STARTED)) {
            // connect if not connected
        }
    }

    @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
    void stop() {
        // disconnect if connected
    }
}
```

Chapter 221: Jackson

Jackson is a multi-purpose Java library for processing JSON. Jackson aims to be the best possible combination of fast, correct, lightweight, and ergonomic for developers.

Jackson features :

Multi processing mode, and very good collaboration

Not only annotations, but also mixed annotations

Fully support generic types

Support polymorphic types

Section 221.1: Full Data Binding Example

JSON data

```
{
  "name" : { "first" : "Joe", "last" : "Sixpack" },
  "gender" : "MALE",
  "verified" : false,
  "userImage" : "keliuyue"
}
```

It takes two lines of Java to turn it into a User instance:

```
ObjectMapper mapper = new ObjectMapper(); // can reuse, share globally
User user = mapper.readValue(new File("user.json"), User.class);
```

User.class

```
public class User {

    public enum Gender {MALE, FEMALE};

    public static class Name {
        private String _first, _last;

        public String getFirst() {
            return _first;
        }

        public String getLast() {
            return _last;
        }

        public void setFirst(String s) {
            _first = s;
        }

        public void setLast(String s) {
            _last = s;
        }
    }
```

```
    private Gender _gender;
    private Name _name;
    private boolean _isVerified;
    private byte[] _userImage;

    public Name getName() {
        return _name;
    }

    public boolean isVerified() {
        return _isVerified;
    }

    public Gender getGender() {
        return _gender;
    }

    public byte[] getUserImage() {
        return _userImage;
    }

    public void setName(Name n) {
        _name = n;
    }

    public void setVerified(boolean b) {
        _isVerified = b;
    }

    public void setGender(Gender g) {
        _gender = g;
    }

    public void setUserImage(byte[] b) {
        _userImage = b;
    }
}
```

Marshalling back to JSON is similarly straightforward:

```
mapper.writeValue(new File("user-modified.json"), user);
```

Chapter 222: Smartcard

Section 222.1: Smart card send and receive

For connection, here is a snippet to help you understand:

```
//Allows you to enumerate and communicate with connected USB devices.
UsbManager mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);
//Explicitly asking for permission
final String ACTION_USB_PERMISSION = "com.android.example.USB_PERMISSION";
PendingIntent mPermissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(ACTION_USB_PERMISSION), 0);
HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();

UsbDevice device = deviceList.get("//the device you want to work with");
if (device != null) {
    mUsbManager.requestPermission(device, mPermissionIntent);
}
```

Now you have to understand that in java the communication takes place using package javax.smarcard which is not available for Android so take a look here for getting an idea as to how you can communicate or send/receive APDU (smartcard command).

Now as told in the answer mentioned above

You cannot simply send an APDU (smartcard command) over the bulk-out endpoint and expect to receive a response APDU over the bulk-in endpoint. For getting the endpoints see the code snippet below :

```
UsbEndpoint epOut = null, epIn = null;
UsbInterface usbInterface;

UsbDeviceConnection connection = mUsbManager.openDevice(device);

    for (int i = 0; i < device.getInterfaceCount(); i++) {
        usbInterface = device.getInterface(i);
        connection.claimInterface(usbInterface, true);

        for (int j = 0; j < usbInterface.getEndpointCount(); j++) {
            UsbEndpoint ep = usbInterface.getEndpoint(j);

            if (ep.getType() == UsbConstants.USB_ENDPOINT_XFER_BULK) {
                if (ep.getDirection() == UsbConstants.USB_DIR_OUT) {
                    // from host to device
                    epOut = ep;

                } else if (ep.getDirection() == UsbConstants.USB_DIR_IN) {
                    // from device to host
                    epIn = ep;
                }
            }
        }
    }
```

Now you have the bulk-in and bulk-out endpoints to send and receive APDU command and APDU response blocks:

For sending commands, see the code snippet below:

```
public void write(UsbDeviceConnection connection, UsbEndpoint epOut, byte[] command) {
```

```
    result = new StringBuilder();
    connection.bulkTransfer(epOut, command, command.length, TIMEOUT);
    //For Printing logs you can use result variable
    for (byte bb : command) {
        result.append(String.format(" %02X ", bb));
    }
}
```

And for receive/ read a response see the code snippet below :

```
public int read(UsbDeviceConnection connection, UsbEndpoint epIn) {
result = new StringBuilder();
final byte[] buffer = new byte[epIn.getMaxPacketSize()];
int byteCount = 0;
byteCount = connection.bulkTransfer(epIn, buffer, buffer.length, TIMEOUT);

//For Printing logs you can use result variable
if (byteCount >= 0) {
    for (byte bb : buffer) {
        result.append(String.format(" %02X ", bb));
    }

    //Buffer received was : result.toString()
} else {
    //Something went wrong as count was : " + byteCount
}

return byteCount;
}
```

Now if you see this answer here the 1st command to be sent is :

PC_to_RDR_IccPowerOn command to activate the card.

which you can create by reading section 6.1.1 of the USB Device Class Specifications doc here.

Now let's take an example of this command like the one here: `6200000000000000000` How you can send this is :

```
write(connection, epOut, "62000000000000000000");
```

Now after you have successfully sent the APDU command, you can read the response using :

```
read(connection, epIn);
```

And receive something like

```
80 18000000 00 00 00 00 00 3BBF11008131FE45455041000000000000000000000000F1
```

Now the response received in the code here will be in the `result` variable of `read()` method from code

Chapter 223: Security

Section 223.1: Verifying App Signature - Tamper Detection

This technique details how to ensure that your .apk has been signed with your developer certificate, and leverages the fact that the certificate remains consistent and that only you have access to it. We can break this technique into 3 simple steps:

- Find your developer certificate signature.
- Embed your signature in a String constant in your app.
- Check that the signature at runtime matches our embedded developer signature.

Here's the code snippet:

```
private static final int VALID = 0;
private static final int INVALID = 1;

public static int checkAppSignature(Context context) {

try {
      PackageInfo packageInfo =
context.getPackageManager().getPackageInfo(context.getPackageName(),
PackageManager.GET_SIGNATURES);

      for (Signature signature : packageInfo.signatures) {

        byte[] signatureBytes = signature.toByteArray();

        MessageDigest md = MessageDigest.getInstance("SHA");

        md.update(signature.toByteArray());

        final String currentSignature = Base64.encodeToString(md.digest(), Base64.DEFAULT);

        Log.d("REMOVE_ME", "Include this string as a value for SIGNATURE:" + currentSignature);

        //compare signatures
        if (SIGNATURE.equals(currentSignature)){
          return VALID;
        };
      }
    } catch (Exception e) {
        //assumes an issue in checking signature., but we let the caller decide on what to do.
    }

    return INVALID;

}
```

Chapter 224: How to store passwords securely

Section 224.1: Using AES for salted password encryption

This examples uses the AES algorithm for encrypting passwords. The salt length can be up to 128 bit.

We are using the `SecureRandom` class to generate a salt, which is combined with the password to generate a secret key. The classes used are already existing in Android packages `javax.crypto` and `java.security`.

Once a key is generated, we have to preserve this key in a variable or store it. We are storing it among the shared preferences in the value `S_KEY`. Then, a password is encrypted using the `doFinal` method of the `Cipher` class once it is initialised in `ENCRYPT_MODE`. Next, the encrypted password is converted from a byte array into a string and stored among the shared preferences. The key used to generate an encrypted password can be used to decrypt the password in a similar way:

```
public class MainActivity extends AppCompatActivity {
    public static final String PROVIDER = "BC";
    public static final int SALT_LENGTH = 20;
    public static final int IV_LENGTH = 16;
    public static final int PBE_ITERATION_COUNT = 100;

    private static final String RANDOM_ALGORITHM = "SHA1PRNG";
    private static final String HASH_ALGORITHM = "SHA-512";
    private static final String PBE_ALGORITHM = "PBEWithSHA256And256BitAES-CBC-BC";
    private static final String CIPHER_ALGORITHM = "AES/CBC/PKCS5Padding";
    public static final String SECRET_KEY_ALGORITHM = "AES";
    private static final String TAG = "EncryptionPassword";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String originalPassword = "ThisIsAndroidStudio%$";
        Log.e(TAG, "originalPassword => " + originalPassword);
        String encryptedPassword = encryptAndStorePassword(originalPassword);
        Log.e(TAG, "encryptedPassword => " + encryptedPassword);
        String decryptedPassword = decryptAndGetPassword();
        Log.e(TAG, "decryptedPassword => " + decryptedPassword);
    }

    private String decryptAndGetPassword() {
        SharedPreferences prefs = getSharedPreferences("pswd", MODE_PRIVATE);
        String encryptedPasswrd = prefs.getString("token", "");
        String passwrd = "";
        if (encryptedPasswrd!=null && !encryptedPasswrd.isEmpty()) {
            try {
                String output = prefs.getString("S_KEY", "");
                byte[] encoded = hexStringToByteArray(output);
                SecretKey aesKey = new SecretKeySpec(encoded, SECRET_KEY_ALGORITHM);
                passwrd = decrypt(aesKey, encryptedPasswrd);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return passwrd;
    }

    public String encryptAndStorePassword(String password) {
```

```
        SharedPreferences.Editor editor = getSharedPreferences("pswd", MODE_PRIVATE).edit();
        String encryptedPassword = "";
        if (password!=null && !password.isEmpty()) {
            SecretKey secretKey = null;
            try {
                secretKey = getSecretKey(password, generateSalt());

                byte[] encoded = secretKey.getEncoded();
                String input = byteArrayToHexString(encoded);
                editor.putString("S_KEY", input);
                encryptedPassword = encrypt(secretKey, password);
            } catch (Exception e) {
                e.printStackTrace();
            }
            editor.putString("token", encryptedPassword);
            editor.commit();
        }
        return encryptedPassword;
    }

    public static String encrypt(SecretKey secret, String cleartext) throws Exception {
        try {
            byte[] iv = generateIv();
            String ivHex = byteArrayToHexString(iv);
            IvParameterSpec ivspec = new IvParameterSpec(iv);

            Cipher encryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM, PROVIDER);
            encryptionCipher.init(Cipher.ENCRYPT_MODE, secret, ivspec);
            byte[] encryptedText = encryptionCipher.doFinal(cleartext.getBytes("UTF-8"));
            String encryptedHex = byteArrayToHexString(encryptedText);

            return ivHex + encryptedHex;

        } catch (Exception e) {
            Log.e("SecurityException", e.getCause().getLocalizedMessage());
            throw new Exception("Unable to encrypt", e);
        }
    }

    public static String decrypt(SecretKey secret, String encrypted) throws Exception {
        try {
            Cipher decryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM, PROVIDER);
            String ivHex = encrypted.substring(0, IV_LENGTH * 2);
            String encryptedHex = encrypted.substring(IV_LENGTH * 2);
            IvParameterSpec ivspec = new IvParameterSpec(hexStringToByteArray(ivHex));
            decryptionCipher.init(Cipher.DECRYPT_MODE, secret, ivspec);
            byte[] decryptedText = decryptionCipher.doFinal(hexStringToByteArray(encryptedHex));
            String decrypted = new String(decryptedText, "UTF-8");
            return decrypted;
        } catch (Exception e) {
            Log.e("SecurityException", e.getCause().getLocalizedMessage());
            throw new Exception("Unable to decrypt", e);
        }
    }

    public static String generateSalt() throws Exception {
        try {
            SecureRandom random = SecureRandom.getInstance(RANDOM_ALGORITHM);
            byte[] salt = new byte[SALT_LENGTH];
            random.nextBytes(salt);
            String saltHex = byteArrayToHexString(salt);
            return saltHex;
```

```
        } catch (Exception e) {
            throw new Exception("Unable to generate salt", e);
        }
    }

    public static String byteArrayToHexString(byte[] b) {
        StringBuffer sb = new StringBuffer(b.length * 2);
        for (int i = 0; i < b.length; i++) {
            int v = b[i] & 0xff;
            if (v < 16) {
                sb.append('0');
            }
            sb.append(Integer.toHexString(v));
        }
        return sb.toString().toUpperCase();
    }

    public static byte[] hexStringToByteArray(String s) {
        byte[] b = new byte[s.length() / 2];
        for (int i = 0; i < b.length; i++) {
            int index = i * 2;
            int v = Integer.parseInt(s.substring(index, index + 2), 16);
            b[i] = (byte) v;
        }
        return b;
    }

    public static SecretKey getSecretKey(String password, String salt) throws Exception {
        try {
            PBEKeySpec pbeKeySpec = new PBEKeySpec(password.toCharArray(),
hexStringToByteArray(salt), PBE_ITERATION_COUNT, 256);
            SecretKeyFactory factory = SecretKeyFactory.getInstance(PBE_ALGORITHM, PROVIDER);
            SecretKey tmp = factory.generateSecret(pbeKeySpec);
            SecretKey secret = new SecretKeySpec(tmp.getEncoded(), SECRET_KEY_ALGORITHM);
            return secret;
        } catch (Exception e) {
            throw new Exception("Unable to get secret key", e);
        }
    }

    private static byte[] generateIv() throws NoSuchAlgorithmException, NoSuchProviderException {
        SecureRandom random = SecureRandom.getInstance(RANDOM_ALGORITHM);
        byte[] iv = new byte[IV_LENGTH];
        random.nextBytes(iv);
        return iv;
    }
}
```

Chapter 225: Secure SharedPreferences

Parameter	Definition
input	String value to encrypt or decrypt.

Shared Preferences are **key-value based XML files**. It is located under `/data/data/package_name/shared_prefs/<filename.xml>`.

So a user with root privileges can navigate to this location and can change its values. If you want to protect values in your shared preferences, you can write a simple encryption and decryption mechanism.

You should know tough, that Shared Preferences were never built to be secure, it's just a simple way to persist data.

Section 225.1: Securing a Shared Preference

Simple Codec

Here to illustrate the working principle we can use simple encryption and decryption as follows.

```
public static String encrypt(String input) {
    // Simple encryption, not very strong!
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```

Implementation Technique

```
public static String pref_name = "My_Shared_Pref";

// To Write
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(encrypt("password"), encrypt("my_dummy_pass"));
editor.apply(); // Or commit if targeting old devices

// To Read
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
String passEncrypted = preferences.getString(encrypt("password"), encrypt("default_value"));
String password = decrypt(passEncrypted);
```

Chapter 226: Secure SharedPreferences

Parameter	Definition
input	String value to encrypt or decrypt.

Shared Preferences are **key-value based XML files**. It is located under /data/data/package_name/shared_prefs/<filename.xml>.

So a user with root privileges can navigate to this location and can change its values. If you want to protect values in your shared preferences, you can write a simple encryption and decryption mechanism.

You should know tough, that Shared Preferences were never built to be secure, it's just a simple way to persist data.

Section 226.1: Securing a Shared Preference

Simple Codec

Here to illustrate the working principle we can use simple encryption and decryption as follows.

```
public static String encrypt(String input) {
    // Simple encryption, not very strong!
    return Base64.encodeToString(input.getBytes(), Base64.DEFAULT);
}

public static String decrypt(String input) {
    return new String(Base64.decode(input, Base64.DEFAULT));
}
```

Implementation Technique

```
public static String pref_name = "My_Shared_Pref";

// To Write
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(encrypt("password"), encrypt("my_dummy_pass"));
editor.apply(); // Or commit if targeting old devices

// To Read
SharedPreferences preferences = getSharedPreferences(pref_name, MODE_PRIVATE);
String passEncrypted = preferences.getString(encrypt("password"), encrypt("default_value"));
String password = decrypt(passEncrypted);
```

Chapter 227: SQLite

SQLite is a relational database management system written in C. To begin working with SQLite databases within the Android framework, define a class that extends SQLiteOpenHelper, and customize as needed.

Section 227.1: onUpgrade() method

`SQLiteOpenHelper` is a helper class to manage database creation and version management.

In this class, the `onUpgrade()` method is responsible for upgrading the database when you make changes to the schema. It is called when the database file already exists, but its version is lower than the one specified in the current version of the app. For each database version, the specific changes you made have to be applied.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Loop through each version when an upgrade occurs.
    for (int version = oldVersion + 1; version <= newVersion; version++) {
        switch (version) {

            case 2:
                // Apply changes made in version 2
                db.execSQL(
                    "ALTER TABLE " +
                    TABLE_PRODUCTS +
                    " ADD COLUMN " +
                    COLUMN_DESCRIPTION +
                    " TEXT;"
                );
                break;

            case 3:
                // Apply changes made in version 3
                db.execSQL(CREATE_TABLE_TRANSACTION);
                break;
        }
    }
}
```

Section 227.2: Reading data from a Cursor

Here is an example of a method that would live inside a `SQLiteOpenHelper` subclass. It uses the `searchTerm` String to filter the results, iterates through the Cursor's contents, and returns those contents in a `List` of `Product` Objects.

First, define the `Product` POJO class that will be the container for each row retrieved from the database:

```
public class Product {
  long mId;
  String mName;
  String mDescription;
  float mValue;
  public Product(long id, String name, String description, float value) {
    mId = id;
    mName = name;
    mDescription = description;
    mValue = value;
  }
}
```

Then, define the method that will query the database, and return a `List` of `Product` Objects:

```
public List<Product> searchForProducts(String searchTerm) {

    // When reading data one should always just get a readable database.
    final SQLiteDatabase database = this.getReadableDatabase();

    final Cursor cursor = database.query(
            // Name of the table to read from
            TABLE_NAME,

            // String array of the columns which are supposed to be read
            new String[]{COLUMN_NAME, COLUMN_DESCRIPTION, COLUMN_VALUE},

            // The selection argument which specifies which row is read.
            // ? symbols are parameters.
            COLUMN_NAME + " LIKE ?",

            // The actual parameters values for the selection as a String array.
            // ? above take the value from here
            new String[]{"%" + searchTerm + "%"},

            // GroupBy clause. Specify a column name to group similar values
            // in that column together.
            null,

            // Having clause. When using the GroupBy clause this allows you to
            // specify which groups to include.
            null,

            // OrderBy clause. Specify a column name here to order the results
            // according to that column. Optionally append ASC or DESC to specify
            // an ascending or descending order.
            null
    );

    // To increase performance first get the index of each column in the cursor
    final int idIndex = cursor.getColumnIndex(COLUMN_ID);
    final int nameIndex = cursor.getColumnIndex(COLUMN_NAME);
    final int descriptionIndex = cursor.getColumnIndex(COLUMN_DESCRIPTION);
    final int valueIndex = cursor.getColumnIndex(COLUMN_VALUE);

    try {

        // If moveToFirst() returns false then cursor is empty
        if (!cursor.moveToFirst()) {
            return new ArrayList<>();
        }

        final List<Product> products = new ArrayList<>();

        do {

            // Read the values of a row in the table using the indexes acquired above
            final long id = cursor.getLong(idIndex);
            final String name = cursor.getString(nameIndex);
            final String description = cursor.getString(descriptionIndex);
            final float value = cursor.getFloat(valueIndex);

            products.add(new Product(id, name, description, value));

        } while (cursor.moveToNext());
```

```
        return products;

    } finally {
        // Don't forget to close the Cursor once you are done to avoid memory leaks.
        // Using a try/finally like in this example is usually the best way to handle this
        cursor.close();

        // close the database
        database.close();
    }
}
```

Section 227.3: Using the SQLiteOpenHelper class

```
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Example.db";
    private static final int DATABASE_VERSION = 3;

    // For all Primary Keys _id should be used as column name
    public static final String COLUMN_ID = "_id";

    // Definition of table and column names of Products table
    public static final String TABLE_PRODUCTS = "Products";
    public static final String COLUMN_NAME = "Name";
    public static final String COLUMN_DESCRIPTION = "Description";
    public static final String COLUMN_VALUE = "Value";

    // Definition of table and column names of Transactions table
    public static final String TABLE_TRANSACTIONS = "Transactions";
    public static final String COLUMN_PRODUCT_ID = "ProductId";
    public static final String COLUMN_AMOUNT = "Amount";

    // Create Statement for Products Table
    private static final String CREATE_TABLE_PRODUCT = "CREATE TABLE " + TABLE_PRODUCTS + "  (" +
            COLUMN_ID + " INTEGER PRIMARY KEY, " +
            COLUMN_DESCRIPTION + " TEXT, " +
            COLUMN_NAME + " TEXT, " +
            COLUMN_VALUE + " REAL" +
            ");";

    // Create Statement for Transactions Table
    private static final String CREATE_TABLE_TRANSACTION = "CREATE TABLE " + TABLE_TRANSACTIONS + "
 (" +
            COLUMN_ID + " INTEGER PRIMARY KEY," +
            COLUMN_PRODUCT_ID + " INTEGER," +
            COLUMN_AMOUNT + " INTEGER," +
            " FOREIGN KEY (" + COLUMN_PRODUCT_ID + ") REFERENCES " + TABLE_PRODUCTS + "(" +
COLUMN_ID + ")" +
            ");";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // onCreate should always create your most up to date database
        // This method is called when the app is newly installed
        db.execSQL(CREATE_TABLE_PRODUCT);
        db.execSQL(CREATE_TABLE_TRANSACTION);
    }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // onUpgrade is responsible for upgrading the database when you make
        // changes to the schema. For each version the specific changes you made
        // in that version have to be applied.
        for (int version = oldVersion + 1; version <= newVersion; version++) {
            switch (version) {

                case 2:
                    db.execSQL("ALTER TABLE " + TABLE_PRODUCTS + " ADD COLUMN " +
COLUMN_DESCRIPTION + " TEXT;");
                    break;

                case 3:
                    db.execSQL(CREATE_TABLE_TRANSACTION);
                    break;
            }
        }
    }
}
```

Section 227.4: Insert data into database

```
// You need a writable database to insert data
final SQLiteDatabase database = openHelper.getWritableDatabase();

// Create a ContentValues instance which contains the data for each column
// You do not need to specify a value for the PRIMARY KEY column.
// Unique values for these are automatically generated.
final ContentValues values = new ContentValues();
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// This call performs the update
// The return value is the rowId or primary key value for the new row!
// If this method returns -1 then the insert has failed.
final int id = database.insert(
        TABLE_NAME, // The table name in which the data will be inserted
        null,       // String: optional; may be null. If your provided values is empty,
                    // no column names are known and an empty row can't be inserted.
                    // If not set to null, this parameter provides the name
                    // of nullable column name to explicitly insert a NULL

        values      // The ContentValues instance which contains the data
);
```

Section 227.5: Bulk insert

Here is an example of inserting large chunks of data at once. All the data you want to insert is gathered inside of a ContentValues array.

```
@Override
public int bulkInsert(Uri uri, ContentValues[] values) {
    int count = 0;
    String table = null;

    int uriType = IChatContract.MessageColumns.uriMatcher.match(uri);
    switch (uriType) {
        case IChatContract.MessageColumns.MESSAGES:
```

```
            table = IChatContract.MessageColumns.TABLE_NAME;
            break;
    }
    mDatabase.beginTransaction();
    try {
        for (ContentValues cv : values) {
            long rowID = mDatabase.insert(table, " ", cv);
            if (rowID <= 0) {
                throw new SQLException("Failed to insert row into " + uri);
            }
        }
        mDatabase.setTransactionSuccessful();
        getContext().getContentResolver().notifyChange(uri, null);
        count = values.length;
    } finally {
        mDatabase.endTransaction();
    }
    return count;
}
```

And here is an example of how to use it:

```
ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//add whatever you like to the valueList
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);
```

Section 227.6: Create a Contract, Helper and Provider for SQLite in Android

DBContract.java

```
//Define the tables and columns of your local database
public final class DBContract {
/*Content Authority its a name for the content provider, is convenient to use the package app name to
be unique on the device */

    public static final String CONTENT_AUTHORITY = "com.yourdomain.yourapp";

    //Use CONTENT_AUTHORITY to create all the database URI's that the app will use to link the
content provider.
    public static final Uri BASE_CONTENT_URI = Uri.parse("content://" + CONTENT_AUTHORITY);

    /*the name of the uri that can be the same as the name of your table.
    this will translate to content://com.yourdomain.yourapp/user/ as a valid URI
    */
    public static final String PATH_USER = "User";

    // To prevent someone from accidentally instantiating the contract class,
    // give it an empty constructor.
    public DBContract () {}

    //Intern class that defines the user table
    public static final class UserEntry implements BaseColumns {
        public static final URI CONTENT_URI =
BASE_CONTENT_URI.buildUpon().appendPath(PATH_USER).build();

        public static final String CONTENT_TYPE =
ContentResolver.CURSOR_DIR_BASE_TYPE+"/"+CONTENT_AUTHORITY+"/"+PATH_USER;
```

```
        //Name of the table
        public static final String TABLE_NAME="User";

        //Columns of the user table
        public static final String COLUMN_Name="Name";
        public static final String COLUMN_Password="Password";

        public static Uri buildUri(long id){
            return ContentUris.withAppendedId(CONTENT_URI,id);
        }
}
```

DBHelper.java

```
public class DBHelper extends SQLiteOpenHelper{

//if you change the schema of the database, you must increment this number
private static final int DATABASE_VERSION=1;
static final String DATABASE_NAME="mydatabase.db";
private static DBHelper mInstance=null;
public static DBHelper getInstance(Context ctx){
    if(mInstance==null){
    mInstance= new DBHelper(ctx.getApplicationContext());
    }
    return mInstance;
}

public DBHelper(Context context){
    super(context,DATABASE_NAME,null,DATABASE_VERSION);
}

public int GetDatabase_Version() {
    return DATABASE_VERSION;
}

@Override
public void onCreate(SQLiteDatabase sqLiteDatabase){
//Create the table users
final String SQL_CREATE_TABLE_USERS="CREATE TABLE "+UserEntry.TABLE_NAME+ " ("+
UserEntry._ID+" INTEGER PRIMARY KEY, "+
UserEntry.COLUMN_Name+" TEXT , "+
UserEntry.COLUMN_Password+" TEXT "+
" ); ";

sqLiteDatabase.execSQL(SQL_CREATE_TABLE_USERS);

}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion, int newVersion) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + UserEntry.TABLE_NAME);
}

}
```

DBProvider.java

```
public class DBProvider extends ContentProvider {

    private static final UriMatcher sUriMatcher = buildUriMatcher();
    private DBHelper mDBHelper;
```

```
    private Context mContext;

    static final int USER = 100;

    static UriMatcher buildUriMatcher() {

        final UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
        final String authority = DBContract.CONTENT_AUTHORITY;

        matcher.addURI(authority, DBContract.PATH_USER, USER);

        return matcher;
    }

    @Override
    public boolean onCreate() {
        mDBHelper = new DBHelper(getContext());
        return false;
    }

    public PeaberryProvider(Context context) {
        mDBHelper = DBHelper.getInstance(context);
        mContext = context;
    }

    @Override
    public String getType(Uri uri) {
        // determine what type of Uri is
        final int match = sUriMatcher.match(uri);

        switch (match) {
            case USER:
                return DBContract.UserEntry.CONTENT_TYPE;

            default:
                throw new UnsupportedOperationException("Uri unknown: " + uri);
        }
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
                        String sortOrder) {
        Cursor retCursor;
        try {
            switch (sUriMatcher.match(uri)) {
                case USER: {
                    retCursor = mDBHelper.getReadableDatabase().query(
                            DBContract.UserEntry.TABLE_NAME,
                            projection,
                            selection,
                            selectionArgs,
                            null,
                            null,
                            sortOrder
                    );
                    break;
                }
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
        } catch (Exception ex) {
            Log.e("Cursor", ex.toString());
```

```
        } finally {
            mDBHelper.close();
        }
        return null;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        Uri returnUri;
        try {

            switch (match) {
                case USER: {
                    long _id = db.insert(DBContract.UserEntry.TABLE_NAME, null, values);
                    if (_id > 0)
                        returnUri = DBContract.UserEntry.buildUri(_id);
                    else
                        throw new android.database.SQLException("Error at inserting row in " +
uri);
                    break;
                }
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            mContext.getContentResolver().notifyChange(uri, null);
            return returnUri;
        } catch (Exception ex) {
            Log.e("Insert", ex.toString());
            db.close();
        } finally {
            db.close();
        }
        return null;
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        final SQLiteDatabase db = DBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int deletedRows;
        if (null == selection) selection = "1";
        try {
            switch (match) {
                case USER:
                    deletedRows = db.delete(
                            DBContract.UserEntry.TABLE_NAME, selection, selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            if (deletedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return deletedRows;
        } catch (Exception ex) {
            Log.e("Insert", ex.toString());
        } finally {
            db.close();
        }
        return 0;
```

```
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
        final SQLiteDatabase db = mDBHelper.getWritableDatabase();
        final int match = sUriMatcher.match(uri);
        int updatedRows;
        try {
            switch (match) {
                case USER:
                    updatedRows = db.update(DBContract.UserEntry.TABLE_NAME, values, selection,
selectionArgs);
                    break;
                default:
                    throw new UnsupportedOperationException("Uri unknown: " + uri);
            }
            if (updatedRows != 0) {
                mContext.getContentResolver().notifyChange(uri, null);
            }
            return updatedRows;
        } catch (Exception ex) {
            Log.e("Update", ex.toString());
        } finally {
            db.close();
        }
        return -1;
    }

}
```

How to Use:

```
public void InsertUser() {
    try {
        ContentValues userValues = getUserData("Jhon","XXXXX");
        DBProvider dbProvider = new DBProvider(mContext);
        dbProvider.insert(UserEntry.CONTENT_URI, userValues);

    } catch (Exception ex) {
        Log.e("Insert", ex.toString());
    }
}

public ContentValues getUserData(String name, String pass) {
    ContentValues userValues = new ContentValues();
    userValues.put(UserEntry.COLUMN_Name, name);
    userValues.put(UserEntry.COLUMN_Password, pass);
    return userValues;
}
```

Section 227.7: Delete row(s) from the table

To delete all rows from the table

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

db.delete(TABLE_NAME, null, null);
db.close();
```

To delete all rows from the table and get the count of the deleted row in return value

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

To delete row(s) with `WHERE` condition

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

String whereClause = KEY_NAME + " = ?";
String[] whereArgs = new String[]{String.valueOf(KEY_VALUE)};

//for multiple condition, join them with AND
//String whereClause = KEY_NAME1 + " = ? AND " + KEY_NAME2 + " = ?";
//String[] whereArgs = new String[]{String.valueOf(KEY_VALUE1), String.valueOf(KEY_VALUE2)};

int numRowsDeleted = db.delete(TABLE_NAME, whereClause, whereArgs);
db.close();
```

Section 227.8: Updating a row in a table

```
// You need a writable database to update a row
final SQLiteDatabase database = openHelper.getWritableDatabase();

// Create a ContentValues instance which contains the up to date data for each column
// Unlike when inserting data you need to specify the value for the PRIMARY KEY column as well
final ContentValues values = new ContentValues();
values.put(COLUMN_ID, model.getId());
values.put(COLUMN_NAME, model.getName());
values.put(COLUMN_DESCRIPTION, model.getDescription());
values.put(COLUMN_VALUE, model.getValue());

// This call performs the update
// The return value tells you how many rows have been updated.
final int count = database.update(
        TABLE_NAME,         // The table name in which the data will be updated
        values,             // The ContentValues instance with the new data
        COLUMN_ID + " = ?", // The selection which specifies which row is updated. ? symbols are
parameters.
        new String[] {      // The actual parameters for the selection as a String[].
                String.valueOf(model.getId())
        }
);
```

Section 227.9: Performing a Transaction

Transactions can be used to make multiple changes to the database atomically. Any normal transaction follows this pattern:

```
// You need a writable database to perform transactions
final SQLiteDatabase database = openHelper.getWritableDatabase();

// This call starts a transaction
database.beginTransaction();
```

```
// Using try/finally is essential to reliably end transactions even
// if exceptions or other problems occur.
try {

    // Here you can make modifications to the database
    database.insert(TABLE_CARS, null, productValues);
    database.update(TABLE_BUILDINGS, buildingValues, COLUMN_ID + " = ?", new String[] {
String.valueOf(buildingId) });

    // This call marks a transaction as successful.
    // This causes the changes to be written to the database once the transaction ends.
    database.setTransactionSuccessful();
} finally {
    // This call ends a transaction.
    // If setTransactionSuccessful() has not been called then all changes
    // will be rolled back and the database will not be modified.
    database.endTransaction();
}
```

Calling `beginTransaction()` inside of an active transactions has no effect.

Section 227.10: Create Database from assets folder

Put your dbname.sqlite or dbname.db file in assets folder of your project.

```
public class Databasehelper extends SQLiteOpenHelper {
        public static final String TAG = Databasehelper.class.getSimpleName();
        public static int flag;
        // Exact Name of you db file that you put in assets folder with extension.
        static String DB_NAME = "dbname.sqlite";
        private final Context myContext;
        String outFileName = "";
        private String DB_PATH;
        private SQLiteDatabase db;

        public Databasehelper(Context context) {
            super(context, DB_NAME, null, 1);
            this.myContext = context;
            ContextWrapper cw = new ContextWrapper(context);
            DB_PATH = cw.getFilesDir().getAbsolutePath() + "/databases/";
            Log.e(TAG, "Databasehelper: DB_PATH " + DB_PATH);
            outFileName = DB_PATH + DB_NAME;
            File file = new File(DB_PATH);
            Log.e(TAG, "Databasehelper: " + file.exists());
            if (!file.exists()) {
                file.mkdir();
            }
        }

        /**
         * Creates a empty database on the system and rewrites it with your own database.
         */
        public void createDataBase() throws IOException {
            boolean dbExist = checkDataBase();
            if (dbExist) {
                //do nothing - database already exist
            } else {
                //By calling this method and empty database will be created into the default system
path
                //of your application so we are gonna be able to overwrite that database with our
database.
```

```
                this.getReadableDatabase();
                try {
                    copyDataBase();
                } catch (IOException e) {
                    throw new Error("Error copying database");
                }
            }
        }

        /**
         * Check if the database already exist to avoid re-copying the file each time you open the
application.
         *
         * @return true if it exists, false if it doesn't
         */
        private boolean checkDataBase() {
            SQLiteDatabase checkDB = null;
            try {
                checkDB = SQLiteDatabase.openDatabase(outFileName, null,
SQLiteDatabase.OPEN_READWRITE);
            } catch (SQLiteException e) {
                try {
                    copyDataBase();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }

            if (checkDB != null) {
                checkDB.close();
            }
            return checkDB != null ? true : false;
        }

        /**
         * Copies your database from your local assets-folder to the just created empty database in
the
         * system folder, from where it can be accessed and handled.
         * This is done by transfering bytestream.
         */

        private void copyDataBase() throws IOException {

            Log.i("Database",
                    "New database is being copied to device!");
            byte[] buffer = new byte[1024];
            OutputStream myOutput = null;
            int length;
            // Open your local db as the input stream
            InputStream myInput = null;
            try {
                myInput = myContext.getAssets().open(DB_NAME);
                // transfer bytes from the inputfile to the
                // outputfile
                myOutput = new FileOutputStream(DB_PATH + DB_NAME);
                while ((length = myInput.read(buffer)) > 0) {
                    myOutput.write(buffer, 0, length);
                }
                myOutput.close();
                myOutput.flush();
                myInput.close();
                Log.i("Database",
```

```
                    "New database has been copied to device!");
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        public void openDataBase() throws SQLException {
            //Open the database
            String myPath = DB_PATH + DB_NAME;
            db = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READWRITE);
            Log.e(TAG, "openDataBase: Open " + db.isOpen());
        }

        @Override
        public synchronized void close() {
            if (db != null)
                db.close();
            super.close();
        }

        public void onCreate(SQLiteDatabase arg0) {

        }

        @Override
        public void onUpgrade(SQLiteDatabase arg0, int arg1, int arg2) {

        }
    }
```

Here is How you can access database object to your activity.

```
// Create Databasehelper class object in your activity.
   private Databasehelper db;
```

Then in onCreate Method initialize it and call createDatabase() Method as show below.

```
db = new Databasehelper(MainActivity.this);
        try {
            db.createDataBase();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Perform all of your insert, update, delete and select operation as shown below.

```
String query = "select Max(Id) as Id from " + TABLE_NAME;
        db.openDataBase();
        int count = db.getId(query);
        db.close();
```

Section 227.11: Store image into SQLite

Setting Up the database

```
public class DatabaseHelper extends SQLiteOpenHelper {
    // Database Version
    private static final int DATABASE_VERSION = 1;
```

```
    // Database Name
    private static final String DATABASE_NAME = "database_name";

    // Table Names
    private static final String DB_TABLE = "table_image";

    // column names
    private static final String KEY_NAME = "image_name";
    private static final String KEY_IMAGE = "image_data";

    // Table create statement
    private static final String CREATE_TABLE_IMAGE = "CREATE TABLE " + DB_TABLE + "("+
                        KEY_NAME + " TEXT," +
                        KEY_IMAGE + " BLOB);";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        // creating table
        db.execSQL(CREATE_TABLE_IMAGE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // on upgrade drop older tables
        db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);

        // create new table
        onCreate(db);
    }
}
```

Insert in the Database:

```
public void addEntry( String name, byte[] image) throws SQLiteException{
    SQLiteDatabase database = this.getWritableDatabase();
    ContentValues cv = new  ContentValues();
    cv.put(KEY_NAME,     name);
    cv.put(KEY_IMAGE,     image);
    database.insert( DB_TABLE, null, cv );
}
```

Retrieving data:

```
 byte[] image = cursor.getBlob(1);
```

Note:

1. Before inserting into database, you need to convert your Bitmap image into byte array first then apply it using database query.
2. When retrieving from database, you certainly have a byte array of image, what you need to do is to convert byte array back to original image. So, you have to make use of BitmapFactory to decode.

Below is an Utility class which I hope could help you:

```
public class DbBitmapUtility {

    // convert from bitmap to byte array
    public static byte[] getBytes(Bitmap bitmap) {
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        bitmap.compress(CompressFormat.PNG, 0, stream);
        return stream.toByteArray();
    }

    // convert from byte array to bitmap
    public static Bitmap getImage(byte[] image) {
        return BitmapFactory.decodeByteArray(image, 0, image.length);
    }
}
```

Section 227.12: Exporting and importing a database

You might want to import and export your database for bacukups for example. Don't forget about the permissions.

```
public void exportDatabase(){
    try
    {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();

        String currentDBPath = "//data//MY.PACKAGE.NAME//databases//MY_DATABASE_NAME";
        String backupDBPath = "MY_DATABASE_FILE.db";
        File currentDB = new File(data, currentDBPath);
        File backupDB = new File(sd, backupDBPath);

        FileChannel src = new FileInputStream(currentDB).getChannel();
        FileChannel dst = new FileOutputStream(backupDB).getChannel();
        dst.transferFrom(src, 0, src.size());
        src.close();
        dst.close();

        Toast.makeText(c, c.getResources().getString(R.string.exporterenToast),
Toast.LENGTH_SHORT).show();
    }
    catch (Exception e) {
        Toast.makeText(c, c.getResources().getString(R.string.portError),
Toast.LENGTH_SHORT).show();
        Log.d("Main", e.toString());
    }
}

public void importDatabase(){
    try
    {
        File sd = Environment.getExternalStorageDirectory();
        File data = Environment.getDataDirectory();

        String currentDBPath = "//data//" + "MY.PACKAGE.NAME" + "//databases//" +
"MY_DATABASE_NAME";
        String backupDBPath = "MY_DATABASE_FILE.db";
        File backupDB = new File(data, currentDBPath);
        File currentDB = new File(sd, backupDBPath);

        FileChannel src = new FileInputStream(currentDB).getChannel();
        FileChannel dst = new FileOutputStream(backupDB).getChannel();
        dst.transferFrom(src, 0, src.size());
```

```
        src.close();
        dst.close();
        Toast.makeText(c, c.getResources().getString(R.string.importerenToast),
Toast.LENGTH_LONG).show();
    }
    catch (Exception e) {
        Toast.makeText(c, c.getResources().getString(R.string.portError),
Toast.LENGTH_SHORT).show();
    }
}
```

Chapter 228: Accessing SQLite databases using the ContentValues class

Section 228.1: Inserting and updating rows in a SQLite database

First, you need to open your SQLite database, which can be done as follows:

```
SQLiteDatabase myDataBase;
String mPath = dbhelper.DATABASE_PATH + dbhelper.DATABASE_NAME;
myDataBase = SQLiteDatabase.openDatabase(mPath, null, SQLiteDatabase.OPEN_READWRITE);
```

After opening the database, you can easily insert or update rows by using the `ContentValues` class. The following examples assume that a first name is given by `str_edtfname` and a last nameby `str_edtlname`. You also need to replace `table_name` by the name of your table that you want to modify.

Inserting data

```
ContentValues values = new ContentValues();
values.put("First_Name", str_edtfname);
values.put("Last_Name", str_edtlname);
myDataBase.insert("table_name", null, values);
```

Updating data

```
ContentValues values = new ContentValues();
values.put("First_Name", str_edtfname);
values.put("Last_Name", str_edtlname);
myDataBase.update("table_name", values, "id" + " = ?", new String[] {id});
```

Chapter 229: Firebase

Firebase is a mobile and web application platform with tools and infrastructure designed to help developers build high-quality apps.

Features

Firebase Cloud Messaging, Firebase Auth, Realtime Database, Firebase Storage, Firebase Hosting, Firebase Test Lab for Android, Firebase Crash Reporting.

Section 229.1: Add Firebase to Your Android Project

Here are simplified steps (based on the official documentation) required to create a Firebase project and connect it with an Android app.

Add Firebase to your app

1. Create a Firebase project in the Firebase console and click **Create New Project**.
2. Click **Add Firebase to your Android app** and follow the setup steps.
3. When prompted, enter your **app's package name**.
   It's important to enter the fully qualified package name your app is using; this can only be set when you add an app to your Firebase project.
4. At the end, you'll download a `google-services.json` file. You can download this file again at any time.
5. If you haven't done so already, copy the `google-services.json` file into your project's module folder, typically `app/`.

The next step is to Add the SDK to integrate the Firebase libraries in the project.

Add the SDK

To integrate the Firebase libraries into one of your own projects, you need to perform a few basic tasks to prepare your Android Studio project. You may have already done this as part of adding Firebase to your app.

1. Add rules to your root-level `build.gradle` file, to include the **google-services plugin**:

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.1.0'
    }
}
```

Then, in your module Gradle file (usually the `app/build.gradle`), add the apply plugin line at the bottom of the file to enable the Gradle plugin:

```
apply plugin: 'com.android.application'

android {
  // ...
}
```

```
dependencies {
  // ...
  compile 'com.google.firebase:firebase-core:11.0.4'
}

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

The final step is to add the dependencies for the Firebase SDK using one or more **libraries available** for the different Firebase features.

Gradle Dependency Line	Service
com.google.firebase:firebase-core:11.0.4	Analytics
com.google.firebase:firebase-database:11.0.4	Realtime Database
com.google.firebase:firebase-storage:11.0.4	Storage
com.google.firebase:firebase-crash:11.0.4	Crash Reporting
com.google.firebase:firebase-auth:11.0.4	Authentication
com.google.firebase:firebase-messaging:11.0.4	Cloud Messaging / Notifications
com.google.firebase:firebase-config:11.0.4	Remote Config
com.google.firebase:firebase-invites:11.0.4	Invites / Dynamic Links
com.google.firebase:firebase-ads:11.0.4	AdMob
com.google.android.gms:play-services-appindexing:11.0.4	App Indexing

Section 229.2: Updating a Firebase users's email

```
public class ChangeEmailActivity extends BaseAppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {

    @BindView(R.id.et_change_email)
    EditText mEditText;
    private FirebaseUser mFirebaseUser;

    @OnClick(R.id.btn_change_email)
    void onChangeEmailClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter valid email");
            return;
        }

        changeEmail(mEditText.getText().toString());
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mFirebaseAuth.getCurrentUser();
    }
```

```
    private void changeEmail(String email) {
        DialogUtils.showProgressDialog(this, "Changing Email", "Please wait...", false);
        mFirebaseUser.updateEmail(email)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            showToast("Email updated successfully.");
                            return;
                        }

                        if (task.getException() instanceof
FirebaseAuthRecentLoginRequiredException) {
                            FragmentManager fm = getSupportFragmentManager();
                            ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
ReAuthenticateDialogFragment();
                            reAuthenticateDialogFragment.show(fm,
reAuthenticateDialogFragment.getClass().getSimpleName());
                        }
                    }
                });
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_email;
    }

    @Override
    public void onReauthenticateSuccess() {
        changeEmail(mEditText.getText().toString());
    }
}
```

Section 229.3: Create a Firebase user

```
public class SignUpActivity extends BaseAppCompatActivity {

    @BindView(R.id.tIETSignUpEmail)
    EditText mEditEmail;
    @BindView(R.id.tIETSignUpPassword)
    EditText mEditPassword;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }

    @OnClick(R.id.btnSignUpSignUp)
    void signUp() {

        FormValidationUtils.clearErrors(mEditEmail, mEditPassword);

        if (FormValidationUtils.isBlank(mEditEmail)) {
            mEditEmail.setError("Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditEmail)) {
```

```
            mEditEmail.setError("Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditPassword.getText())) {
            mEditPassword.setError("Please enter password");
            return;
        }

        createUserWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
    }

    private void createUserWithEmailAndPassword(String email, String password) {
        DialogUtils.showProgressDialog(this, "", getString(R.string.str_creating_account), false);
        mFirebaseAuth
                .createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (!task.isSuccessful()) {
                            Toast.makeText(SignUpActivity.this, task.getException().getMessage(),
                                    Toast.LENGTH_SHORT).show();
                            DialogUtils.dismissProgressDialog();
                        } else {
                            Toast.makeText(SignUpActivity.this,
R.string.str_registration_successful, Toast.LENGTH_SHORT).show();
                            DialogUtils.dismissProgressDialog();
                            startActivity(new Intent(SignUpActivity.this, HomeActivity.class));
                        }
                    }
                });
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_sign_up;
    }
}
```

Section 229.4: Change Password

```
public class ChangePasswordActivity extends BaseAppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {
    @BindView(R.id.et_change_password)
    EditText mEditText;
    private FirebaseUser mFirebaseUser;

    @OnClick(R.id.btn_change_password)
    void onChangePasswordClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter password");
            return;
        }

        changePassword(mEditText.getText().toString());
    }
```

```
    private void changePassword(String password) {
        DialogUtils.showProgressDialog(this, "Changing Password", "Please wait...", false);
        mFirebaseUser.updatePassword(password)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            showToast("Password updated successfully.");
                            return;
                        }

                        if (task.getException() instanceof
FirebaseAuthRecentLoginRequiredException) {
                            FragmentManager fm = getSupportFragmentManager();
                            ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
ReAuthenticateDialogFragment();
                            reAuthenticateDialogFragment.show(fm,
reAuthenticateDialogFragment.getClass().getSimpleName());
                        }
                    }
                });
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mFirebaseAuth.getCurrentUser();
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_password;
    }

    @Override
    public void onReauthenticateSuccess() {
        changePassword(mEditText.getText().toString());
    }
}
```

Section 229.5: Firebase Cloud Messaging

First of all you need to setup your project adding Firebase to your Android project following the steps described in this topic.

Set up Firebase and the FCM SDK

Add the FCM dependency to your app-level `build.gradle` file

```
dependencies {
 compile 'com.google.firebase:firebase-messaging:11.0.4'
}
```

And at the very bottom (this is important) add:

```
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Edit your app manifest

Add the following to your app's manifest:

- A service that extends `FirebaseMessagingService`. This is required if you want to do any message handling beyond receiving notifications on apps in the background.
- A service that extends `FirebaseInstanceIdService` to handle the creation, rotation, and updating of registration tokens.

For example:

```
    <service
       android:name=".MyInstanceIdListenerService">
        <intent-filter>
            <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
        </intent-filter>
    </service>
    <service
       android:name=".MyFcmListenerService">
        <intent-filter>
            <action android:name="com.google.firebase.MESSAGING_EVENT" />
        </intent-filter>
    </service>
```

Here are simple implementations of the 2 services.

To retrieve the current registration token extend the `FirebaseInstanceIdService` class and override the `onTokenRefresh()` method:

```
public class MyInstanceIdListenerService extends FirebaseInstanceIdService {

    // Called if InstanceID token is updated. Occurs if the security of the previous token had been
    // compromised. This call is initiated by the InstanceID provider.
    @Override
    public void onTokenRefresh() {
        // Get updated InstanceID token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();

        // Send this token to your server or store it locally
    }
}
```

To receive messages, use a service that extends `FirebaseMessagingService` and override the `onMessageReceived` method.

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * Called when message is received.
     *
     * @param remoteMessage Object representing the message received from Firebase Cloud Messaging.
     */
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        String from = remoteMessage.getFrom();

        // Check if message contains a data payload.
        if (remoteMessage.getData().size() > 0) {
```

```
            Log.d(TAG, "Message data payload: " + remoteMessage.getData());
            Map<String, String> data = remoteMessage.getData();
        }

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            Log.d(TAG, "Message Notification Body: " + remoteMessage.getNotification().getBody());
        }

        // do whatever you want with this, post your own notification, or update local state
    }
```

in **Firebase** can grouped user by their behavior like "AppVersion,free user,purchase user,or any specific rules" and then send notification to specific group by send **Topic** Feature in fireBase.
to register user in topic use

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

then in fireBase console, send notification by topic name

More info in the dedicated topic Firebase Cloud Messaging.

Section 229.6: Firebase Storage Operations

With this example, you will be able to perform following operations:

1. Connect to Firebase Storage
2. Create a directory named “images”
3. Upload a file in images directory
4. Download a file from images directory
5. Delete a file from images directory

```
public class MainActivity extends AppCompatActivity {

    private static final int REQUEST_CODE_PICK_IMAGE = 1;
    private static final int PERMISSION_READ_WRITE_EXTERNAL_STORAGE = 2;

    private FirebaseStorage mFirebaseStorage;
    private StorageReference mStorageReference;
    private StorageReference mStorageReferenceImages;
    private Uri mUri;
    private ImageView mImageView;
    private ProgressDialog mProgressDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        mImageView = (ImageView) findViewById(R.id.imageView);
        setSupportActionBar(toolbar);

        // Create an instance of Firebase Storage
        mFirebaseStorage = FirebaseStorage.getInstance();
    }

    private void pickImage() {
        Intent intent = new Intent(Intent.ACTION_PICK,
```

```
android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
        intent.addFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
        startActivityForResult(intent, REQUEST_CODE_PICK_IMAGE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK) {
            if (requestCode == REQUEST_CODE_PICK_IMAGE) {
                String filePath = FileUtil.getPath(this, data.getData());
                mUri = Uri.fromFile(new File(filePath));
                uploadFile(mUri);
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == PERMISSION_READ_WRITE_EXTERNAL_STORAGE) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                pickImage();
            }
        }
    }

    private void showProgressDialog(String title, String message) {
        if (mProgressDialog != null && mProgressDialog.isShowing())
            mProgressDialog.setMessage(message);
        else
            mProgressDialog = ProgressDialog.show(this, title, message, true, false);
    }

    private void hideProgressDialog() {
        if (mProgressDialog != null && mProgressDialog.isShowing()) {
            mProgressDialog.dismiss();
        }
    }

    private void showToast(String message) {
        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }

    public void showHorizontalProgressDialog(String title, String body) {

        if (mProgressDialog != null && mProgressDialog.isShowing()) {
            mProgressDialog.setTitle(title);
            mProgressDialog.setMessage(body);
        } else {
            mProgressDialog = new ProgressDialog(this);
            mProgressDialog.setTitle(title);
            mProgressDialog.setMessage(body);
            mProgressDialog.setIndeterminate(false);
            mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
            mProgressDialog.setProgress(0);
            mProgressDialog.setMax(100);
            mProgressDialog.setCancelable(false);
            mProgressDialog.show();
        }
    }
```

```
    public void updateProgress(int progress) {
        if (mProgressDialog != null && mProgressDialog.isShowing()) {
            mProgressDialog.setProgress(progress);
        }
    }

    /**
     * Step 1: Create a Storage
     *
     * @param view
     */
    public void onCreateReferenceClick(View view) {
        mStorageReference = mFirebaseStorage.getReferenceFromUrl("gs://**something**.appspot.com");
        showToast("Reference Created Successfully.");
        findViewById(R.id.button_step_2).setEnabled(true);
    }

    /**
     * Step 2: Create a directory named "Images"
     *
     * @param view
     */
    public void onCreateDirectoryClick(View view) {
        mStorageReferenceImages = mStorageReference.child("images");
        showToast("Directory 'images' created Successfully.");
        findViewById(R.id.button_step_3).setEnabled(true);
    }

    /**
     * Step 3: Upload an Image File and display it on ImageView
     *
     * @param view
     */
    public void onUploadFileClick(View view) {
        if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED ||
ActivityCompat.checkSelfPermission(MainActivity.this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
!= PackageManager.PERMISSION_GRANTED)
            ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE, Manifest.permission.WRITE_EXTERNAL_STORAGE},
PERMISSION_READ_WRITE_EXTERNAL_STORAGE);
        else {
            pickImage();
        }
    }

    /**
     * Step 4: Download an Image File and display it on ImageView
     *
     * @param view
     */
    public void onDownloadFileClick(View view) {
        downloadFile(mUri);
    }

    /**
     * Step 5: Delete am Image File and remove Image from ImageView
     *
     * @param view
     */
    public void onDeleteFileClick(View view) {
        deleteFile(mUri);
```

```
    }

    private void showAlertDialog(Context ctx, String title, String body,
DialogInterface.OnClickListener okListener) {

        if (okListener == null) {
            okListener = new DialogInterface.OnClickListener() {

                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            };
        }

        AlertDialog.Builder builder = new
AlertDialog.Builder(ctx).setMessage(body).setPositiveButton("OK", okListener).setCancelable(false);

        if (!TextUtils.isEmpty(title)) {
            builder.setTitle(title);
        }

        builder.show();
    }

    private void uploadFile(Uri uri) {
        mImageView.setImageResource(R.drawable.placeholder_image);

        StorageReference uploadStorageReference =
mStorageReferenceImages.child(uri.getLastPathSegment());
        final UploadTask uploadTask = uploadStorageReference.putFile(uri);
        showHorizontalProgressDialog("Uploading", "Please wait...");
        uploadTask
                .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                        hideProgressDialog();
                        Uri downloadUrl = taskSnapshot.getDownloadUrl();
                        Log.d("MainActivity", downloadUrl.toString());
                        showAlertDialog(MainActivity.this, "Upload Complete",
downloadUrl.toString(), new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {
                                findViewById(R.id.button_step_3).setEnabled(false);
                                findViewById(R.id.button_step_4).setEnabled(true);
                            }
                        });

                        Glide.with(MainActivity.this)
                                .load(downloadUrl)
                                .into(mImageView);
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception exception) {
                        exception.printStackTrace();
                        // Handle unsuccessful uploads
                        hideProgressDialog();
                    }
                })
                .addOnProgressListener(MainActivity.this, new
OnProgressListener<UploadTask.TaskSnapshot>() {
```

```
                    @Override
                    public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
                        int progress = (int) (100 * (float) taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
                        Log.i("Progress", progress + "");
                        updateProgress(progress);
                    }
                });
    }

    private void downloadFile(Uri uri) {
        mImageView.setImageResource(R.drawable.placeholder_image);
        final StorageReference storageReferenceImage =
mStorageReferenceImages.child(uri.getLastPathSegment());
        File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
                Environment.DIRECTORY_PICTURES), "Firebase Storage");
        if (!mediaStorageDir.exists()) {
            if (!mediaStorageDir.mkdirs()) {
                Log.d("MainActivity", "failed to create Firebase Storage directory");
            }
        }

        final File localFile = new File(mediaStorageDir, uri.getLastPathSegment());
        try {
            localFile.createNewFile();
        } catch (IOException e) {
            e.printStackTrace();
        }

        showHorizontalProgressDialog("Downloading", "Please wait...");
        storageReferenceImage.getFile(localFile).addOnSuccessListener(new
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
                hideProgressDialog();
                showAlertDialog(MainActivity.this, "Download Complete",
localFile.getAbsolutePath(), new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        findViewById(R.id.button_step_4).setEnabled(false);
                        findViewById(R.id.button_step_5).setEnabled(true);
                    }
                });

                Glide.with(MainActivity.this)
                        .load(localFile)
                        .into(mImageView);
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                // Handle any errors
                hideProgressDialog();
                exception.printStackTrace();
            }
        }).addOnProgressListener(new OnProgressListener<FileDownloadTask.TaskSnapshot>() {
            @Override
            public void onProgress(FileDownloadTask.TaskSnapshot taskSnapshot) {
                int progress = (int) (100 * (float) taskSnapshot.getBytesTransferred() /
taskSnapshot.getTotalByteCount());
                Log.i("Progress", progress + "");
                updateProgress(progress);
```

```
            }
        });
    }

    private void deleteFile(Uri uri) {
        showProgressDialog("Deleting", "Please wait...");
        StorageReference storageReferenceImage =
mStorageReferenceImages.child(uri.getLastPathSegment());
        storageReferenceImage.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                hideProgressDialog();
                showAlertDialog(MainActivity.this, "Success", "File deleted successfully.", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        mImageView.setImageResource(R.drawable.placeholder_image);
                        findViewById(R.id.button_step_3).setEnabled(true);
                        findViewById(R.id.button_step_4).setEnabled(false);
                        findViewById(R.id.button_step_5).setEnabled(false);
                    }
                });
                File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
                        Environment.DIRECTORY_PICTURES), "Firebase Storage");
                if (!mediaStorageDir.exists()) {
                    if (!mediaStorageDir.mkdirs()) {
                        Log.d("MainActivity", "failed to create Firebase Storage directory");
                    }
                }
                deleteFiles(mediaStorageDir);
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                hideProgressDialog();
                exception.printStackTrace();
            }
        });
    }

    private void deleteFiles(File directory) {
        if (directory.isDirectory())
            for (File child : directory.listFiles())
                child.delete();
    }
}
```

By default, Firebase Storage rules applies Authentication restriction. If user is authenticated, only then, he can perform operations on Firebase Storage, else he cannot. I have disabled the authentication part in this demo by updating Storage rules. Previously, rules were looking like:

```
service firebase.storage {
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

But I changed to skip the authentication:

```
service firebase.storage {
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```

Section 229.7: Firebase Realtime Database: how to set/get data

Note: Let's setup some anonymous authentication for the example

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Once it is done, create a child by editing your database address. For example:

https://your-project.firebaseio.com/ to **https://your-project.firebaseio.com/chat**

We will put data to this location from our Android device. You **don't have to** create the database structure (tabs, fields... etc), it will be automatically created when you'll send Java object to Firebase!

Create a Java object that contains all the attributes you want to send to the database:

```
public class ChatMessage {
    private String username;
    private String message;

    public ChatMessage(String username, String message) {
        this.username = username;
        this.message = message;
    }

    public ChatMessage() {} // you MUST have an empty constructor

    public String getUsername() {
        return username;
    }

    public String getMessage() {
        return message;
    }
}
```

Then in your activity:

```
if (FirebaseAuth.getInstance().getCurrentUser() == null) {
        FirebaseAuth.getInstance().signInAnonymously().addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isComplete() && task.isSuccessful()){
                    FirebaseDatabase database = FirebaseDatabase.getInstance();
```

```
                    DatabaseReference reference = database.getReference("chat"); // reference is
'chat' because we created the database at /chat
                }
            }
        });
}
```

To send a value:

```
ChatMessage msg = new ChatMessage("user1", "Hello World!");
reference.push().setValue(msg);
```

To receive changes that occurs in the database:

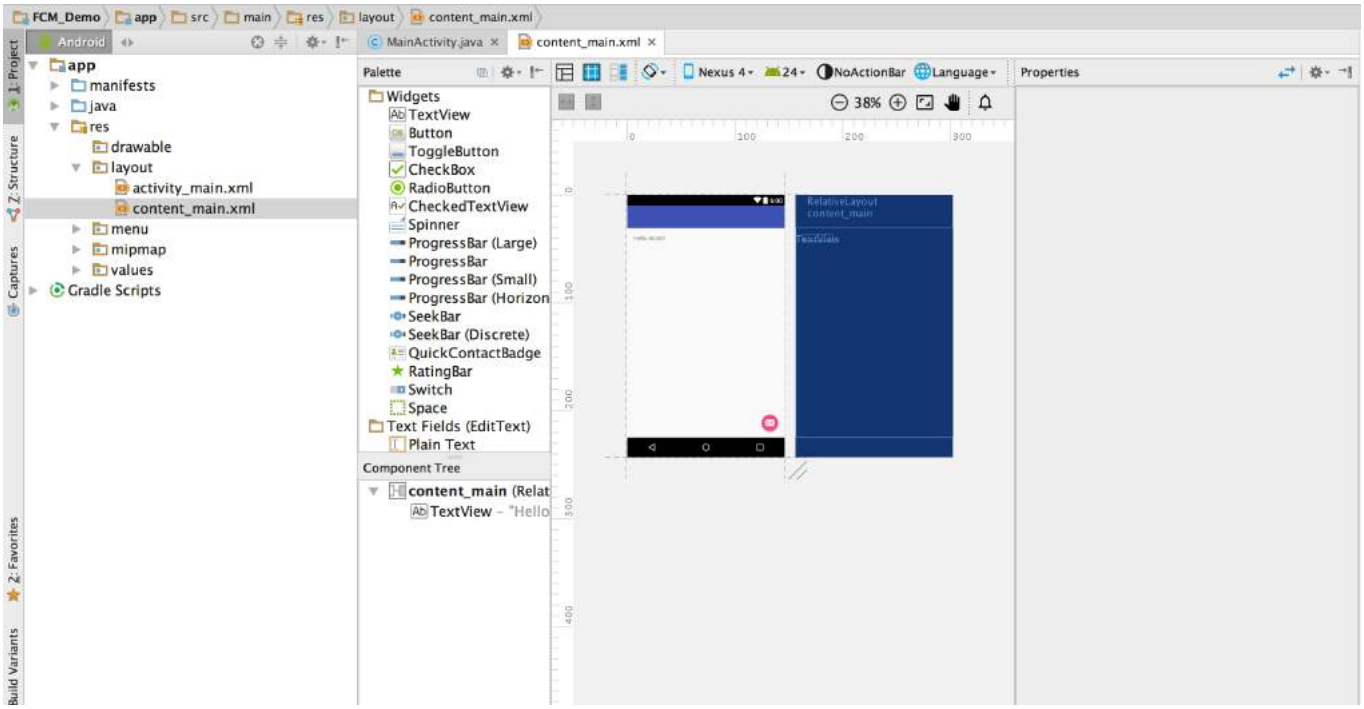
```
reference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        ChatMessage msg = dataSnapshot.getValue(ChatMessage.class);
        Log.d(TAG, msg.getUsername()+" "+msg.getMessage());
    }

    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}
    public void onChildRemoved(DataSnapshot dataSnapshot) {}
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}
    public void onCancelled(DatabaseError databaseError) {}
});
```

Section 229.8: Demo of FCM based notifications

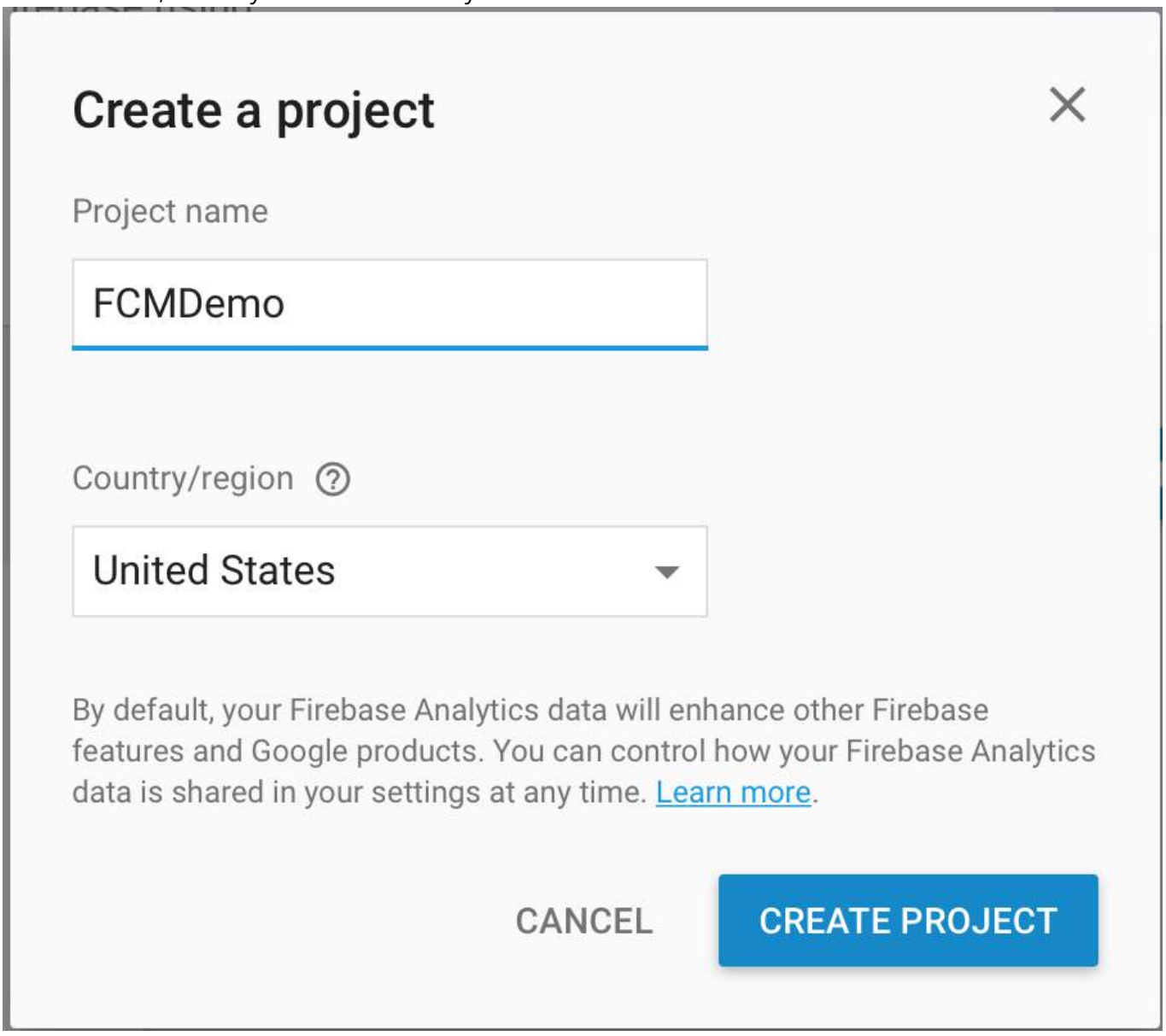
This example shows how to use the Firebase Cloud Messaging(FCM) platform. FCM is a successor of Google Cloud Messaging(GCM). It does not require C2D_MESSAGE permissions from the app users.

Steps to integrate FCM are as follows.

1. Create sample hello world project in Android Studio Your Android studio screen would look like the following picture.



2. Next step is to set up firebase project. Visit https://console.firebase.google.com and create a project with an identical name, so that you can track it easily.



3. Now it is time to add firebase to your sample android project you have just created. You will need package name of your project and Debug signing certificate SHA-1(optional).

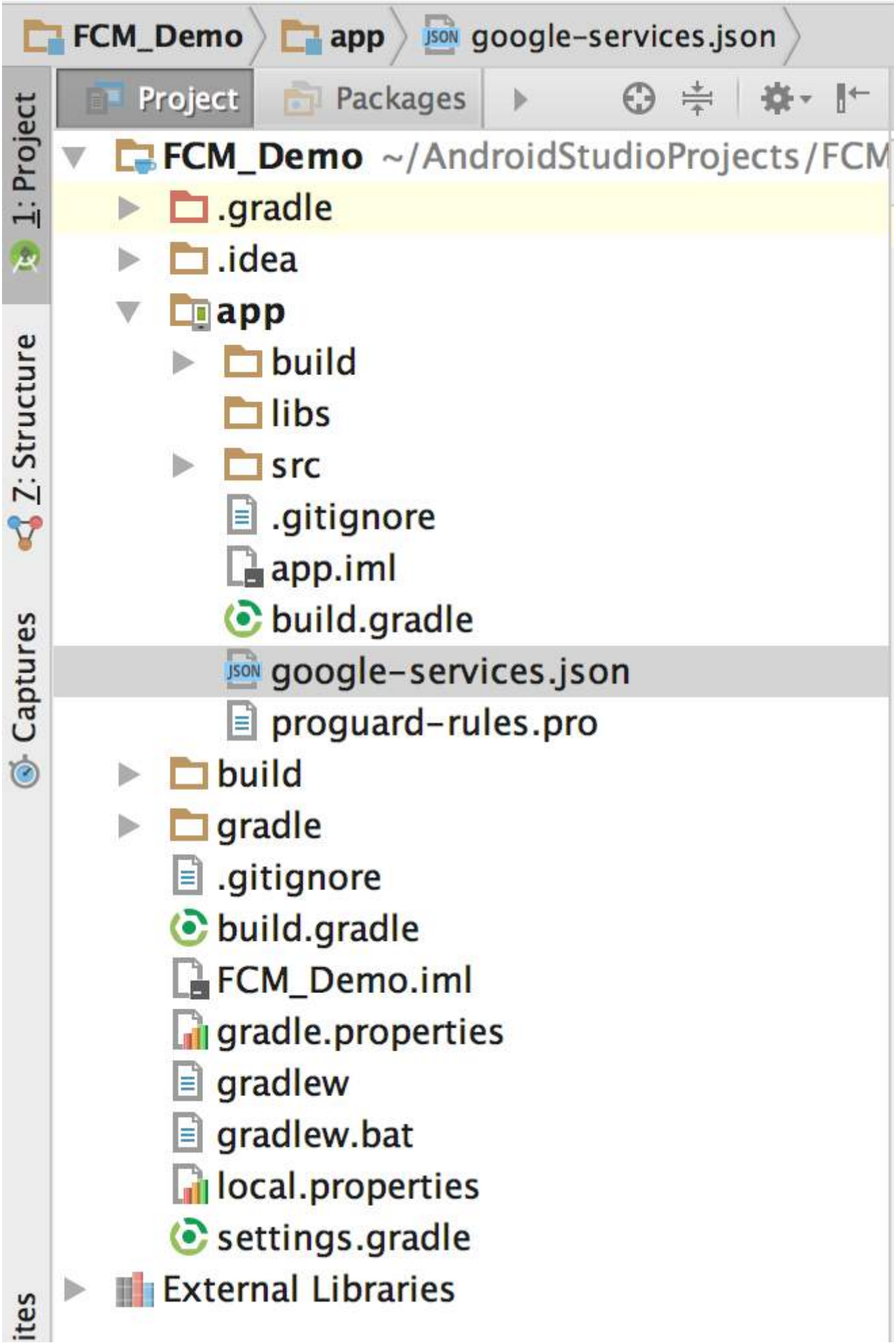
   a. Package name - It can be found from the android manifest XML file.

   b. Debug signing SHA-1 certificate - It can be found by running following command in the terminal.

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -
keypass android
```

Enter this information in the firebase console and add the app to firebase project. Once you click on add app button, your browser would automatically download a JSON file named "google-services.json".

4. Now copy the google-services.json file you have just downloaded into your Android app module root directory.



5. Follow the instructions given on the firebase console as you proceed ahead. a. Add following code line to

your project level build.gradle

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' .....
```

b. Add following code line at the end of your app level build.gradle.

```
    //following are the dependencies to be added
    compile 'com.google.firebase:firebase-messaging:11.0.4'
    compile 'com.android.support:multidex:1.0.1'
}
// this line goes to the end of the file
apply plugin: 'com.google.gms.google-services'
```

c. Android studio would ask you to sync project. Click on Sync now.

6. Next task is to add two services. a. One extending FirebaseMessagingService with intent-filter as following

```
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
```

b. One extending FirebaseInstanceIDService.

```
<intent-filter>
    <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
</intent-filter>
```

7. FirebaseMessagingService code should look like this.

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.google.firebase.messaging.FirebaseMessagingService;

public class MyFirebaseMessagingService extends FirebaseMessagingService {
    public MyFirebaseMessagingService() {
    }
}
```

8. FirebaseInstanceIdService should look like this.

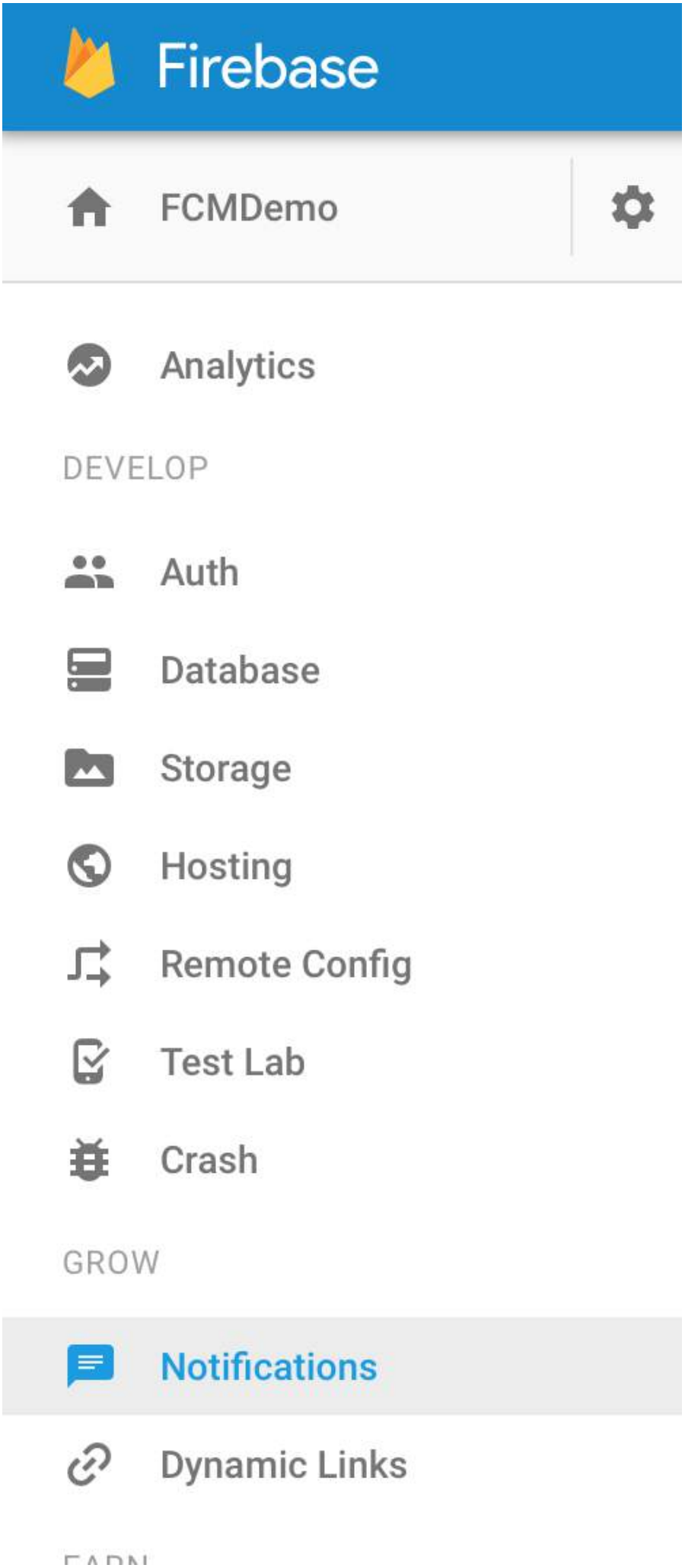
```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.google.firebase.iid.FirebaseInstanceIdService;

public class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {
    public MyFirebaseInstanceIDService() {
    }
}
```

9. Now it is time to capture the device registration token. Add following line of code to MainActivity's onCreate method.

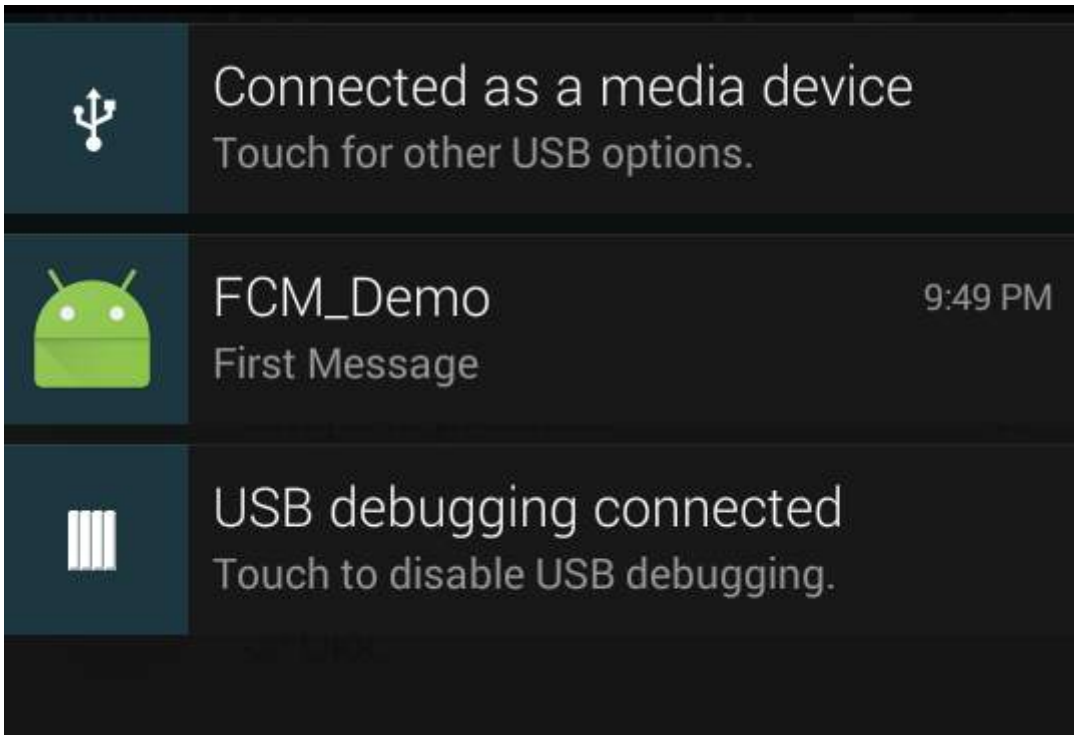
```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. Once we have the access token, we can use firebase console to send out the notification. Run the app on

your android handset.



Click on Notification in Firebase console and UI will help you to send out your first message. Firebase offers functionality to send messages to single device(By using the device token id we captured) or all the users using our app or to specific group of users. Once you send your first message, your mobile screen should look like following.



Thank you

Section 229.9: Sign In Firebase user with email and password

```
public class LoginActivity extends BaseAppCompatActivity {

    @BindView(R.id.tIETLoginEmail)
    EditText mEditEmail;
    @BindView(R.id.tIETLoginPassword)
    EditText mEditPassword;

    @Override
    protected void onResume() {
        super.onResume();
        FirebaseUser firebaseUser = mFirebaseAuth.getCurrentUser();
        if (firebaseUser != null)
            startActivity(new Intent(this, HomeActivity.class));
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_login;
    }

    @OnClick(R.id.btnLoginLogin)
    void onSignInClick() {

        FormValidationUtils.clearErrors(mEditEmail, mEditPassword);

        if (FormValidationUtils.isBlank(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter email");
            return;
        }
```

```
        if (!FormValidationUtils.isEmailValid(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditPassword.getText())) {
            FormValidationUtils.setError(null, mEditPassword, "Please enter password");
            return;
        }

        signInWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
    }

    private void signInWithEmailAndPassword(String email, String password) {
        DialogUtils.showProgressDialog(this, "", getString(R.string.sign_in), false);
        mFirebaseAuth
                .signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {

                        DialogUtils.dismissProgressDialog();

                        if (task.isSuccessful()) {
                            Toast.makeText(LoginActivity.this, "Login Successful",
Toast.LENGTH_SHORT).show();
                            startActivity(new Intent(LoginActivity.this, HomeActivity.class));
                            finish();
                        } else {
                            Toast.makeText(LoginActivity.this, task.getException().getMessage(),
                                    Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }

    @OnClick(R.id.btnLoginSignUp)
    void onSignUpClick() {
        startActivity(new Intent(this, SignUpActivity.class));
    }


    @OnClick(R.id.btnLoginForgotPassword)
    void forgotPassword() {
        startActivity(new Intent(this, ForgotPasswordActivity.class));
    }
}
```

Section 229.10: Send Firebase password reset email

```
public class ForgotPasswordActivity extends AppCompatActivity {

    @BindView(R.id.tIETForgotPasswordEmail)
    EditText mEditEmail;
    private FirebaseAuth mFirebaseAuth;
    private FirebaseAuth.AuthStateListener mAuthStateListener;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_forgot_password);
        ButterKnife.bind(this);

        mFirebaseAuth = FirebaseAuth.getInstance();

        mAuthStateListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
                FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();
                if (firebaseUser != null) {
                    // Do whatever you want with the UserId by firebaseUser.getUid()
                } else {

                }
            }
        };
    }

    @Override
    protected void onStart() {
        super.onStart();
        mFirebaseAuth.addAuthStateListener(mAuthStateListener);
    }

    @Override
    protected void onStop() {
        super.onStop();
        if (mAuthStateListener != null) {
            mFirebaseAuth.removeAuthStateListener(mAuthStateListener);
        }
    }

    @OnClick(R.id.btnForgotPasswordSubmit)
    void onSubmitClick() {

        if (FormValidationUtils.isBlank(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter valid email");
            return;
        }

        DialogUtils.showProgressDialog(this, "", "Please wait...", false);
        mFirebaseAuth.sendPasswordResetEmail(mEditEmail.getText().toString())
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            Toast.makeText(ForgotPasswordActivity.this, "An email has been sent to
you.", Toast.LENGTH_SHORT).show();
                            finish();
                        } else {
                            Toast.makeText(ForgotPasswordActivity.this,
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }
```

```
}
```

Section 229.11: Re-Authenticate Firebase user

```
public class ReAuthenticateDialogFragment extends DialogFragment {

    @BindView(R.id.et_dialog_reauthenticate_email)
    EditText mEditTextEmail;
    @BindView(R.id.et_dialog_reauthenticate_password)
    EditText mEditTextPassword;
    private OnReauthenticateSuccessListener mOnReauthenticateSuccessListener;

    @OnClick(R.id.btn_dialog_reauthenticate)
    void onReauthenticateClick() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditTextPassword.getText())) {
            FormValidationUtils.setError(null, mEditTextPassword, "Please enter password");
            return;
        }

        reauthenticateUser(mEditTextEmail.getText().toString(),
mEditTextPassword.getText().toString());
    }

    private void reauthenticateUser(String email, String password) {
        DialogUtils.showProgressDialog(getActivity(), "Re-Authenticating", "Please wait...",
false);
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        AuthCredential authCredential = EmailAuthProvider.getCredential(email, password);
        firebaseUser.reauthenticate(authCredential)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            mOnReauthenticateSuccessListener.onReauthenticateSuccess();
                            dismiss();
                        } else {
                            ((BaseAppCompatActivity)
getActivity()).showToast(task.getException().getMessage());
                        }
                    }
                });
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        mOnReauthenticateSuccessListener = (OnReauthenticateSuccessListener) context;
```

```
    }

    @OnClick(R.id.btn_dialog_reauthenticate_cancel)
    void onCancelClick() {
        dismiss();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.dialog_reauthenticate, container);
        ButterKnife.bind(this, view);
        return view;
    }

    @Override
    public void onResume() {
        super.onResume();
        Window window = getDialog().getWindow();
        window.setLayout(WindowManager.LayoutParams.MATCH_PARENT,
WindowManager.LayoutParams.WRAP_CONTENT);
    }

    interface OnReauthenticateSuccessListener {
        void onReauthenticateSuccess();
    }
}
```

Section 229.12: Firebase Sign Out

Initialization of variable

```
private GoogleApiClient mGoogleApiClient;
```

You must have to Write this Code in onCreate() method of all that when u put signout button.

```
 mGoogleApiClient = new GoogleApiClient.Builder(this)
            .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
            .addApi(Auth.GOOGLE_SIGN_IN_API)
            .build();
```

Put below code on signout button.

```
Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
                    new ResultCallback<Status>() {
                        @Override
                        public void onResult(Status status) {
                            FirebaseAuth.getInstance().signOut();
                            Intent i1 = new Intent(MainActivity.this, GoogleSignInActivity.class);
                            startActivity(i1);
                            Toast.makeText(MainActivity.this, "Logout Successfully!",
Toast.LENGTH_SHORT).show();
                        }
                    });
```

Chapter 230: Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost.

Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user reengagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to `4KB` to a client app.

Section 230.1: Set Up a Firebase Cloud Messaging Client App on Android

1. Complete the Installation and setup part to connect your app to Firebase.
   This will create the project in Firebase.
2. Add the dependency for Firebase Cloud Messaging to your module-level `build.gradle` file:

```
dependencies {
     compile 'com.google.firebase:firebase-messaging:10.2.1'
}
```

Now you are ready to work with the FCM in Android.

FCM clients require devices running `Android 2.3` or higher that also have the Google Play Store app installed, or an emulator running Android 2.3 with Google APIs.

Edit your `AndroidManifest.xml` file

```
<service
   android:name=".MyFirebaseMessagingService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>

<service
   android:name=".MyFirebaseInstanceIDService">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>
```

Section 230.2: Receive Messages

To receive messages, use a service that extends `FirebaseMessagingService` and override the `onMessageReceived` method.

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * Called when message is received.
     *
     * @param remoteMessage Object representing the message received from Firebase Cloud Messaging.
     */
    @Override
    public void onMessageReceived(RemoteMessage message) {
```

```
        String from = message.getFrom();

        // Check if message contains a data payload.
        if (remoteMessage.getData().size() > 0) {
            Log.d(TAG, "Message data payload: " + remoteMessage.getData());
            Map<String, String> data = message.getData();
        }

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            Log.d(TAG, "Message Notification Body: " + remoteMessage.getNotification().getBody());
        }

        //.....
    }
```

When the app is in the background, Android directs notification messages to the system tray. A user tap on the notification opens the app launcher by default.

This includes messages that contain both notification and data payload (and all messages sent from the Notifications console). In these cases, the notification is delivered to the device's system tray, and the data payload is delivered in the extras of the intent of your launcher Activity.

Here a short recap:

App state	Notification	Data	Both
Foreground	`onMessageReceived`	`onMessageReceived`	`onMessageReceived`
Background	System tray	`onMessageReceived`	Notification: system tray Data: in extras of the intent.

Section 230.3: This code that i have implemnted in my app for pushing image,message and also link for opening in your webView

This is my FirebaseMessagingService

```
    public class MyFirebaseMessagingService extends FirebaseMessagingService {
Bitmap bitmap;
@Override
public void onMessageReceived(RemoteMessage remoteMessage) {
    String message = remoteMessage.getData().get("message");
    //imageUri will contain URL of the image to be displayed with Notification
    String imageUri = remoteMessage.getData().get("image");
    String link=remoteMessage.getData().get("link");

    //To get a Bitmap image from the URL received
    bitmap = getBitmapfromUrl(imageUri);
    sendNotification(message, bitmap,link);

}


/**
 * Create and show a simple notification containing the received FCM message.
 */

private void sendNotification(String messageBody, Bitmap image, String link) {
    Intent intent = new Intent(this, NewsListActivity.class);
```

```
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    intent.putExtra("LINK",link);
    PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */, intent,
            PendingIntent.FLAG_ONE_SHOT);
    Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
            .setLargeIcon(image)/*Notification icon image*/
            .setSmallIcon(R.drawable.hindi)
            .setContentTitle(messageBody)
            .setStyle(new NotificationCompat.BigPictureStyle()
                    .bigPicture(image))/*Notification with Image*/
            .setAutoCancel(true)
            .setSound(defaultSoundUri)
            .setContentIntent(pendingIntent);
    NotificationManager notificationManager =
            (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

    notificationManager.notify(0 /* ID of notification */, notificationBuilder.build());
}
public Bitmap getBitmapfromUrl(String imageUrl) {
    try {
        URL url = new URL(imageUrl);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setDoInput(true);
        connection.connect();
        InputStream input = connection.getInputStream();
        Bitmap bitmap = BitmapFactory.decodeStream(input);
        return bitmap;

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        return null;

    }
}}
```

And this is MainActivity to open link in my WebView or other browser depand on your requirement through intents.

```
if (getIntent().getExtras() != null) {
    if (getIntent().getStringExtra("LINK")!=null) {
        Intent i=new Intent(this,BrowserActivity.class);
        i.putExtra("link",getIntent().getStringExtra("LINK"));
        i.putExtra("PUSH","yes");
        NewsListActivity.this.startActivity(i);
        finish();
    }}
```

Section 230.4: Registration token

On initial startup of your app, the FCM SDK generates a registration token for the client app instance.
If you want to target single devices or create device groups, you'll need to access this token by extending `FirebaseInstanceIdService`.

The `onTokenRefresh` callback fires whenever a new token is generated and you can use the method `FirebaseInstanceID.getToken()` to retrieve the current token.

Example:

```
public class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {

   /**
     * Called if InstanceID token is updated. This may occur if the security of
     * the previous token had been compromised. Note that this is called when the InstanceID token
     * is initially generated so this is where you would retrieve the token.
     */

    @Override
    public void onTokenRefresh() {
        // Get updated InstanceID token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();
        Log.d(TAG, "Refreshed token: " + refreshedToken);

    }

}
```

Section 230.5: Subscribe to a topic

Client apps can subscribe to any existing topic, or they can create a new topic. When a client app subscribes to a new topic name, a new topic of that name is created in FCM and any client can subsequently subscribe to it.

To subscribe to a topic use the `subscribeToTopic()` method specifying the topic name:

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

Chapter 231: Firebase Realtime DataBase

Section 231.1: Quick setup

1. Complete the Installation and setup part to connect your app to Firebase.
   This will create the project in Firebase.
2. Add the dependency for Firebase Realtime Database to your module-level `build.gradle` file:

```
compile 'com.google.firebase:firebase-database:10.2.1'
```

3. Configure Firebase Database Rules

Now you are ready to work with the Realtime Database in Android.

For example you write a `Hello World` message to the database under the `message` key.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Section 231.2: Firebase Realtime DataBase event handler

First Initialize FirebaseDatabase:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Write to your database:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Read from your database:

```
// Read from the database
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        String value = dataSnapshot.getValue(String.class);
        Log.d(TAG, "Value is: " + value);
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException());
    }
});
```

Retrieve Data on Android events:

```
ChildEventListener childEventListener = new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildAdded:" + dataSnapshot.getKey());
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildChanged:" + dataSnapshot.getKey());
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
        Log.d(TAG, "onChildRemoved:" + dataSnapshot.getKey());

    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildMoved:" + dataSnapshot.getKey());

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.w(TAG, "postComments:onCancelled", databaseError.toException());
        Toast.makeText(mContext, "Failed to load comments.",
                Toast.LENGTH_SHORT).show();
    }
};
ref.addChildEventListener(childEventListener);
```

Section 231.3: Understanding firebase JSON database

Before we get our hands dirty with code, I feel it is necessary to understand how data is stored in firebase. Unlike relational databases, firebase stores data in JSON format. Think of each row in a relational database as a JSON object (which is basically unordered key-value pair). So the column name becomes key and the value stored in that column for one particular row is the value. This way the entire row is represented as a JSON object and a list of these represent an entire database table. The immediate benefit that I see for this is schema modification becomes much more cheaper operation compared to old RDBMS. It is easier to add a couple of more attributes to a JSON than altering a table structure.

here is a sample JSON to show how data is stored in firebase:

```
{
  "user_base" : {
    "342343" : {
      "email" : "kaushal.xxxxx@gmail.com",
      "authToken" : "some string",
      "name" : "Kaushal",
      "phone" : "+919916xxxxxx",
      "serviceProviderId" : "firebase",
      "signInServiceType" : "google",
    },
    "354895" : {
      "email" : "xxxxx.devil@gmail.com",
      "authToken" : "some string",
```

```
        "name" : "devil",
        "phone" : "+919685xxxxxx",
        "serviceProviderId" : "firebase",
        "signInServiceType" : "github"
      },
      "371298" : {
        "email" : "bruce.wayne@wayneinc.com",
        "authToken" : "I am batman",
        "name" : "Bruce Wayne",
        "phone" : "+14085xxxxxx",
        "serviceProviderId" : "firebase",
        "signInServiceType" : "shield"
      }
    },
    "user_prefs": {
      "key1":{
        "data": "for key one"
      },
      "key2":{
        "data": "for key two"
      },
      "key3":{
        "data": "for key three"
      }
    },
    //other structures
  }
```

This clearly shows how data that we used to store in relational databases can be stored in JSON format. Next let's see how to read this data in android devices.

Section 231.4: Retrieving data from firebase

I am gonna assume you already know about adding gradle dependencies firebase in android studio. If you don't just follow the guide from here. Add your app in firebase console, gradle sync android studio after adding dependencies. All dependencies are not needed just firebase database and firebase auth.

Now that we know how data is stored and how to add gradle dependencies let's see how to use the imported firebase android SDK to retrieve data.

create a firebase database reference

```
DatabaseReference userDBRef = FirebaseDatabase.getInstance().getReference();
// above statement point to base tree
userDBRef = DatabaseReference.getInstance().getReference().child("user_base")
// points to user_base table JSON (see previous section)
```

from here you can chain multiple child() method calls to point to the data you are interested in. For example if data is stored as depicted in previous section and you want to point to Bruce Wayne user you can use:

```
DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 is key of bruce wayne user in JSON structure (previous section)
```

Or simply pass the whole reference to the JSON object:

```
DatabaseReference bruceWayneRef = DatabaseReference.getInstance().getReference()
     .child("user_base/371298");
// deeply nested data can also be referenced this way, just put the fully
```

```
// qualified path in pattern shown in above code "blah/blah1/blah1-2/blah1-2-3..."
```

Now that we have the reference of the data we want to fetch, we can use listeners to fetch data in android apps. Unlike the traditional calls where you fire REST API calls using retrofit or volley, here a simple callback listener is required to get the data. Firebase sdk calls the callback methods and you are done.

There are basically two types of listeners you can attach, one is ValueEventListener and the other one is ChildEventListener (described in next section). For any change in data under the node we have references and added listeners to, value event listeners return the entire JSON structure and child event listener returns specific child where the change has happened. Both of these are useful in their own way. To fetch the data from firebase we can add one or more listeners to a firebase database reference (list userDBRef we created earlier).

Here is some sample code (code explanation after code):

```
userDBRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        User bruceWayne = dataSnapshot.child("371298").getValue(User.class);
        // Do something with the retrieved data or Bruce Wayne
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.e("UserListActivity", "Error occurred");
        // Do something about the error
    });
```

Did you notice the Class type passed. DataSnapshot can convert JSON data into our defined POJOs, simple pass the right class type.

If your use case does not require the entire data (in our case user_base table) every time some little change occurs or say you want to ***fetch the data only once***, you can use ***addListenerForSingleValueEvent()*** method of Database reference. This fires the callback only once.

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Above samples will give you the value of the JSON node. To get the key simply call:

```
String myKey = dataSnapshot.getKey();
```

Section 231.5: Listening for child updates

Take a use case, like a chat app or a collaborative grocery list app (that basically requires a list of objects to be synced across users). If you use firebase database and add a value event listener to the chat parent node or grocery list parent node, you will end with entire chat structure from the beginning of time (i meant beginning of your chat) every time a chat node is added (i.e. anyone says hi). That we don't want to do, what we are interested in is only the new node or only the old node that got deleted or modified, the unchanged ones should not be returned.

In this case we can use ChildEvenListener. Without any further adieu, here is code sample (see prev sections for sample JSON data):

```
userDBRef.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        //If not dealing with ordered data forget about this
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
    });
```

Method names are self explanatory. As you can see whenever a new user is added or some property of existing user is modified or user is deleted or removed appropriate callback method of child event listener is called with relevant data. So if you are keeping UI refreshed for say chat app, get the JSON from onChildAdded() parse into POJO and fit it in your UI. Just remember to remove your listener when user leaves the screen.

onChildChanged() gives the entire child value with changed properties (new ones).

onChiledRemoved() returns the removed child node.

Section 231.6: Retrieving data with pagination

When you have a huge JSON database, adding a value event listener doesn't make sense. It will return the huge JSON and parsing it would be time consuming. In such cases we can use pagination and fetch part of data and display or process it. Kind of like lazy loading or like fetching old chats when user clicks on show older chat. In this case Query can used.

Let's take the our old example in previous sections. The user base contains 3 users, if it grows to say 3 hundred thousand user and you want to fetch the user list in batches of 50:

```
// class level
final int limit = 50;
int start = 0;

// event level
Query userListQuery = userDBRef.orderByChild("email").limitToFirst(limit)
        .startAt(start)
userListQuery.addValueEventListener(new ValueEventListener() {
    @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
        start += (limit+1);
    }
```

```
    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Here value or child events can be added and listened to. Call query again to fetch next 50. **Make sure to add the orderByChild() method**, this will not work without that. Firebase needs to know the order by which you are paginating.

Section 231.7: Denormalization: Flat Database Structure

Denormalization and a flat database structure is neccessary to efficiently download separate calls. With the following structure, it is also possible to maintain two-way relationships. The disadvantage of this approach is, that you always need to update the data in multiple places.

For an example, imagine an app which allows the user to store messages to himself (memos).

Desired flat database structure:

```
|--database
  |-- memos
     |-- memokey1
       |-- title: "Title"
       |-- content: "Message"
     |-- memokey2
       |-- title: "Important Title"
       |-- content: "Important Message"
  |-- users
     |-- userKey1
       |-- name: "John Doe"
       |-- memos
         |-- memokey1 : true //The values here don't matter, we only need the keys.
         |-- memokey2 : true
     |-- userKey2
       |-- name: "Max Doe"
```

The used memo class

```
public class Memo {
    private String title, content;
    //getters and setters ...

    //toMap() is necessary for the push process
    private Map<String, Object> toMap() {
        HashMap<String, Object> result = new HashMap<>();
        result.put("title", title);
        result.put("content", content);
        return result;
    }
}
```

Retrieving the memos of a user

```
//We need to store the keys and the memos separately
private ArrayList<String> mKeys = new ArrayList<>();
private ArrayList<Memo> mMemos = new ArrayList<>();

//The user needs to be logged in to retrieve the uid
```

```
String currentUserId = FirebaseAuth.getInstance().getCurrentUser().getUid();

//This is the reference to the list of memos a user has
DatabaseReference currentUserMemoReference = FirebaseDatabase.getInstance().getReference()
    .child("users").child(currentUserId).child("memos");

//This is a reference to the list of all memos
DatabaseReference memoReference = FirebaseDatabase.getInstance().getReference()
    .child("memos");

//We start to listen to the users memos,
//this will also retrieve the memos initially
currentUserMemoReference.addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String s) {
            //Here we retrieve the key of the memo the user has.
            String key = dataSnapshot.getKey(); //for example memokey1
            //For later manipulations of the lists, we need to store the key in a list
            mKeys.add(key);
            //Now that we know which message belongs to the user,
            //we request it from our memos:
            memoReference.child(key).addValueEventListener(new ValueEventListener() {
                @Override
                    public void onDataChange(DataSnapshot dataSnapshot) {
                        //Here we retrieve our memo:
                        Memo memo = dataSnapshot.getValue(Memo.class);
                        mMemos.add(memo);
                    }

                @Override
                public void onCancelled(DatabaseError databaseError) { }
            });
        }

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String s) { }

        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) { }

        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String s) { }

        @Override
        public void onCancelled(DatabaseError databaseError) { }
    }
```

Creating a memo

```
//The user needs to be logged in to retrieve the uid
String currentUserUid = FirebaseAuth.getInstance().getCurrentUser().getUid();

//This is the path to the list of memos a user has
String userMemoPath = "users/" + currentUserUid + "/memos/";

//This is the path to the list of all memos
String memoPath = "memos/";

//We need to retrieve an unused key from the memos reference
DatabaseReference memoReference = FirebaseDatabase.getInstance().getReference().child("memos");
String key = memoReference.push().getKey();
```

```
Memo newMemo = new Memo("Important numbers", "1337, 42, 3.14159265359");

Map<String, Object> childUpdates = new HashMap<>();
//The second parameter **here** (the value) does not matter, it's just that the key exists
childUpdates.put(userMemoPath + key, true);
childUpdates.put(memoPath + key, newMemo.toMap());

FirebaseDatabase.getInstance().getReference().updateChildren(childUpdates);
```

After the push, or database looks like this:

```
|--database
  |-- memos
     |-- memokey1
       |-- title: "Title"
       |-- content: "Message"
     |-- memokey2
       |-- title: "Important Title"
       |-- content: "Important Message"
     |-- generatedMemokey3
       |-- title: "Important numbers"
       |-- content: "1337, 42, 3.14159265359"
  |-- users
     |-- userKey1
       |-- name: "John Doe"
       |-- memos
         |-- memokey1 : true //The values here don't matter, we only need the keys.
         |-- memokey2 : true
         |-- generatedMemokey3 : true
     |-- userKey2
       |-- name: "Max Doe"
```

Section 231.8: Designing and understanding how to retrieve realtime data from the Firebase Database

This example assumes that you have already set up a Firebase Realtime Database. If you are a starter, then please inform yourself here on how to add Firebase to your Android project.

First, add the dependency of the Firebase Database to the app level *build.gradle* file:

```
compile 'com.google.firebase:firebase-database:9.4.0'
```

Now, let us create a chat app which stores data into the Firebase Database.

Step 1: Create a class named Chat

Just create a class with some basic variables required for the chat:

```
public class Chat{
    public String name, message;
}
```

Step 2: Create some JSON data

For sending/retrieving data to/from the Firebase Database, you need to use JSON. Let us assume that some chats are already stored at the root level in the database. The data of these chats could look like as follows:

```
[
```

```
    {
        "name":"John Doe",
        "message":"My first Message"
    },
    {
        "name":"John Doe",
        "message":"Second Message"
    },
    {
        "name":"John Doe",
        "message":"Third Message"
    }
]
```

Step 3: Adding the listeners

There are three types of listeners. In the following example we are going to use the `childEventListener`:

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference() // Referencing the root of
the database.
        .child("chats"); // Referencing the "chats" node under the root.

chatDb.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        // This function is called for every child id chat in this case, so using the above
        // example, this function is going to be called 3 times.

        // Retrieving the Chat object from this function is simple.
        Chat chat; // Create a null chat object.

        // Use the getValue function in the dataSnapshot and pass the object's class name to
        // which you want to convert and get data. In this case it is Chat.class.
        chat = dataSnapshot.getValue(Chat.class);

        // Now you can use this chat object and add it into an ArrayList or something like
        // that and show it in the recycler view.
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
        // This function is called when any of the node value is changed, dataSnapshot will
        // get the data with the key of the child, so you can swap the new value with the
        // old one in the ArrayList or something like that.

        // To get the key, use the .getKey() function.
        // To get the value, use code similar to the above one.
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
        // This function is called when any of the child node is removed. dataSnapshot will
        // get the data with the key of the child.

        // To get the key, use the s String parameter .
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        // This function is called when any of the child nodes is moved to a different position.

        // To get the key, use the s String parameter.
```

```
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // If anything goes wrong, this function is going to be called.

        // You can get the exception by using databaseError.toException();
    }
});
```

Step 4: Add data to the database

Just create a Chat class object and add the values as follows:

```
Chat chat=new Chat();
chat.name="John Doe";
chat.message="First message from android";
```

Now get a reference to the chats node as done in the retrieving session:

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

Before you start adding data, keep in mind that you need one more deep reference since a chat node has several more nodes and adding a new chat means adding a new node containing the chat details. We can generate a new and unique name of the node using the `push()` function on the `DatabaseReference` object, which will return another `DatabaseReference`, which in turn points to a newly formed node to insert the chat data.

Example

```
// The parameter is the chat object that was newly created a few lines above.
chatDb.push().setValue(chat);
```

The `setValue()` function will make sure that all of the application's `onDataChanged` functions are getting called (including the same device), which happens to be the attached listener of the "chats" node.

Chapter 232: Firebase App Indexing

Section 232.1: Supporting Http URLs

Step 1: Allow Google to Crawl to your content.Edit server's robot.txt file.You can control google crawling for your content by editing this file,you can refer to this link for more details.

Step 2: Associate your App with your website.Include assetlinks.json You upload it to your web server's .well-known directory.Content of your assetlinks.json are as-

```
      [{
"relation": ["delegate_permission/common.handle_all_urls"],
"target" :
{ "namespace": "android_app",
"package_name": "<your_package_name>",
"sha256_cert_fingerprints": ["<hash_of_app_certificate>"] }
```

}]

Step 3: Include App links in your manifest file to redirect Urls into your Application like below,

```
<activity
      android:name=".activity.SampleActivity"
      android:label="@string/app_name"
      android:windowSoftInputMode="adjustResize|stateAlwaysHidden">
       <intent-filter>
             <action android:name="android.intent.action.VIEW" />
             <category android:name="android.intent.category.DEFAULT" />
             <category android:name="android.intent.category.BROWSABLE" />
       <data
            android:host="example.live"
            android:pathPrefix="/vod"
            android:scheme="https"/>
       <data
            android:host="example.live"
            android:pathPrefix="/vod"
            android:scheme="http"/>
    </intent-filter>
  </activity>
```

Refer to this if you want learn about each and every tag here.

< action> Specify the ACTION_VIEW intent action so that the intent filter can be reached from Google Search.

< data> Add one or more tags, where each tag represents a URI format that resolves to the activity. At minimum, the tag must include the android:scheme attribute. You can add additional attributes to further refine the type of URI that the activity accepts. For example, you might have multiple activities that accept similar URIs, but which differ simply based on the path name. In this case, use the android:path attribute or its variants (pathPattern or pathPrefix) to differentiate which activity the system should open for different URI paths.

< category> Include the BROWSABLE category. The BROWSABLE category is required in order for the intent filter to be accessible from a web browser. Without it, clicking a link in a browser cannot resolve to your app. The DEFAULT category is optional, but recommended. Without this category, the activity can be started only with an explicit intent, using your app component name.

Step 4: Handle incoming URLS

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_schedule);
        onNewIntent(getIntent());
    }

    protected void onNewIntent(Intent intent) {
    String action = intent.getAction();
    Uri data = intent.getData();
    if (Intent.ACTION_VIEW.equals(action) && data != null) {
        articleId = data.getLastPathSegment();
        TextView linkText = (TextView)findViewById(R.id.link);
        linkText.setText(data.toString());
    }
```

}

Step 5: You can test this by using Android Debug Bridge command or studio configurations. Adb command: Launch your application and then run this command:

```
adb shell am start -a android.intent.action.VIEW -d "{URL}" < package name >
```

Android Studio Configurations: **Android studio > Build > Edit Configuration >Launch options>select URL>then type in your Url here >Apply** and test.Run your application if “Run” window shows error then you need to check your URL format with your applinks mentioned in manifest otherwise it will successfully run,and redirect to page mentioned your URL if specified.

Section 232.2: Add AppIndexing API

For Adding this to project you can find official doc easily but in this example I'm going to highlight some of the key areas to be taken care of.

Step 1: Add google service

```
dependencies {
      ...
      compile 'com.google.android.gms:play-services-appindexing:9.4.0'
      ...
    }
```

Step 2: Import classes

```
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;
```

Step 3: Add App Indexing API calls

```
private GoogleApiClient mClient;
private Uri mUrl;
private String mTitle;
private String mDescription;

//If you know the values that to be indexed then you can initialize these variables in onCreate()
@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
mClient = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
mUrl = "http://examplepetstore.com/dogs/standard-poodle";
mTitle = "Standard Poodle";
mDescription = "The Standard Poodle stands at least 18 inches at the withers";
}

//If your data is coming from a network request, then initialize these value in onResponse() and make
checks for NPE so that your code won't fall apart.

//setting title and description for App Indexing
mUrl = Uri.parse("android-app://com.famelive/https/m.fame.live/vod/" +model.getId());
mTitle =   model.getTitle();
mDescription = model.getDescription();

mClient.connect();
AppIndex.AppIndexApi.start(mClient, getAction());

@Override
protected void onStop() {
if (mTitle != null && mDescription != null && mUrl != null) //if your response fails then check
whether these are initialized or not
   if (getAction() != null) {
       AppIndex.AppIndexApi.end(mClient, getAction());
       mClient.disconnect();
    }
 super.onStop();
}

 public Action getAction() {
   Thing object = new Thing.Builder()
       .setName(mTitle)
       .setDescription(mDescription)
       .setUrl(mUrl)
       .build();

 return new Action.Builder(Action.TYPE_WATCH)
       .setObject(object)
       .setActionStatus(Action.STATUS_TYPE_COMPLETED)
       .build();
}
```

To test this just follow the step 4 in Remarks given below.

Chapter 233: Firebase Crash Reporting

Section 233.1: How to report an error

Firebase Crash Reporting automatically generates reports for fatal errors (or uncaught exceptions).

You can create your custom report using:

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

You can check in the log when FirebaseCrash initialized the module:

> 07–20 08:57:24.442 D/FirebaseCrashApiImpl: **FirebaseCrash reporting API initialized** 07–20 08:57:24.442 I/FirebaseCrash: **FirebaseCrash reporting initialize**d com.google.firebase.crash.internal.zzg@3333d325 07–20 08:57:24.442 D/FirebaseApp: **Initialized class com.google.firebase.crash.FirebaseCrash.**

And then when it sent the exception:

> 07–20 08:57:47.052 D/FirebaseCrashApiImpl: **throwable java.lang.Exception: My first Android non-fatal error** 07–20 08:58:18.822 D/FirebaseCrashSenderServiceImpl: **Response code: 200** 07–20 08:58:18.822 D/FirebaseCrashSenderServiceImpl: **Report sent**

You can add custom logs to your report with

```
FirebaseCrash.log("Activity created");
```

Section 233.2: How to add Firebase Crash Reporting to your app

In order to add *Firebase Crash Reporting* to your app, perform the following steps:

- Create an app on the *Firebase Console* here.
- Copy the `google-services.json` file from your project into your in `app/` directory.
- Add the following rules to your root-level *build.gradle* file in order to include the `google-services` plugin:

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

- In your module Gradle file, add the `apply plugin` line at the bottom of the file to enable the Gradle plugin:

```
apply plugin: 'com.google.gms.google-services'
```

- Add the dependency for *Crash Reporting* to your app-level *build.gradle* file:

```
compile 'com.google.firebase:firebase-crash:10.2.1'
```

- You can then fire a custom exception from your application by using the following line:

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

  All your fatal exceptions will be reported to your *Firebase Console*.

- If you want to add custom logs to a console, you can use the following code:

```
FirebaseCrash.log("Level 2 completed.");
```

For more information, please visit:

- Official documentation
- Stack Overflow dedicated topic

Chapter 234: Twitter APIs

Section 234.1: Creating login with twitter button and attach a callback to it

1. Inside your layout, add a Login button with the following code:

```
<com.twitter.sdk.android.core.identity.TwitterLoginButton
   android:id="@+id/twitter_login_button"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"/>
```

2. In the Activity or Fragment that displays the button, you need to create and attach a Callback to the Login Buttonas the following:

```
import com.twitter.sdk.android.core.Callback;
import com.twitter.sdk.android.core.Result;
import com.twitter.sdk.android.core.TwitterException;
import com.twitter.sdk.android.core.TwitterSession;
import com.twitter.sdk.android.core.identity.TwitterLoginButton;
...

loginButton = (TwitterLoginButton) findViewById(R.id.login_button);
loginButton.setCallback(new Callback<TwitterSession>() {
    @Override
    public void success(Result<TwitterSession> result) {
        Log.d(TAG, "userName: " + session.getUserName());
        Log.d(TAG, "userId: " + session.getUserId());
        Log.d(TAG, "authToken: " + session.getAuthToken());
        Log.d(TAG, "id: " + session.getId());
        Log.d(TAG, "authToken: " + session.getAuthToken().token);
        Log.d(TAG, "authSecret: " + session.getAuthToken().secret);
    }

    @Override
    public void failure(TwitterException exception) {
        // Do something on failure
    }
});
```

3. Pass the result of the authentication Activity back to the button:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // Make sure that the loginButton hears the result from any
    // Activity that it triggered.
    loginButton.onActivityResult(requestCode, resultCode, data);
}
```

Note, If using the TwitterLoginButton in a Fragment, use the following steps instead:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
    super.onActivityResult(requestCode, resultCode, data);

    // Pass the activity result to the fragment, which will then pass the result to the login
    // button.
    Fragment fragment = getFragmentManager().findFragmentById(R.id.your_fragment_id);
    if (fragment != null) {
        fragment.onActivityResult(requestCode, resultCode, data);
    }
}
```

4. Add the following lines to your **build.gradle** dependencies:

```
apply plugin: 'io.fabric'

repositories {
    maven { url 'https://maven.fabric.io/public' }
}

compile('com.twitter.sdk.android:twitter:1.14.1@aar') {
    transitive = true;
}
```

Chapter 235: Youtube-API

Section 235.1: Activity extending YouTubeBaseActivity

```
public class CustomYouTubeActivity extends YouTubeBaseActivity implements
YouTubePlayer.OnInitializedListener, YouTubePlayer.PlayerStateChangeListener {

    private YouTubePlayerView mPlayerView;
    private YouTubePlayer mYouTubePlayer;
    private String mVideoId = "B08iLAtS3AQ";
    private String mApiKey;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mApiKey = Config.YOUTUBE_API_KEY;
        mPlayerView = new YouTubePlayerView(this);
        mPlayerView.initialize(mApiKey, this); // setting up OnInitializedListener
        addContentView(mPlayerView, new LayoutParams(LayoutParams.MATCH_PARENT,
                LayoutParams.MATCH_PARENT)); //show it in full screen
    }

    //Called when initialization of the player succeeds.
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
                                        YouTubePlayer player,
                                        boolean wasRestored) {

        player.setPlayerStateChangeListener(this); // setting up the player state change listener
        this.mYouTubePlayer = player;
        if (!wasRestored)
            player.cueVideo(mVideoId);
    }

        @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
                                        YouTubeInitializationResult errorReason) {

        Toast.makeText(this, "Error While initializing", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAdStarted() {
    }

    @Override
    public void onLoaded(String videoId) { //video has been loaded
        if(!TextUtils.isEmpty(mVideoId) && !this.isFinishing() && mYouTubePlayer != null)
            mYouTubePlayer.play(); // if we don't call play then video will not auto play, but user
still has the option to play via play button
    }

    @Override
    public void onLoading() {
    }

    @Override
    public void onVideoEnded() {
    }
```

```
    @Override
    public void onVideoStarted() {

    }

    @Override
    public void onError(ErrorReason reason) {
        Log.e("onError", "onError : " + reason.name());
    }

}
```

Section 235.2: Consuming YouTube Data API on Android

This example will guide you how to get playlist data using the YouTube Data API on Android.

SHA-1 fingerprint

First you need to get an SHA-1 fingerprint for your machine. There are various methods for retrieving it. You can choose any method provided in this Q&A.

Google API console and YouTube key for Android

Now that you have an SHA-1 fingerprint, open the Google API console and create a project. Go to this page and create a project using that SHA-1 key and enable the YouTube Data API. Now you will get a key. This key will be used to send requests from Android and fetch data.

Gradle part

You will have to add the following lines to your Gradle file for the YouTube Data API:

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

In order to use YouTube's native client to send requests, we have to add the following lines in Gradle:

```
compile 'com.google.http-client:google-http-client-android:+'
compile 'com.google.api-client:google-api-client-android:+'
compile 'com.google.api-client:google-api-client-gson:+'
```

The following configuration also needs to be added in Gradle in order to avoid conflicts:

```
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
```

Below it is shown how the *gradle.build* would finally look like.

build.gradle

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.aam.skillschool"
        minSdkVersion 19
        targetSdkVersion 25
```

```
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    configurations.all {
        resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:support-v4:25.3.1'
    compile 'com.google.http-client:google-http-client-android:+'
    compile 'com.google.api-client:google-api-client-android:+'
    compile 'com.google.api-client:google-api-client-gson:+'
}
```

Now comes the Java part. Since we will be using `HttpTransport` for networking and `GsonFactory` for converting JSON into POJO, we don't need any other library to send any requests.

Now I want to show how to get playlists via the YouTube API by providing the playlist IDs. For this task I will use `AsyncTask`. To understand how we request parameters and to understand the flow, please take a look at the YouTube Data API.

```
public class GetPlaylistDataAsyncTask extends AsyncTask<String[], Void, PlaylistListResponse> {
    private static final String YOUTUBE_PLAYLIST_PART = "snippet";
    private static final String YOUTUBE_PLAYLIST_FIELDS = "items(id,snippet(title))";

    private YouTube mYouTubeDataApi;

    public GetPlaylistDataAsyncTask(YouTube api) {
        mYouTubeDataApi = api;
    }

    @Override
    protected PlaylistListResponse doInBackground(String[]... params) {

        final String[] playlistIds = params[0];

        PlaylistListResponse playlistListResponse;
        try {
            playlistListResponse = mYouTubeDataApi.playlists()
                    .list(YOUTUBE_PLAYLIST_PART)
                    .setId(TextUtils.join(",", playlistIds))
                    .setFields(YOUTUBE_PLAYLIST_FIELDS)
                    .setKey(AppConstants.YOUTUBE_KEY) //Here you will have to provide the keys
                    .execute();
        } catch (IOException e) {
            e.printStackTrace();
            return null;
```

```
        }

        return playlistListResponse;
    }
}
```

The above asynchronous task will return an instance of `PlaylistListResponse` which is a build-in class of the YouTube SDK. It has all the required fields, so we don't have to create POJOs ourself.

Finally, in our `MainActivity` we will have to do the following:

```
public class MainActivity extends AppCompatActivity {
    private YouTube mYoutubeDataApi;
    private final GsonFactory mJsonFactory = new GsonFactory();
    private final HttpTransport mTransport = AndroidHttp.newCompatibleTransport();
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_review);
        mYoutubeDataApi = new YouTube.Builder(mTransport, mJsonFactory, null)
                .setApplicationName(getResources().getString(R.string.app_name))
                .build();
        String[] ids = {"some playlists ids here separated by "," };
        new GetPlaylistDataAsyncTask(mYoutubeDataApi) {
            ProgressDialog progressDialog = new ProgressDialog(getActivity());

            @Override
            protected void onPreExecute() {
                progressDialog.setTitle("Please wait.....");
                progressDialog.show();
                super.onPreExecute();
            }

            @Override
            protected void onPostExecute(PlaylistListResponse playlistListResponse) {
                super.onPostExecute(playlistListResponse);
                //Here we get the playlist data
                progressDialog.dismiss();
                Log.d(TAG, playlistListResponse.toString());
            }
        }.execute(ids);
    }
}
```

Section 235.3: Launching StandAlonePlayerActivity

1. Launch standalone player activity

```
    Intent standAlonePlayerIntent = YouTubeStandalonePlayer.createVideoIntent((Activity)
context,
            Config.YOUTUBE_API_KEY, // which you have created in step 3
            videoId, // video which is to be played
            100,     //The time, in milliseconds, where playback should start in the video
            true,    //autoplay or not
            false);   //lightbox mode or not; false will show in fullscreen
    context.startActivity(standAlonePlayerIntent);
```

Section 235.4: YoutubePlayerFragment in portrait Activty

The following code implements a simple YoutubePlayerFragment. The activity's layout is locked in portrait mode and when orientation changes or the user clicks full screen at the YoutubePlayer it turns to lansscape with the YoutubePlayer filling the screen. The YoutubePlayerFragment does not need to extend an activity provided by the Youtube library. It needs to implement YouTubePlayer.OnInitializedListener in order to get the YoutubePlayer initialized. So our Activity's class is the following

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.Toast;

import com.google.android.youtube.player.YouTubeInitializationResult;
import com.google.android.youtube.player.YouTubePlayer;
import com.google.android.youtube.player.YouTubePlayerFragment;

public class MainActivity extends AppCompatActivity implements YouTubePlayer.OnInitializedListener
{

    public static final String API_KEY ;
    public static final String VIDEO_ID = "B08iLAtS3AQ";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        YouTubePlayerFragment youTubePlayerFragment = (YouTubePlayerFragment) getFragmentManager()
                .findFragmentById(R.id.youtubeplayerfragment);

        youTubePlayerFragment.initialize(API_KEY, this);

    }

    /**
     *
     * @param provider The provider which was used to initialize the YouTubePlayer
     * @param youTubePlayer A YouTubePlayer which can be used to control video playback in the
provider.
     * @param wasRestored Whether the player was restored from a previously saved state, as part of
the YouTubePlayerView
     *                    or YouTubePlayerFragment restoring its state. true usually means playback
is resuming from where
     *                    the user expects it would, and that a new video should not be loaded
     */
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer
youTubePlayer, boolean wasRestored) {

youTubePlayer.setFullscreenControlFlags(YouTubePlayer.FULLSCREEN_FLAG_CONTROL_ORIENTATION |
                    YouTubePlayer.FULLSCREEN_FLAG_ALWAYS_FULLSCREEN_IN_LANDSCAPE);


        if(!wasRestored) {
            youTubePlayer.cueVideo(VIDEO_ID);
        }
    }

    /**
```

```
     *
     * @param provider The provider which failed to initialize a YouTubePlayer.
     * @param error The reason for this failure, along with potential resolutions to this failure.
     */
    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
YouTubeInitializationResult error) {

        final int REQUEST_CODE = 1;

        if(error.isUserRecoverableError()) {
            error.getErrorDialog(this,REQUEST_CODE).show();
        } else {
            String errorMessage = String.format("There was an error initializing the YoutubePlayer
(%1$s)", error.toString());
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();
        }
    }
}
```

A YoutubePlayerFragment can be added to the activity's layout xaml as followed

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical"
              android:paddingBottom="@dimen/activity_vertical_margin"
              android:paddingLeft="@dimen/activity_horizontal_margin"
              android:paddingRight="@dimen/activity_horizontal_margin"
              android:paddingTop="@dimen/activity_vertical_margin"
              tools:context=".MainActivity">

    <fragment
       android:id="@+id/youtubeplayerfragment"
       android:name="com.google.android.youtube.player.YouTubePlayerFragment"
       android:layout_width="match_parent"
       android:layout_height="wrap_content" />


    <ScrollView
       android:layout_width="match_parent"
       android:layout_height="match_parent">

        <LinearLayout
           android:layout_width="match_parent"
           android:layout_height="wrap_content"
           android:orientation="vertical">


            <TextView
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:layout_gravity="center_horizontal"
               android:layout_marginTop="20dp"
               android:text="This is a YoutubePlayerFragment example"
               android:textStyle="bold" />

            <TextView
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
```

```
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="20dp"
                android:text="This is a YoutubePlayerFragment example"
                android:textStyle="bold"/>

        </LinearLayout>
    </ScrollView>

</LinearLayout>
```

Lastly you need to add the following attributes in your Manifest file inside the activity's tag

```
android:configChanges="keyboardHidden|orientation|screenSize"
android:screenOrientation="portrait"
```

Section 235.5: YouTube Player API

Obtaining the Android API Key :

First you'll need to get the SHA-1 fingerprint on your machine using java keytool. Execute the below command in cmd/terminal to get the SHA-1 fingerprint.

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -
keypass android
```

MainActivity.java

```
public class Activity extends YouTubeBaseActivity implements YouTubePlayer.OnInitializedListener {

    private static final int RECOVERY_DIALOG_REQUEST = 1;

    // YouTube player view
    private YouTubePlayerView youTubeView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.activity_main);

        youTubeView = (YouTubePlayerView) findViewById(R.id.youtube_view);

        // Initializing video player with developer key
        youTubeView.initialize(Config.DEVELOPER_KEY, this);

    }

    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
                                        YouTubeInitializationResult errorReason) {
        if (errorReason.isUserRecoverableError()) {
            errorReason.getErrorDialog(this, RECOVERY_DIALOG_REQUEST).show();
        } else {
            String errorMessage = String.format(
                    getString(R.string.error_player), errorReason.toString());
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
                                        YouTubePlayer player, boolean wasRestored) {
        if (!wasRestored) {

            // loadVideo() will auto play video
            // Use cueVideo() method, if you don't want to play it automatically
            player.loadVideo(Config.YOUTUBE_VIDEO_CODE);

            // Hiding player controls
            player.setPlayerStyle(YouTubePlayer.PlayerStyle.CHROMELESS);
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == RECOVERY_DIALOG_REQUEST) {
            // Retry initialization if user performed a recovery action
            getYouTubePlayerProvider().initialize(Config.DEVELOPER_KEY, this);
        }
    }
```

```
    private YouTubePlayer.Provider getYouTubePlayerProvider() {
        return (YouTubePlayerView) findViewById(R.id.youtube_view);
    }
}
```

Now create `Config.java` file. This file holds the Google Console API developer key and YouTube video id

Config.java

```
public class Config {

    // Developer key
    public static final String DEVELOPER_KEY = "AIzaSyDZtE10od_hXM5aXYEh6Zn7c6brV9ZjKuk";

    // YouTube video id
    public static final String YOUTUBE_VIDEO_CODE = "_oEA18Y8gM0";
}
```

xml file

```
<com.google.android.youtube.player.YouTubePlayerView
               android:id="@+id/youtube_view"
               android:layout_width="match_parent"
               android:layout_height="wrap_content"
               android:layout_marginBottom="30dp" />
```

Chapter 236: Integrate Google Sign In

Parameter	Detail
TAG	A String used while logging
GoogleSignInHelper	A static reference for helper
AppCompatActivity	An Activity reference
GoogleApiClient	A reference of GoogleAPIClient
RC_SIGN_IN	An integer represents activity result constant
isLoggingOut	A boolean to check if log-out task is running or not

Section 236.1: Google Sign In with Helper class

Add below to your `build.gradle` out of `android` tag:

```
// Apply plug-in to app.
apply plugin: 'com.google.gms.google-services'
```

Add below helper class to your util package:

```
/**
 * Created by Andy
 */
public class GoogleSignInHelper implements GoogleApiClient.OnConnectionFailedListener,
        GoogleApiClient.ConnectionCallbacks {
    private static final String TAG = GoogleSignInHelper.class.getSimpleName();

    private static GoogleSignInHelper googleSignInHelper;
    private AppCompatActivity mActivity;
    private GoogleApiClient mGoogleApiClient;
    public static final int RC_SIGN_IN = 9001;
    private boolean isLoggingOut = false;

    public static GoogleSignInHelper newInstance(AppCompatActivity mActivity) {
        if (googleSignInHelper == null) {
            googleSignInHelper = new GoogleSignInHelper(mActivity, fireBaseAuthHelper);
        }
        return googleSignInHelper;
    }

    public GoogleSignInHelper(AppCompatActivity mActivity) {
        this.mActivity = mActivity;
        initGoogleSignIn();
    }


    private void initGoogleSignIn() {
        // [START config_sign_in]
        // Configure Google Sign In
        GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(mActivity.getString(R.string.default_web_client_id))
                .requestEmail()
                .build();
        // [END config_sign_in]

        mGoogleApiClient = new GoogleApiClient.Builder(mActivity)
                .enableAutoManage(mActivity /* FragmentActivity */, this /*
```

```
OnConnectionFailedListener */)
                .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
                .addConnectionCallbacks(this)
                .build();

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
        // An unresolvable error has occurred and Google APIs (including Sign-In) will not
        // be available.
        Log.d(TAG, "onConnectionFailed:" + connectionResult);
        Toast.makeText(mActivity, "Google Play Services error.", Toast.LENGTH_SHORT).show();
    }

    public void getGoogleAccountDetails(GoogleSignInResult result) {
        // Google Sign In was successful, authenticate with FireBase
        GoogleSignInAccount account = result.getSignInAccount();
        // You are now logged into Google
    }
    public void signOut() {

        if (mGoogleApiClient.isConnected()) {

            // Google sign out
            Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
                    new ResultCallback<Status>() {
                        @Override
                        public void onResult(@NonNull Status status) {
                            isLoggingOut = false;
                        }
                    });
        } else {
            isLoggingOut = true;
        }
    }

    public GoogleApiClient getGoogleClient() {
        return mGoogleApiClient;
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        Log.w(TAG, "onConnected");
        if (isLoggingOut) {
            signOut();
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
        Log.w(TAG, "onConnectionSuspended");
    }
}
```

Add below code to your `OnActivityResult` in Activity file:

```
 // [START onactivityresult]
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
        if (requestCode == GoogleSignInHelper.RC_SIGN_IN) {
            GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
            if (result.isSuccess()) {
                googleSignInHelper.getGoogleAccountDetails(result);
            } else {
                // Google Sign In failed, update UI appropriately
                // [START_EXCLUDE]
                Log.d(TAG, "signInWith Google failed");
                // [END_EXCLUDE]
            }
        }
    }
    // [END onactivityresult]

 // [START signin]
    public void signIn() {
        Intent signInIntent =
Auth.GoogleSignInApi.getSignInIntent(googleSignInHelper.getGoogleClient());
        startActivityForResult(signInIntent, GoogleSignInHelper.RC_SIGN_IN);
    }

    // [END signin]
```

Chapter 237: Google signin integration on android

This topic is based on How to integrate google sign-in, On android apps

Section 237.1: Integration of google Auth in your project. (Get a configuration file)

First get the Configuration File for Sign-in from

Open link below

[https://developers.google.com/identity/sign-in/android/start-integrating][1]

click on get A configuration file

- Enter App name And package name and click on choose and configure services
- provide SHA1 Enable google SIGNIN and generate configuration files

Download the configuration file and place the file in app/ folder of your project

1. Add the dependency to your project-level build.gradle:

> classpath 'com.google.gms:google-services:3.0.0'

2. Add the plugin to your app-level build.gradle:(bottom)

> apply plugin: 'com.google.gms.google-services'

3. add this dependency to your app gradle file

> dependencies { compile 'com.google.android.gms:play-services-auth:9.8.0' }

Section 237.2: Code Implementation Google SignIn

- In your sign-in activity's onCreate method, configure Google Sign-In to request the user data required by your app.

```
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
        .build();
```

- create a GoogleApiClient object with access to the Google Sign-In API and the options you specified.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();
```

- Now When User click on Google signin button call this Function.

```
private void signIn() {
Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- implement OnActivityResult to get the response.

```
 @Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
        handleSignInResult(result);
    }
}
```

- Last step Handle The Result and get User Data

```
private void handleSignInResult(GoogleSignInResult result) {
Log.d(TAG, "handleSignInResult:" + result.isSuccess());
if (result.isSuccess()) {
    // Signed in successfully, show authenticated UI.
    GoogleSignInAccount acct = result.getSignInAccount();
    mStatusTextView.setText(getString(R.string.signed_in_fmt, acct.getDisplayName()));
    updateUI(true);
} else {
    // Signed out, show unauthenticated UI.
    updateUI(false);
}
}
```

Chapter 238: Google Awareness APIs

Section 238.1: Get changes for location within a certain range using Fence API

If you want to detect when your user enters a specific location, you can create a fence for the specific location with a radius you want and be notified when your user enters or leaves the location.

```
// Your own action filter, like the ones used in the Manifest
private static final String FENCE_RECEIVER_ACTION = BuildConfig.APPLICATION_ID +
    "FENCE_RECEIVER_ACTION";
private static final String FENCE_KEY = "locationFenceKey";
private FenceReceiver mFenceReceiver;
private PendingIntent mPendingIntent;

// Make sure to initialize your client as described in the Remarks section
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // etc

    // The 0 is a standard Activity request code that can be changed for your needs
    mPendingIntent = PendingIntent.getBroadcast(this, 0,
        new Intent(FENCE_RECEIVER_ACTION), 0);
    registerReceiver(mFenceReceiver, new IntentFilter(FENCE_RECEIVER_ACTION));

    // Create the fence
    AwarenessFence fence = LocationFence.entering(48.136334, 11.581660, 25);
    // Register the fence to receive callbacks.
    Awareness.FenceApi.updateFences(client, new FenceUpdateRequest.Builder()
        .addFence(FENCE_KEY, fence, mPendingIntent)
        .build())
        .setResultCallback(new ResultCallback<Status>() {
            @Override
            public void onResult(@NonNull Status status) {
                if (status.isSuccess()) {
                    Log.i(FENCE_KEY, "Successfully registered.");
                } else {
                    Log.e(FENCE_KEY, "Could not be registered: " + status);
                }
            }
        });
    }
}
```

Now create a BroadcastReciver to recive updates in user state:

```
public class FenceReceiver extends BroadcastReceiver {

    private static final String TAG = "FenceReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        // Get the fence state
        FenceState fenceState = FenceState.extract(intent);

        switch (fenceState.getCurrentState()) {
            case FenceState.TRUE:
                Log.i(TAG, "User is in location");
                break;
```

```
            case FenceState.FALSE:
                Log.i(TAG, "User is not in location");
                break;
            case FenceState.UNKNOWN:
                Log.i(TAG, "User is doing something unknown");
                break;
        }
    }
}
```

Section 238.2: Get current location using Snapshot API

```
// Remember to intialize your client as described in the Remarks section
Awareness.SnapshotApi.getLocation(client)
    .setResultCallback(new ResultCallback<LocationResult>() {
        @Override
        public void onResult(@NonNull LocationResult locationResult) {
            Location location = locationResult.getLocation();
            Log.i(getClass().getSimpleName(), "Coordinates: "location.getLatitude() + "," +
                location.getLongitude() + ", radius : " + location.getAccuracy());
        }
    });
```

Section 238.3: Get changes in user activity with Fence API

If you want to detect when your user starts or finishes an activity such as walking, running, or any other activity of the DetectedActivityFence class, you can create a fence for the activity that you want to detect, and get notified when your user starts/finishes this activity. By using a `BroadcastReceiver`, you will get an `Intent` with data that contains the activity:

```
// Your own action filter, like the ones used in the Manifest.
private static final String FENCE_RECEIVER_ACTION = BuildConfig.APPLICATION_ID +
    "FENCE_RECEIVER_ACTION";
private static final String FENCE_KEY = "walkingFenceKey";
private FenceReceiver mFenceReceiver;
private PendingIntent mPendingIntent;

// Make sure to initialize your client as described in the Remarks section.
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // etc.

    // The 0 is a standard Activity request code that can be changed to your needs.
    mPendingIntent = PendingIntent.getBroadcast(this, 0,
        new Intent(FENCE_RECEIVER_ACTION), 0);
    registerReceiver(mFenceReceiver, new IntentFilter(FENCE_RECEIVER_ACTION));

    // Create the fence.
    AwarenessFence fence = DetectedActivityFence.during(DetectedActivityFence.WALKING);
    // Register the fence to receive callbacks.
    Awareness.FenceApi.updateFences(client, new FenceUpdateRequest.Builder()
        .addFence(FENCE_KEY, fence, mPendingIntent)
        .build())
        .setResultCallback(new ResultCallback<Status>() {
            @Override
            public void onResult(@NonNull Status status) {
                if (status.isSuccess()) {
                    Log.i(FENCE_KEY, "Successfully registered.");
                } else {
```

```
                    Log.e(FENCE_KEY, "Could not be registered: " + status);
                }
            }
        });
    }
}
```

Now you can receive the intent with a `BroadcastReceiver` to get callbacks when the user changes the activity:

```
public class FenceReceiver extends BroadcastReceiver {

    private static final String TAG = "FenceReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        // Get the fence state
        FenceState fenceState = FenceState.extract(intent);

        switch (fenceState.getCurrentState()) {
            case FenceState.TRUE:
                Log.i(TAG, "User is walking");
                break;
            case FenceState.FALSE:
                Log.i(TAG, "User is not walking");
                break;
            case FenceState.UNKNOWN:
                Log.i(TAG, "User is doing something unknown");
                break;
        }
    }
}
```

Section 238.4: Get current user activity using Snapshot API

For one-time, non-constant requests for a user's physical activity, use the Snapshot API:

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getDetectedActivity(client)
    .setResultCallback(new ResultCallback<DetectedActivityResult>() {
        @Override
        public void onResult(@NonNull DetectedActivityResult detectedActivityResult) {
            if (!detectedActivityResult.getStatus().isSuccess()) {
                Log.e(getClass().getSimpleName(), "Could not get the current activity.");
                return;
            }
            ActivityRecognitionResult result = detectedActivityResult
                .getActivityRecognitionResult();
            DetectedActivity probableActivity = result.getMostProbableActivity();
            Log.i(getClass().getSimpleName(), "Activity received : " +
                probableActivity.toString());
        }
    });
```

Section 238.5: Get headphone state with Snapshot API

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getHeadphoneState(client)
    .setResultCallback(new ResultCallback<HeadphoneStateResult>() {
        @Override
```

```
        public void onResult(@NonNull HeadphoneStateResult headphoneStateResult) {
            Log.i(TAG, "Headphone state connection state: " +
                headphoneStateResult.getHeadphoneState()
                .getState() == HeadphoneState.PLUGGED_IN));
        }
    });
```

Section 238.6: Get nearby places using Snapshot API

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getPlaces(client)
    .setResultCallback(new ResultCallback<PlacesResult>() {
        @Override
        public void onResult(@NonNull PlacesResult placesResult) {
            List<PlaceLikelihood> likelihoodList = placesResult.getPlaceLikelihoods();
            if (likelihoodList == null || likelihoodList.isEmpty()) {
                Log.e(getClass().getSimpleName(), "No likely places");
            }
        }
    });
```

As for getting the data in those places, here are some options:

```
Place place = placeLikelihood.getPlace();
String likelihood = placeLikelihood.getLikelihood();
Place place = likelihood.getPlace();
String placeName = place.getName();
String placeAddress = place.getAddress();
String placeCoords = place.getLatLng();
String locale = extractFromLocale(place.getLocale()));
```

Section 238.7: Get current weather using Snapshot API

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getWeather(client)
    .setResultCallback(new ResultCallback<WeatherResult>() {
        @Override
        public void onResult(@NonNull WeatherResult weatherResult) {
            Weather weather = weatherResult.getWeather();
            if (weather == null) {
                Log.e(getClass().getSimpleName(), "No weather received");
            } else {
                Log.i(getClass().getSimpleName(), "Temperature is " +
                        weather.getTemperature(Weather.CELSIUS) + ", feels like " +
                        weather.getFeelsLikeTemperature(Weather.CELSIUS) +
                        ", humidity is " + weather.getHumidity());
            }
        }
    });
```

Chapter 239: Google Maps API v2 for Android

Parameter	Details
GoogleMap	the `GoogleMap` is an object that is received on a `onMapReady()` event
MarkerOptions	`MarkerOptions` is the builder class of a `Marker`, and is used to add one marker to a map.

Section 239.1: Custom Google Map Styles

Map Style

Google Maps come with a set of different styles to be applied, using this code :

```
// Sets the map type to be "hybrid"
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

The different map styles are :

Normal

```
map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
```

Typical road map. Roads, some man-made features, and important natural features such as rivers are shown. Road and feature labels are also visible.

Hybrid

```
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

Satellite photograph data with road maps added. Road and feature labels are also visible.

Satellite

```
map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
```

Satellite photograph data. Road and feature labels are not visible.

Terrain

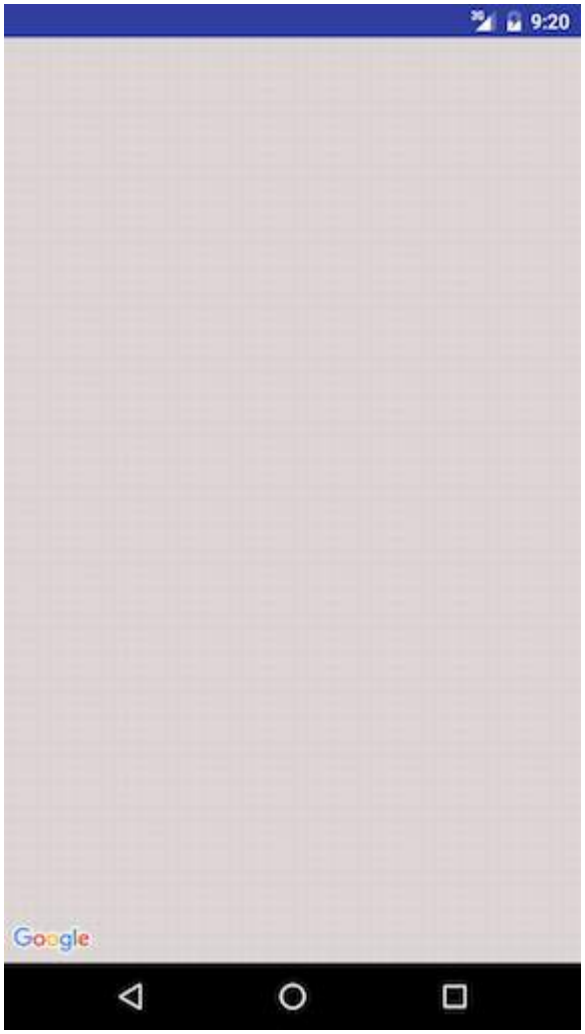
```
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

Topographic data. The map includes colors, contour lines and labels, and perspective shading. Some roads and labels are also visible.

None

```
map.setMapType(GoogleMap.MAP_TYPE_NONE);
```

No tiles. The map will be rendered as an empty grid with no tiles loaded.



OTHER STYLE OPTIONS

Indoor Maps

At high zoom levels, the map will show floor plans for indoor spaces. These are called indoor maps, and are displayed only for the 'normal' and 'satellite' map types.

to enable or disable indoor maps, this is how it's done :

```
GoogleMap.setIndoorEnabled(true).
GoogleMap.setIndoorEnabled(false).
```

We can add custom styles to maps.

In onMapReady method add the following code snippet

```
mMap = googleMap;
    try {
        // Customise the styling of the base map using a JSON object defined
        // in a raw resource file.
        boolean success = mMap.setMapStyle(
                MapStyleOptions.loadRawResourceStyle(
                        MapsActivity.this, R.raw.style_json));

        if (!success) {
            Log.e(TAG, "Style parsing failed.");
        }
    } catch (Resources.NotFoundException e) {
```

```
        Log.e(TAG, "Can't find style.", e);
    }
```

under *res* folder create a folder name *raw* and add the styles json file. Sample style.json file

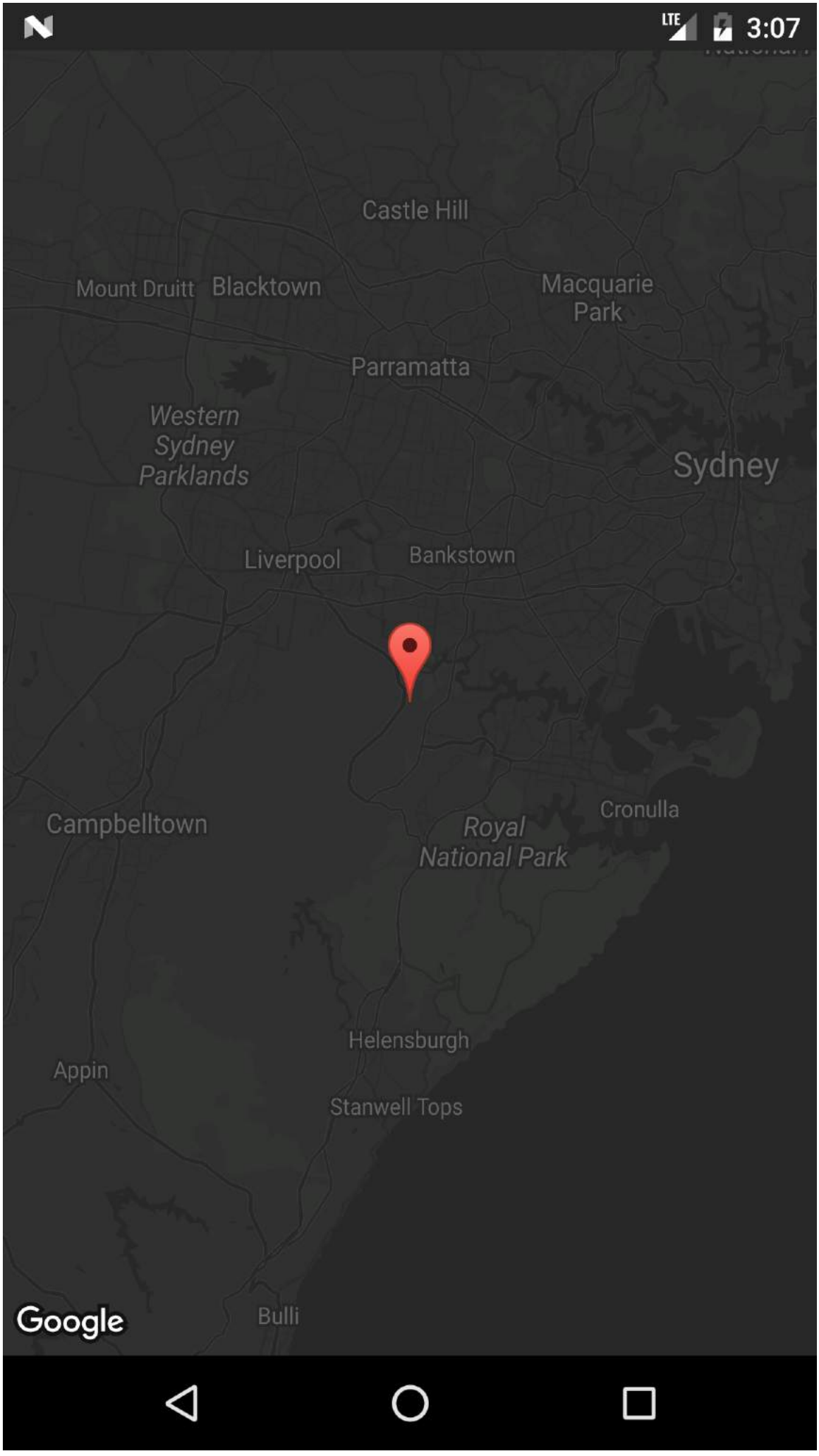
```
    [
  {
    "featureType": "all",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#242f3e"
      }
    ]
  },
  {
    "featureType": "all",
    "elementType": "labels.text.stroke",
    "stylers": [
      {
        "lightness": -80
      }
    ]
  },
  {
    "featureType": "administrative",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#746855"
      }
    ]
  },
  {
    "featureType": "administrative.locality",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#d59563"
      }
    ]
  },
  {
    "featureType": "poi",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#d59563"
      }
    ]
  },
  {
    "featureType": "poi.park",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#263c3f"
      }
    ]
  },
  {
```

```
    "featureType": "poi.park",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#6b9a76"
      }
    ]
  },
  {
    "featureType": "road",
    "elementType": "geometry.fill",
    "stylers": [
      {
        "color": "#2b3544"
      }
    ]
  },
  {
    "featureType": "road",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#9ca5b3"
      }
    ]
  },
  {
    "featureType": "road.arterial",
    "elementType": "geometry.fill",
    "stylers": [
      {
        "color": "#38414e"
      }
    ]
  },
  {
    "featureType": "road.arterial",
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#212a37"
      }
    ]
  },
  {
    "featureType": "road.highway",
    "elementType": "geometry.fill",
    "stylers": [
      {
        "color": "#746855"
      }
    ]
  },
  {
    "featureType": "road.highway",
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#1f2835"
      }
    ]
  },
```

```
  {
    "featureType": "road.highway",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#f3d19c"
      }
    ]
  },
  {
    "featureType": "road.local",
    "elementType": "geometry.fill",
    "stylers": [
      {
        "color": "#38414e"
      }
    ]
  },
  {
    "featureType": "road.local",
    "elementType": "geometry.stroke",
    "stylers": [
      {
        "color": "#212a37"
      }
    ]
  },
  {
    "featureType": "transit",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#2f3948"
      }
    ]
  },
  {
    "featureType": "transit.station",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#d59563"
      }
    ]
  },
  {
    "featureType": "water",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#17263c"
      }
    ]
  },
  {
    "featureType": "water",
    "elementType": "labels.text.fill",
    "stylers": [
      {
        "color": "#515c6d"
      }
    ]
```

```
  },
  {
    "featureType": "water",
    "elementType": "labels.text.stroke",
    "stylers": [
      {
        "lightness": -20
      }
    ]
  }
]
```

To generate styles json file click this link



Section 239.2: Default Google Map Activity

This Activity code will provide basic functionality for including a Google Map using a SupportMapFragment.

The Google Maps V2 API includes an all-new way to load maps.

Activities now have to implement the **OnMapReadyCallBack** interface, which comes with a **onMapReady()** method override that is executed every time we run **SupportMapFragment.getMapAsync(OnMapReadyCallback)**; and the call is successfully completed.

Maps use **Markers** , **Polygons** and **PolyLines** to show interactive information to the user.

MapsActivity.java:

```
public class MapsActivity extends AppCompatActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney, Australia, and move the camera.
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

Notice that the code above inflates a layout, which has a SupportMapFragment nested inside the container Layout, defined with an ID of `R.id.map`. The layout file is shown below:

activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent">

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:tools="http://schemas.android.com/tools"
       xmlns:map="http://schemas.android.com/apk/res-auto"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:id="@+id/map"
       tools:context="com.example.app.MapsActivity"
       android:name="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>
```

Section 239.3: Show Current Location in a Google Map

Here is a full Activity class that places a Marker at the current location, and also moves the camera to the current position.

There are a few thing going on in sequence here:

- Check Location permission
- Once Location permission is granted, call `setMyLocationEnabled()`, build the GoogleApiClient, and connect it
- Once the GoogleApiClient is connected, request location updates

```
public class MapLocationActivity extends AppCompatActivity
        implements OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationListener {

    GoogleMap mGoogleMap;
    SupportMapFragment mapFrag;
    LocationRequest mLocationRequest;
    GoogleApiClient mGoogleApiClient;
    Location mLastLocation;
    Marker mCurrLocationMarker;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        getSupportActionBar().setTitle("Map Location Activity");

        mapFrag = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
        mapFrag.getMapAsync(this);
    }

    @Override
    public void onPause() {
        super.onPause();

        //stop location updates when Activity is no longer active
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }

    @Override
    public void onMapReady(GoogleMap googleMap)
    {
        mGoogleMap=googleMap;
        mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);

        //Initialize Google Play Services
        if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
                //Location Permission already granted
                buildGoogleApiClient();
                mGoogleMap.setMyLocationEnabled(true);
            } else {
```

```
                //Request Location Permission
                checkLocationPermission();
            }
        }
        else {
            buildGoogleApiClient();
            mGoogleMap.setMyLocationEnabled(true);
        }
    }

    protected synchronized void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnected(Bundle bundle) {
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);
        mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {}

    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {}

    @Override
    public void onLocationChanged(Location location)
    {
        mLastLocation = location;
        if (mCurrLocationMarker != null) {
            mCurrLocationMarker.remove();
        }

        //Place current location marker
        LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
        markerOptions.title("Current Position");

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));
        mCurrLocationMarker = mGoogleMap.addMarker(markerOptions);

        //move map camera
        mGoogleMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mGoogleMap.animateCamera(CameraUpdateFactory.zoomTo(11));

        //stop location updates
        if (mGoogleApiClient != null) {
```

```
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
        }
    }

    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;
    private void checkLocationPermission() {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {

            // Should we show an explanation?
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)) {

                // Show an explanation to the user *asynchronously* -- don't block
                // this thread waiting for the user's response! After the user
                // sees the explanation, try again to request the permission.
                new AlertDialog.Builder(this)
                        .setTitle("Location Permission Needed")
                        .setMessage("This app needs the Location permission, please accept to use
location functionality")
                        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {
                                //Prompt the user once explanation has been shown
                                ActivityCompat.requestPermissions(MapLocationActivity.this,
                                        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                                        MY_PERMISSIONS_REQUEST_LOCATION );
                            }
                        })
                        .create()
                        .show();


            } else {
                // No explanation needed, we can request the permission.
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION );
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           String permissions[], int[] grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_LOCATION: {
                // If request is cancelled, the result arrays are empty.
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                    // permission was granted, yay! Do the
                    // location-related task you need to do.
                    if (ContextCompat.checkSelfPermission(this,
                            Manifest.permission.ACCESS_FINE_LOCATION)
                            == PackageManager.PERMISSION_GRANTED) {

                        if (mGoogleApiClient == null) {
                            buildGoogleApiClient();
                        }
                        mGoogleMap.setMyLocationEnabled(true);
                    }
```

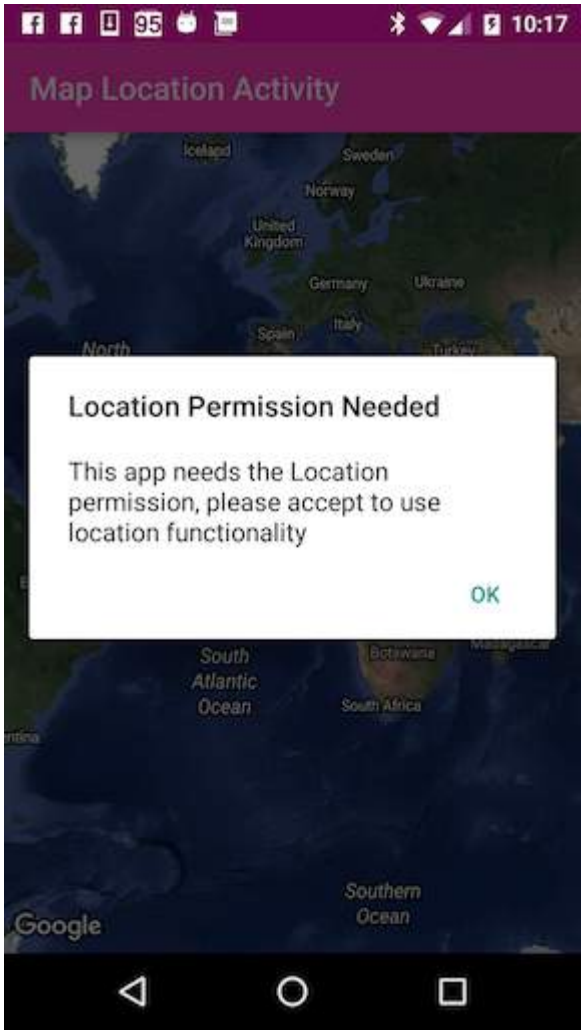
```
                } else {

                    // permission denied, boo! Disable the
                    // functionality that depends on this permission.
                    Toast.makeText(this, "permission denied", Toast.LENGTH_LONG).show();
                }
                return;
            }

            // other 'case' lines to check for other
            // permissions this app might request
        }
    }

}
```

activity_main.xml:

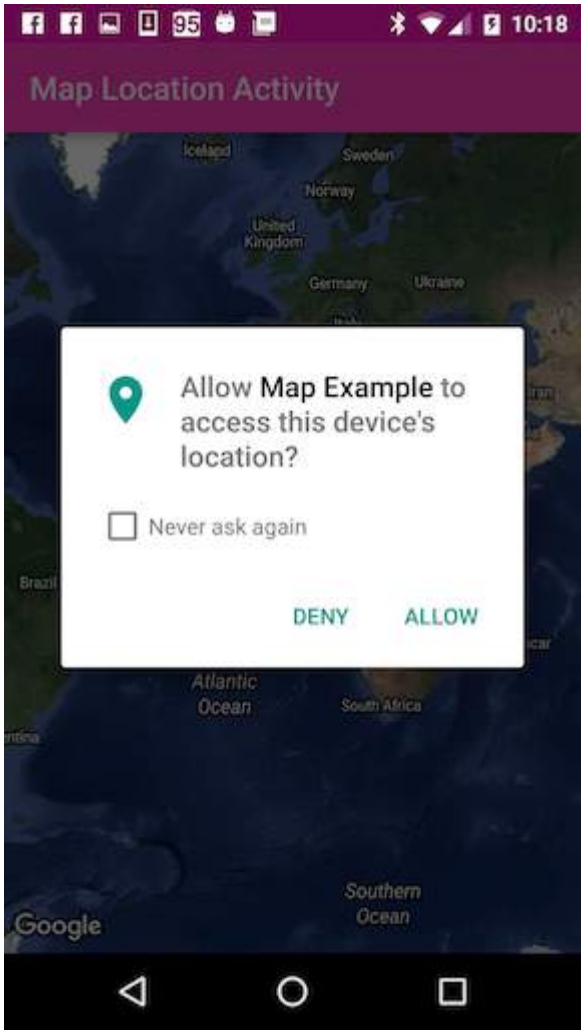
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent">

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:tools="http://schemas.android.com/tools"
       xmlns:map="http://schemas.android.com/apk/res-auto"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:id="@+id/map"
       tools:context="com.example.app.MapLocationActivity"
       android:name="com.google.android.gms.maps.SupportMapFragment"/>

</LinearLayout>
```

Result:

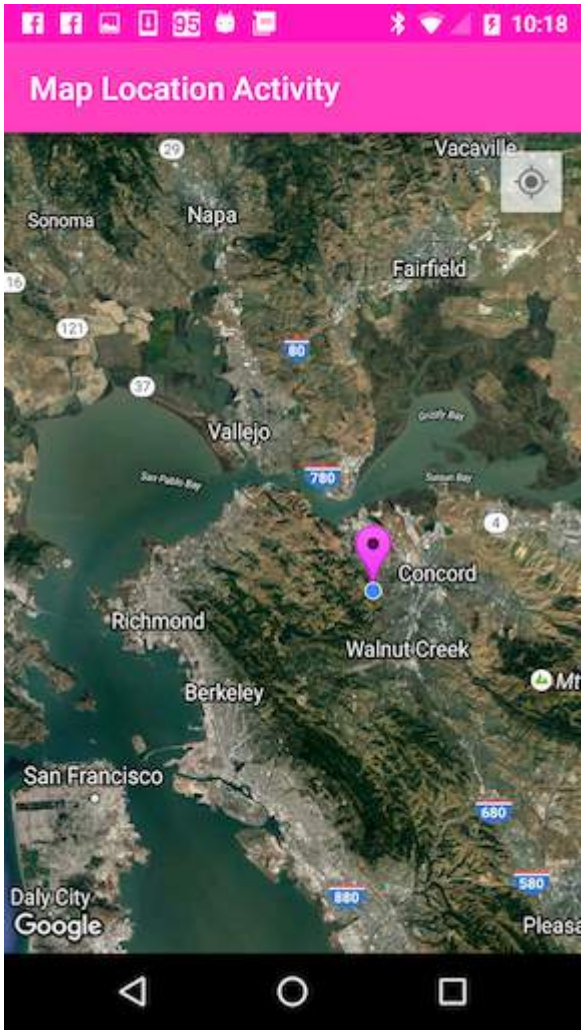
Show explanation if needed on Marshmallow and Nougat using an AlertDialog (this case happens when the user had previously denied a permission request, or had granted the permission and then later revoked it in the settings):



Prompt the user for Location permission on Marshmallow and Nougat by calling `ActivityCompat.requestPermissions()`:



Move camera to current location and place Marker when the Location permission is granted:



Section 239.4: Change Offset

By changing mappoint *x* and *y* values as you need you can change offset possition of google map,by default it will be in the center of the map view. Call below method where you want to change it! Better to use it inside your `onLocationChanged` like `changeOffsetCenter(location.getLatitude(),location.getLongitude());`

```
 public void changeOffsetCenter(double latitude,double longitude) {
            Point mappoint = mGoogleMap.getProjection().toScreenLocation(new LatLng(latitude,
longitude));
            mappoint.set(mappoint.x, mappoint.y-100); // change these values as you need , just hard
coded a value if you want you can give it based on a ratio like using DisplayMetrics  as well

mGoogleMap.animateCamera(CameraUpdateFactory.newLatLng(mGoogleMap.getProjection().fromScreenLocatio
n(mappoint)));
        }
```

Section 239.5: MapView: embedding a GoogleMap in an existing layout

It is possible to treat a GoogleMap as an Android view if we make use of the provided MapView class. Its usage is very similar to MapFragment.

In your layout use MapView as follows:

```
<com.google.android.gms.maps.MapView
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:map="http://schemas.android.com/apk/res-auto"
```

```
   android:id="@+id/map"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   <!--
   map:mapType="0" Specifies a change to the initial map type
   map:zOrderOnTop="true"  Control whether the map view's surface is placed on top of its window
   map:useVieLifecycle="true" When using a MapFragment, this flag specifies whether the lifecycle
of the map should be tied to the fragment's view or the fragment itself
   map:uiCompass="true" Enables or disables the compass
   map:uiRotateGestures="true" Sets the preference for whether rotate gestures should be enabled or
disabled
   map:uiScrollGestures="true" Sets the preference for whether scroll gestures should be enabled or
disabled
   map:uiTiltGestures="true" Sets the preference for whether tilt gestures should be enabled or
disabled
   map:uiZoomGestures="true" Sets the preference for whether zoom gestures should be enabled or
disabled
   map:uiZoomControls="true" Enables or disables the zoom controls
   map:liteMode="true" Specifies whether the map should be created in lite mode
   map:uiMapToolbar="true" Specifies whether the mapToolbar should be enabled
   map:ambientEnabled="true" Specifies whether ambient-mode styling should be enabled
   map:cameraMinZoomPreference="0.0" Specifies a preferred lower bound for camera zoom
   map:cameraMaxZoomPreference="1.0" Specifies a preferred upper bound for camera zoom -->
    />
```

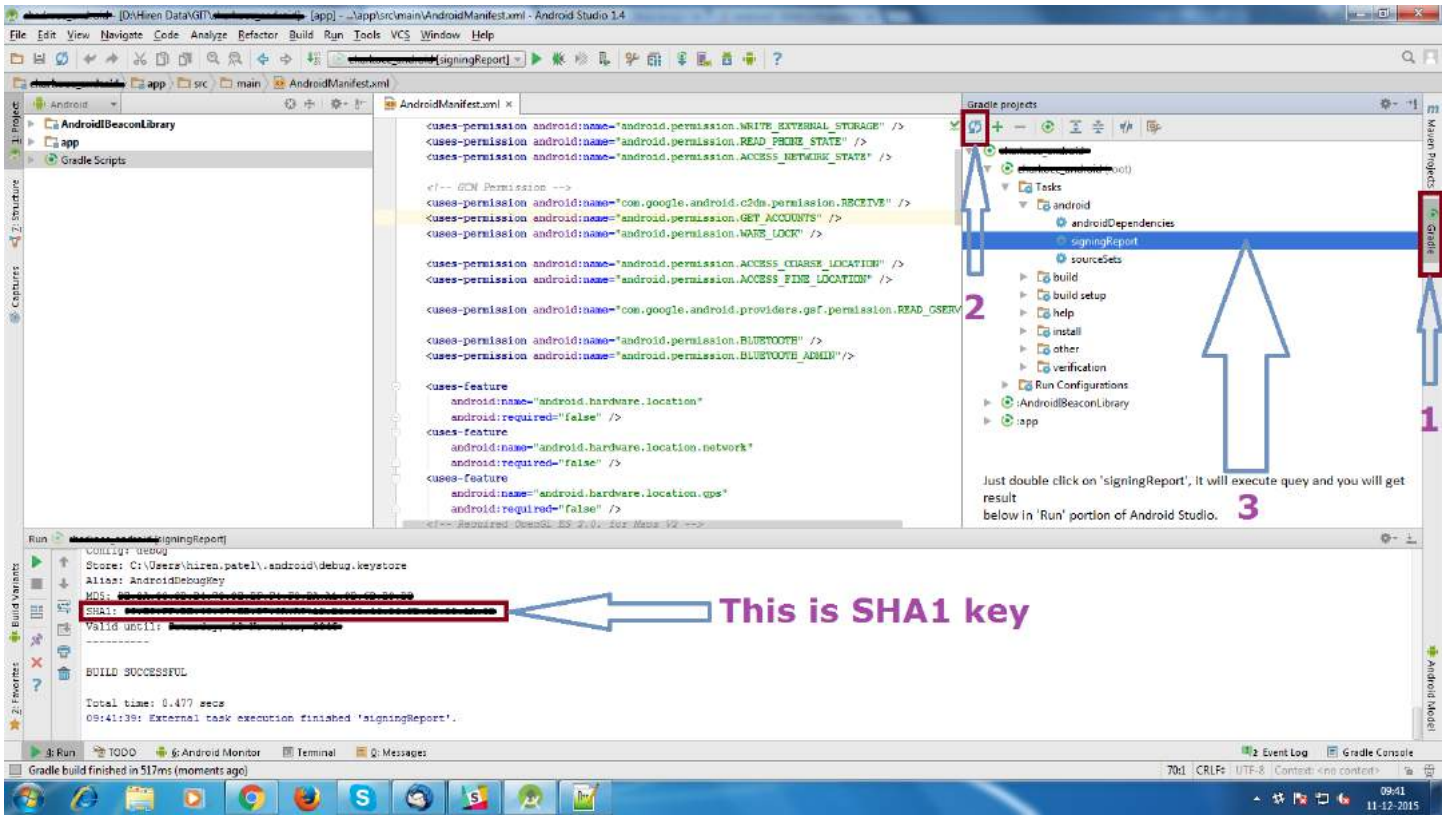
Your activity needs to implement the OnMapReadyCallback interface in order to work:

```
/**
* This shows how to create a simple activity with a raw MapView and add a marker to it. This
* requires forwarding all the important lifecycle methods onto MapView.
*/
public class RawMapViewDemoActivity extends AppCompatActivity implements OnMapReadyCallback {

    private MapView mMapView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.raw_mapview_demo);

        mMapView = (MapView) findViewById(R.id.map);
        mMapView.onCreate(savedInstanceState);

        mMapView.getMapAsync(this);
    }

    @Override
    protected void onResume() {
        super.onResume();
        mMapView.onResume();
    }

    @Override
    public void onMapReady(GoogleMap map) {
        map.addMarker(new MarkerOptions().position(new LatLng(0, 0)).title("Marker"));
    }

    @Override
    protected void onPause() {
        mMapView.onPause();
        super.onPause();
```

```
    }

    @Override
    protected void onDestroy() {
        mMapView.onDestroy();
        super.onDestroy();
    }

    @Override
    public void onLowMemory() {
        super.onLowMemory();
        mMapView.onLowMemory();
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        mMapView.onSaveInstanceState(outState);
    }
}
```

Section 239.6: Get debug SHA1 fingerprint

1. Open Android Studio
2. Open Your Project
3. Click on Gradle (From Right Side Panel, you will see **Gradle Bar**)
4. Click on Refresh (Click on Refresh from **Gradle Bar**, you will see **List** Gradle scripts of your Project)
5. Click on Your Project (Your Project Name form **List** (root))
6. Click on Tasks
7. Click on android
8. Double Click on signingReport (You will get **SHA1** and **MD5** in **Run Bar**)



Section 239.7: Adding markers to a map

To add markers to a Google Map, for example from an `ArrayList` of `MyLocation` Objects, we can do it this way.

The `MyLocation` holder class:

```
public class MyLocation {
  LatLng latLng;
  String title;
  String snippet;
}
```

Here is a method that would take a list of `MyLocation` Objects and place a Marker for each one:

```
private void LocationsLoaded(List<MyLocation> locations){

 for (MyLocation myLoc : locations){
    mMap.addMarker(new MarkerOptions()
      .position(myLoc.latLng)
      .title(myLoc.title)
      .snippet(myLoc.snippet)
      .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA));
 }
}
```

Note: For the purpose of this example, `mMap` is a class member variable of the Activity, where we've assigned it to the map reference received in the `onMapReady()` override.

Section 239.8: UISettings

Using `UISettings`, the appearance of the Google Map can be modified.

Here is an example of some common settings:

```
    mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
    mGoogleMap.getUiSettings().setMapToolbarEnabled(true);
    mGoogleMap.getUiSettings().setZoomControlsEnabled(true);
    mGoogleMap.getUiSettings().setCompassEnabled(true);
```

Result:

Section 239.9: InfoWindow Click Listener

Here is an example of how to define a different action for each Marker's InfoWindow click event.

Use a HashMap in which the marker ID is the key, and the value is the corresponding action it should take when the InfoWindow is clicked.

Then, use a `OnInfoWindowClickListener` to handle the event of a user clicking the InfoWindow, and use the HashMap to determine which action to take.

In this simple example we will open up a different Activity based on which Marker's InfoWindow was clicked.

Declare the HashMap as an instance variable of the Activity or Fragment:

```
 //Declare HashMap to store mapping of marker to Activity
HashMap<String, String> markerMap = new HashMap<String, String>();
```

Then, each time you add a Marker, make an entry in the HashMap with the Marker ID and the action it should take when it's InfoWindow is clicked.

For example, adding two Markers and defining an action to take for each:

```
Marker markerOne = googleMap.addMarker(new MarkerOptions().position(latLng1)
        .title("Marker One")
        .snippet("This is Marker One");
String idOne = markerOne.getId();
markerMap.put(idOne, "action_one");
```

```
Marker markerTwo = googleMap.addMarker(new MarkerOptions().position(latLng2)
        .title("Marker Two")
        .snippet("This is Marker Two");
String idTwo = markerTwo.getId();
markerMap.put(idTwo, "action_two");
```

In the InfoWindow click listener, get the action from the HashMap, and open up the corresponding Activity based on the action of the Marker:

```
mGoogleMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {
 @Override
 public void onInfoWindowClick(Marker marker) {

   String actionId = markerMap.get(marker.getId());

   if (actionId.equals("action_one")) {
     Intent i = new Intent(MainActivity.this, ActivityOne.class);
     startActivity(i);
   } else if (actionId.equals("action_two")) {
     Intent i = new Intent(MainActivity.this, ActivityTwo.class);
     startActivity(i);
   }
  }
});
```

Note If the code is in a Fragment, replace MainActivity.this with getActivity().

Section 239.10: Obtaining the SH1-Fingerprint of your certificate keystore file

In order to obtain a Google Maps API key for your certificate, you must provide the API console with the SH1-fingerprint of your debug/release keystore.

You can obtain the keystore by using the **JDK's keytool** program as described here in the docs.

Another approach is to obtain the fingerprint programmatically by running this snippet with your app signed with the debug/release certificate and printing the hash to the log.

```
PackageInfo info;
try {
    info = getPackageManager().getPackageInfo("com.package.name", PackageManager.GET_SIGNATURES);
    for (Signature signature : info.signatures) {
        MessageDigest md;
        md = MessageDigest.getInstance("SHA");
        md.update(signature.toByteArray());
        String hash= new String(Base64.encode(md.digest(), 0));
        Log.e("hash", hash);
    }
} catch (NameNotFoundException e1) {
    Log.e("name not found", e1.toString());
} catch (NoSuchAlgorithmException e) {
    Log.e("no such an algorithm", e.toString());
} catch (Exception e) {
    Log.e("exception", e.toString());
}
```

Section 239.11: Do not launch Google Maps when the map is clicked (lite mode)

When a Google Map is displayed in lite mode clicking on a map will open the Google Maps application. To disable this functionality you must call `setClickable(false)` on the `MapView`, *e.g.*:

```
final MapView mapView = (MapView)view.findViewById(R.id.map);
mapView.setClickable(false);
```

Chapter 240: Google Drive API

Google Drive is a file hosting service created by **Google**. It provides file storage service and allows the user to upload files in the cloud and also share with other people. Using Google Drive API, we can synchronize files between computer or mobile device and Google Drive Cloud.

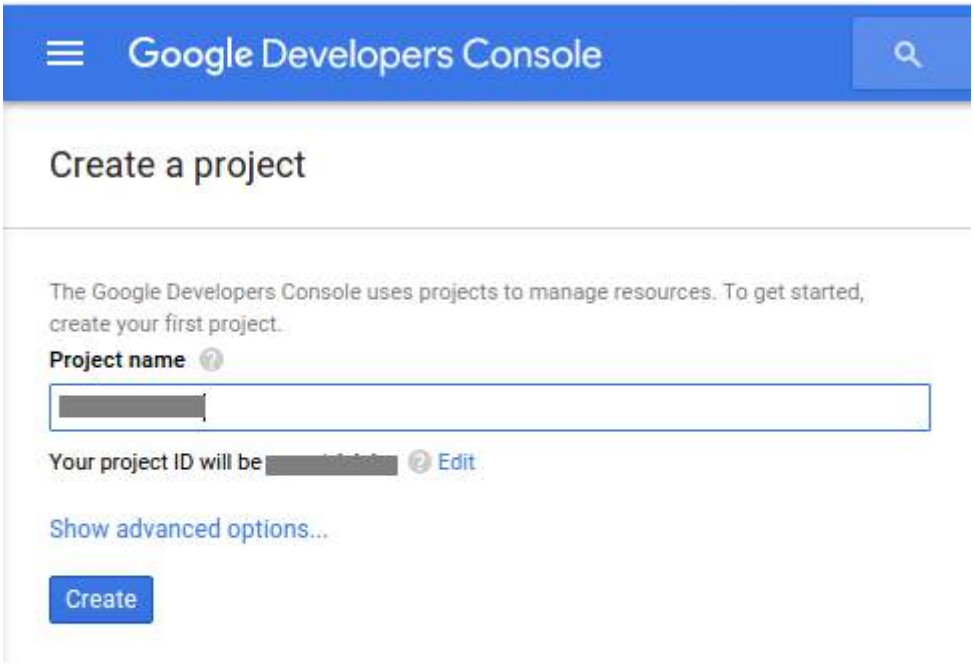
Section 240.1: Integrate Google Drive in Android

Create a New Project on Google Developer Console

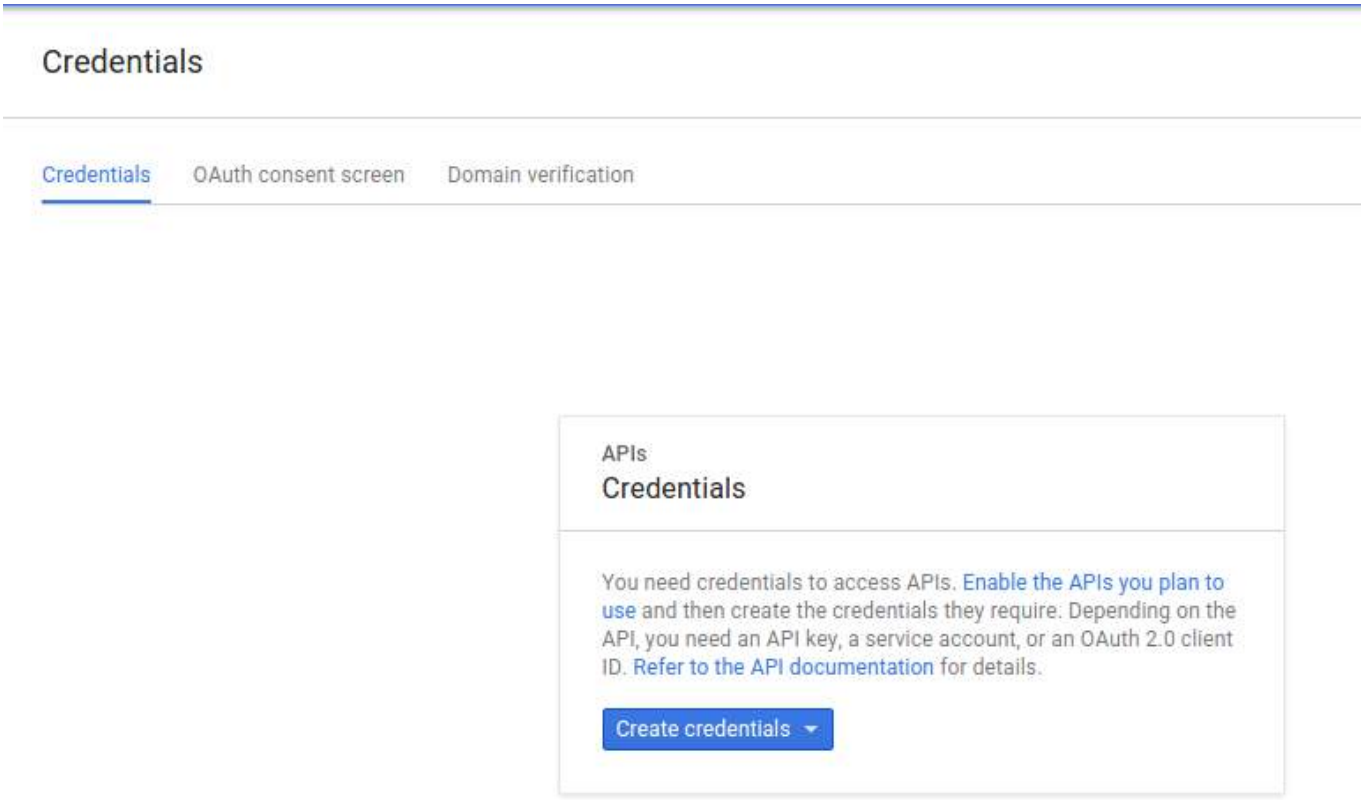
To integrate Android application with Google Drive, create the credentials of project in the Google Developers Console. So, we need to create a project on Google Developer console.

To create a project on Google Developer Console, follow these steps:

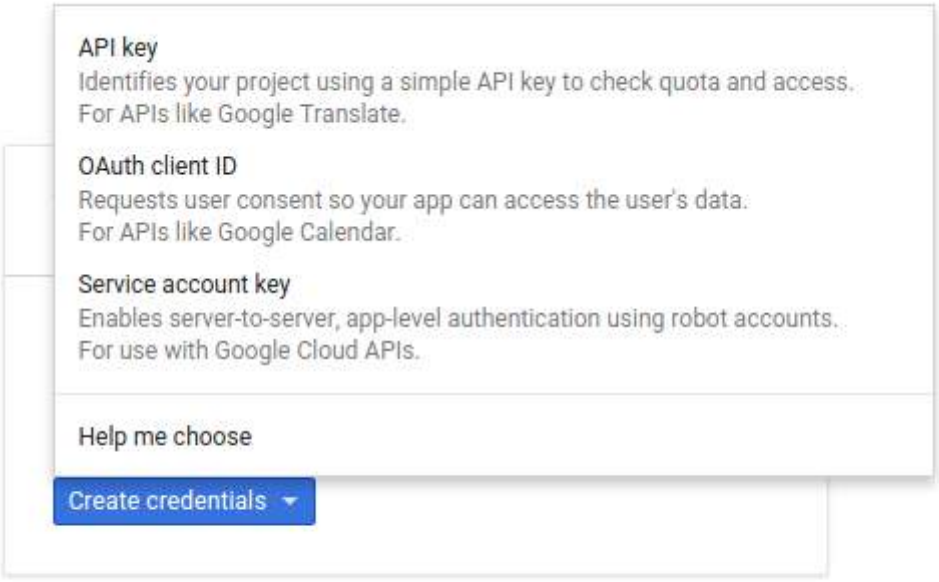
- Go to Google Developer Console for Android. Fill your **project name** in the input field and click on the **create** button to create a new project on Google Developer console.



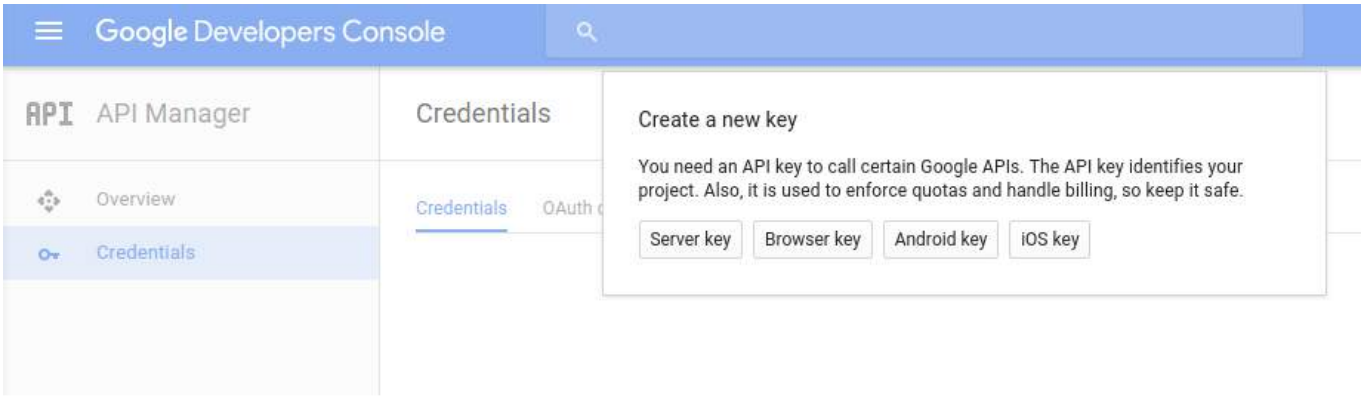
- We need to create credentials to access API. So, click on the **Create credentials** button.



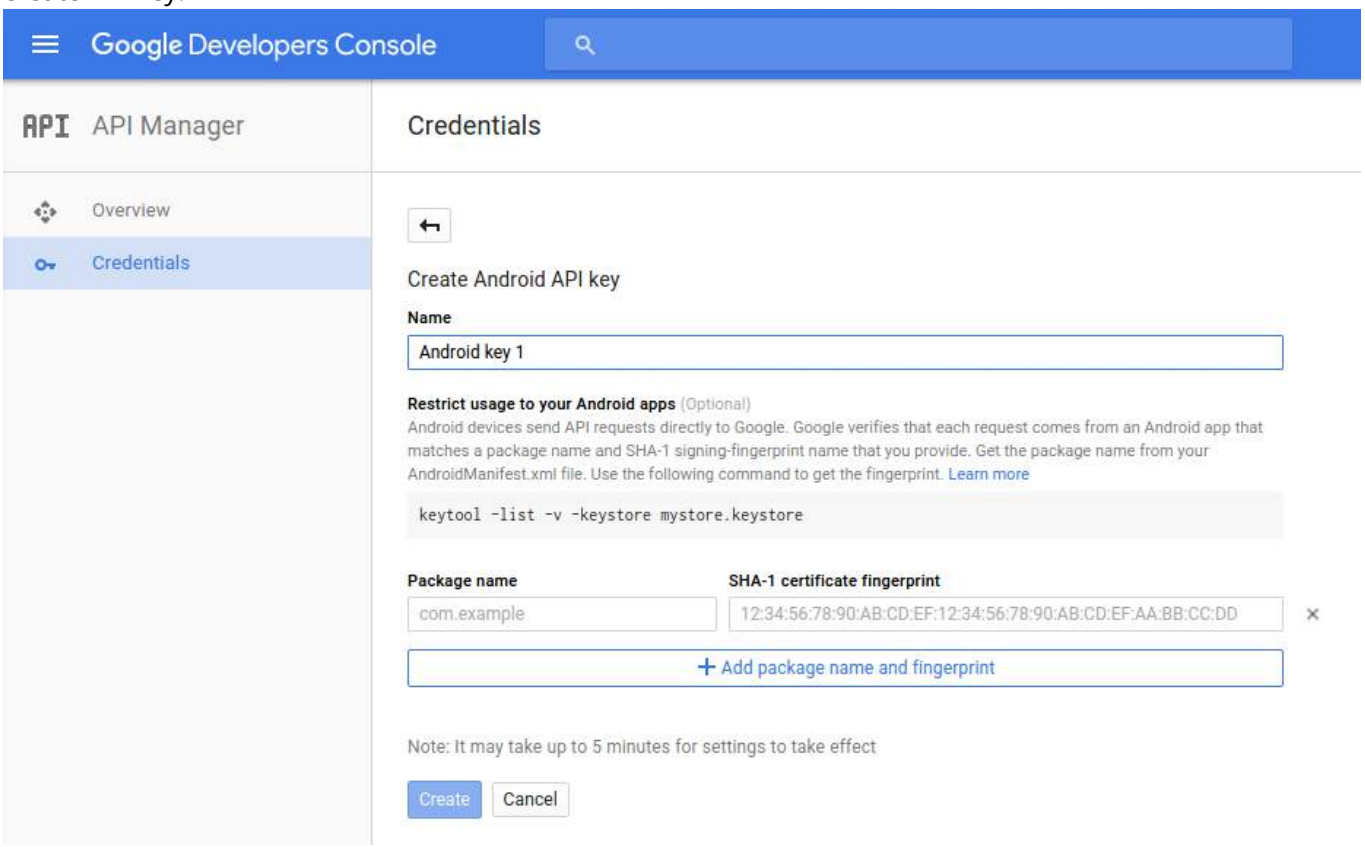
- Now, a pop window will open. Click on **API Key** option in the list to create API key.



- We need an API key to call Google APIs for Android. So, click on the **Android Key** to identify your Android Project.



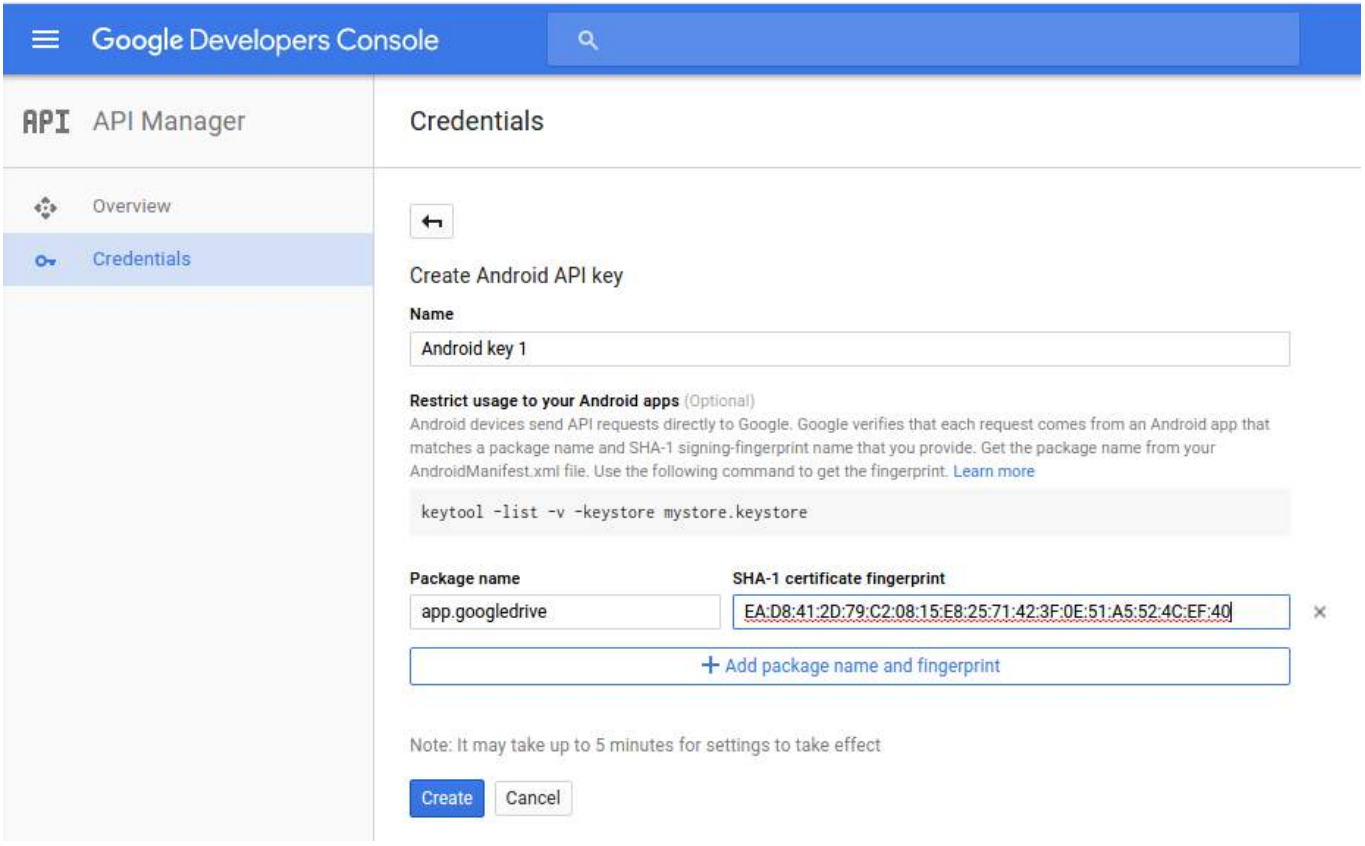
- Next, we need to add Package Name of the Android Project and **SHA-1 fingerprint** in the input fields to create API key.



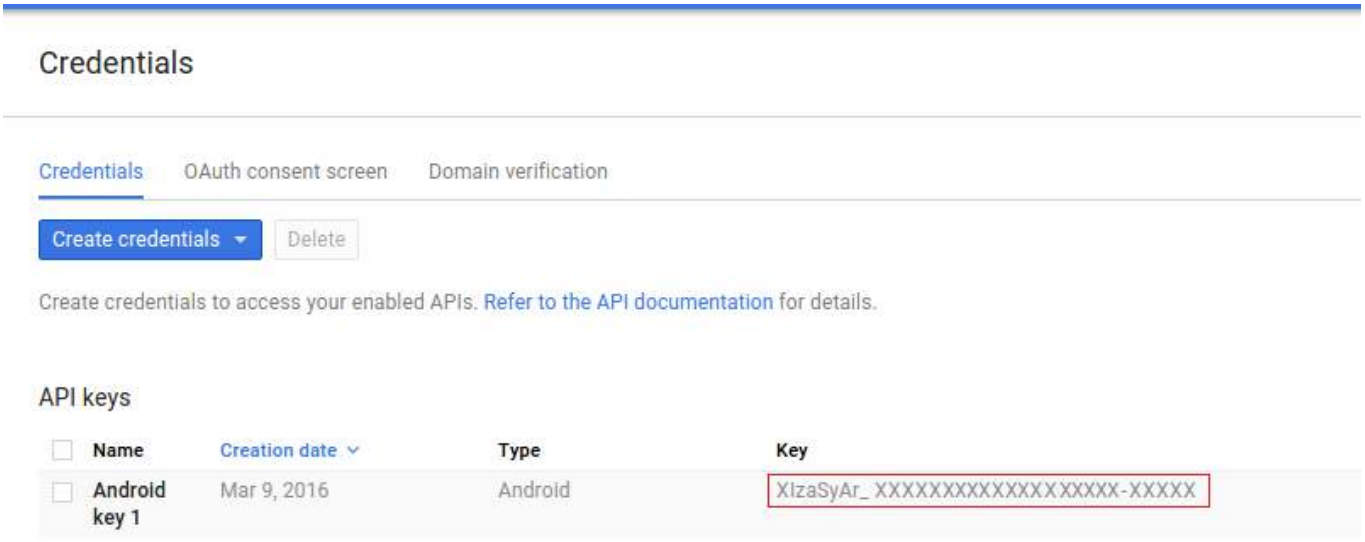
- We need to generate **SHA-1 fingerprint**. So, open your terminal and run **Keytool utility** to get the SHA1 fingerprint. While running Keytool utility, you need to provide **keystore password**. Default development keytool password is "**android**". `keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v`

```
[redacted]@[redacted]:~$ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: 18 Jul, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 3adbdb98
Valid from: Sat Jul 18 09:32:08 IST 2015 until: Mon Jul 10 09:32:08 IST 2045
Certificate fingerprints:
         MD5:  77:C7:A9:6A:30:0F:43:B9:84:E0:61:0F:B2:B6:22:74
         SHA1: EA:D8:41:2D:79:C2:08:15:E8:25:71:42:3F:0E:51:A5:52:4C:EF:40
         SHA256: A2:12:5A:18:E2:F3:FE:8B:93:E8:03:0C:12:3A:52:8D:B5:B0:70:32:C4:
F3:A7:C3:47:F0:9E:B6:8E:AF:33:68
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 8F C7 0C 95 B4 DA 73   6B 67 99 5A A3 C0 05 4A  .......skg.Z...J
0010: 93 BE 25 4F                                        ..%O
]
]
```

- Now, add **Package name** and **SHA-1 fingerprint** in input fields on credentials page. Finally, click on create button to create API key.



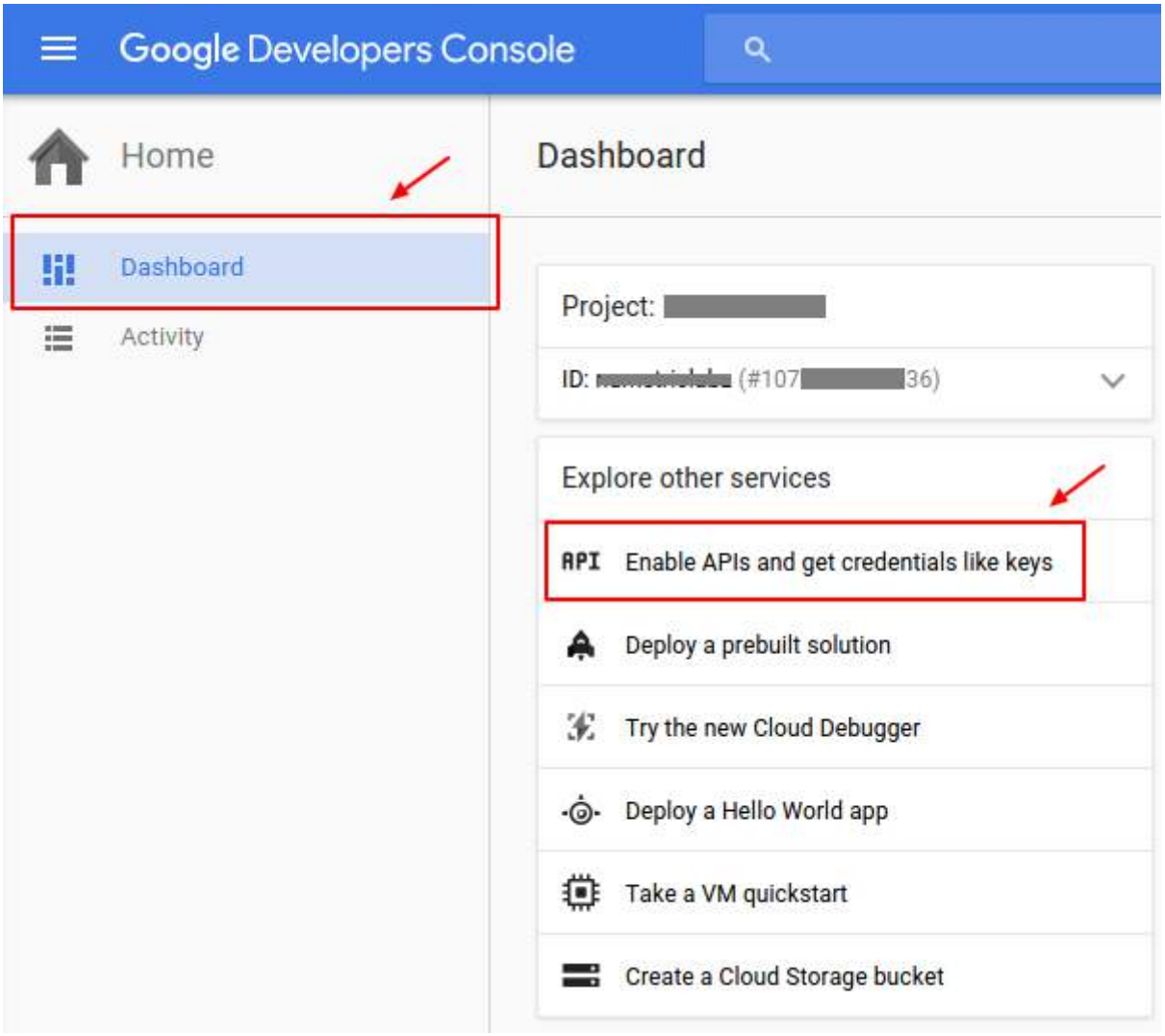
- This will create API key for Android. We will use the this API key to integrate Android application with Google Drive.



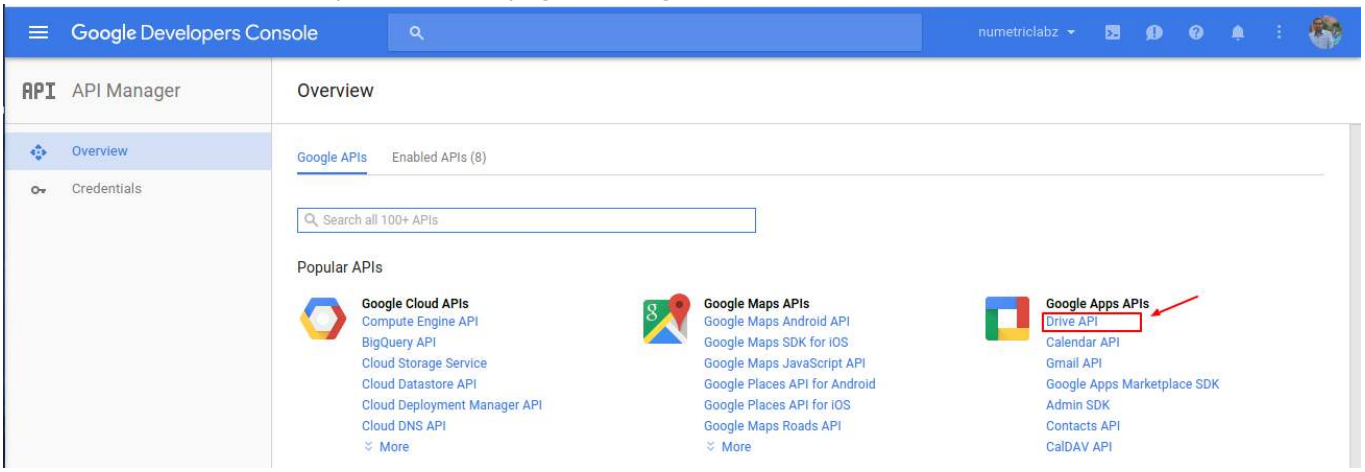
Enable Google Drive API

We need to enable Google Drive Api to access files stored on Google Drive from Android application. To enable Google Drive API, follow below steps:

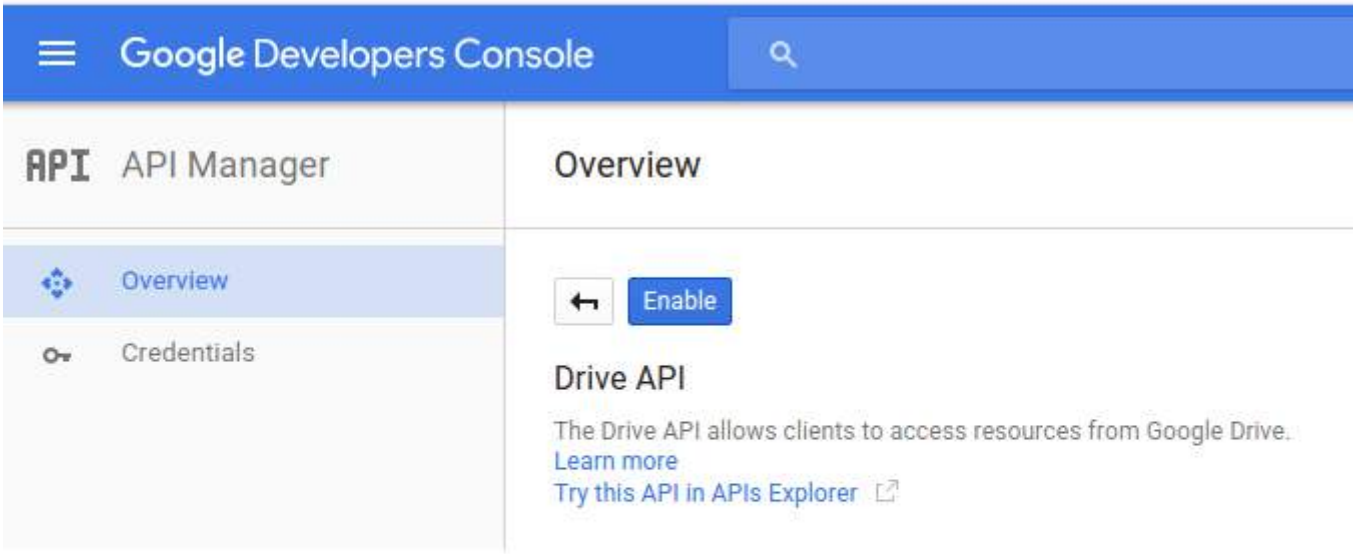
- Go to your Google Developer console Dashboard and click on **Enable APIs get credentials like keys** then you will see popular Google APIs.



- Click on **Drive API** link to open overview page of Google Drive API.



- Click on the Enable button to enable Google drive API. It allows client access to Google Drive.



Add Internet Permission

App needs **Internet** access Google Drive files. Use the following code to set up Internet permissions in AndroidManifest.xml file :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Add Google Play Services

We will use **Google play services API** which includes the **Google Drive Android API**. So, we need to setup Google play services SDK in Android Application. Open your `build.gradle`(app module) file and add Google play services SDK as a dependencies.

```
dependencies {
  ....
    compile 'com.google.android.gms:play-services:<latest_version>'
  ....
}
```

Add API key in Manifest file

To use Google API in Android application, we need to add API key and version of the Google Play Service in the AndroidManifest.xml file. Add the correct metadata tags inside the tag of the AndroidManifest.xml file.

Connect and Authorize the Google Drive Android API

We need to authenticate and connect **Google Drive Android API** with Android application. Authorization of **Google Drive Android API** is handled by the **GoogleApiClient**. We will use **GoogleApiClient** within **onResume()** method.

```
/**
 * Called when the activity will start interacting with the user.
 * At this point your activity is at the top of the activity stack,
```

```
 * with user input going to it.
 */
@Override
protected void onResume() {
   super.onResume();
   if (mGoogleApiClient == null) {

       /**
        * Create the API client and bind it to an instance variable.
        * We use this instance as the callback for connection and connection failures.
        * Since no account name is passed, the user is prompted to choose.
        */
         mGoogleApiClient = new GoogleApiClient.Builder(this)
                 .addApi(Drive.API)
                 .addScope(Drive.SCOPE_FILE)
                 .addConnectionCallbacks(this)
                 .addOnConnectionFailedListener(this)
                 .build();
        }

        mGoogleApiClient.connect();
    }
```

Disconnect Google Deive Android API

When activity stops, we will disconnected Google Drive Android API connection with Android application by calling **disconnect()** method inside activity's **onStop()** method.

```
@Override
protected void onStop() {
    super.onStop();
    if (mGoogleApiClient != null) {

         // disconnect Google Android Drive API connection.
         mGoogleApiClient.disconnect();
    }
    super.onPause();
}
```

Implement Connection Callbacks and Connection Failed Listener

We will implement Connection Callbacks and Connection Failed Listener of Google API client in MainActivity.java file to know status about connection of Google API client. These listeners provide **onConnected(), onConnectionFailed(), onConnectionSuspended()** method to handle the connection issues between app and Drive.

If user has authorized the application, the **onConnected()** method is invoked. If user has not authorized application, **onConnectionFailed()** method is invoked and a dialog is displayed to user that your app is not authorized to access Google Drive. In case connection is suspended, **onConnectionSuspended()** method is called.

You need to implement **ConnectionCallbacks** and **OnConnectionFailedListener** in your activity. Use the following code in your Java file.

```
@Override
    public void onConnectionFailed(ConnectionResult result) {

        // Called whenever the API client fails to connect.
        Log.i(TAG, "GoogleApiClient connection failed:" + result.toString());
```

```
        if (!result.hasResolution()) {

            // show the localized error dialog.
            GoogleApiAvailability.getInstance().getErrorDialog(this, result.getErrorCode(),
0).show();
            return;
        }

        /**
         *  The failure has a resolution. Resolve it.
         *  Called typically when the app is not yet authorized, and an  authorization
         *  dialog is displayed to the user.
         */

        try {

            result.startResolutionForResult(this, REQUEST_CODE_RESOLUTION);

        } catch (SendIntentException e) {

            Log.e(TAG, "Exception while starting resolution activity", e);
        }
    }

   /**
    * It invoked when Google API client connected
    * @param connectionHint
    */
    @Override
    public void onConnected(Bundle connectionHint) {

        Toast.makeText(getApplicationContext(), "Connected", Toast.LENGTH_LONG).show();
    }

   /**
    * It invoked when connection suspended
    * @param cause
    */
    @Override
    public void onConnectionSuspended(int cause) {

        Log.i(TAG, "GoogleApiClient connection suspended");
    }
```

Section 240.2: Create a File on Google Drive

We will add a file on Google Drive. We will use the `createFile()` method of a `Drive` object to create file programmatically on Google Drive. In this example we are adding a new text file in the user's root folder. When a file is added, we need to specify the initial set of metadata, file contents, and the parent folder.

We need to create a `CreateMyFile()` callback method and within this method, use the `Drive` object to retrieve a `DriveContents` resource. Then we pass the API client to the `Drive` object and call the `driveContentsCallback` callback method to handle result of `DriveContents`.

A `DriveContents` resource contains a temporary copy of the file's binary stream which is only available to the application.

```
public void CreateMyFile(){
    fileOperation = true;
```

```
    // Create new contents resource.
    Drive.DriveApi.newDriveContents(mGoogleApiClient)
                  .setResultCallback(driveContentsCallback);
}
```

Result Handler of DriveContents

Handling the response requires to check if the call was successful or not. If the call was successful, we can retrieve the `DriveContents` resource.

We will create a result handler of `DriveContents`. Within this method, we call the `CreateFileOnGoogleDrive()` method and pass the result of `DriveContentsResult`:

```
/**
 * This is the Result result handler of Drive contents.
 * This callback method calls the CreateFileOnGoogleDrive() method.
 */
final ResultCallback<DriveContentsResult> driveContentsCallback =
         new ResultCallback<DriveContentsResult>() {
            @Override
            public void onResult(DriveContentsResult result) {
                if (result.getStatus().isSuccess()) {
                    if (fileOperation == true){
                        CreateFileOnGoogleDrive(result);
                    }
                }
            }
        };
```

Create File Programmatically

To create files, we need to use a `MetadataChangeSet` object. By using this object, we set the title (file name) and file type. Also, we must use the `createFile()` method of the `DriveFolder` class and pass the Google client API, the `MetaDataChangeSet` object, and the `driveContents` to create a file. We call the result handler callback to handle the result of the created file.

We use the following code to create a new text file in the user's root folder:

```
/**
 * Create a file in the root folder using a MetadataChangeSet object.
 * @param result
 */
public void CreateFileOnGoogleDrive(DriveContentsResult result){

    final DriveContents driveContents = result.getDriveContents();

    // Perform I/O off the UI thread.
    new Thread() {
        @Override
        public void run() {
            // Write content to DriveContents.
            OutputStream outputStream = driveContents.getOutputStream();
            Writer writer = new OutputStreamWriter(outputStream);
            try {
                writer.write("Hello Christlin!");
                writer.close();
            } catch (IOException e) {
                Log.e(TAG, e.getMessage());
            }

            MetadataChangeSet changeSet = new MetadataChangeSet.Builder()
```

```
                    .setTitle("My First Drive File")
                    .setMimeType("text/plain")
                    .setStarred(true).build();

            // Create a file in the root folder.
            Drive.DriveApi.getRootFolder(mGoogleApiClient)
                    .createFile(mGoogleApiClient, changeSet, driveContents)
                    setResultCallback(fileCallback);
        }
    }.start();
}
```

Handle result of Created File

The following code will create a callback method to handle the result of the created file:

```
/**
 * Handle result of Created file
 */
final private ResultCallback<DriveFolder.DriveFileResult> fileCallback = new
        ResultCallback<DriveFolder.DriveFileResult>() {
            @Override
            public void onResult(DriveFolder.DriveFileResult result) {
                if (result.getStatus().isSuccess()) {
                    Toast.makeText(getApplicationContext(), "file created: "+
                                result.getDriveFile().getDriveId(), Toast.LENGTH_LONG).show();
                }
                return;
            }
        };
```

Chapter 241: Displaying Google Ads

Section 241.1: Adding Interstitial Ad

Interstitial ads are full-screen ads that cover the interface of their host app. They're typically displayed at natural transition points in the flow of an app, such as between activities or during the pause between levels in a game.

Make sure you have necessary permissions in your `Manifest` file:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

1. Go to your AdMob account.
2. Click on **Monetize** tab.
3. Select or Create the app and choose the platform.
4. Select Interstitial and give an ad unit name.
5. Once the ad unit is created, you can notice the Ad unit ID on the dashboard. For example: ca-app-pub-00000000000/000000000
6. Add dependencies

```
compile 'com.google.firebase:firebase-ads:10.2.1'
```

This one should be on the bottom.

```
apply plugin: 'com.google.gms.google-services'
```

Add your **Ad unit ID** to your `strings.xml` file

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

Add ConfigChanges and meta-data to your manifest:

```
<activity
            android:name="com.google.android.gms.ads.AdActivity"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestS
creenSize"
            android:theme="@android:style/Theme.Translucent" />
```

and

```
<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

Activity:

```
public class AdActivity extends AppCompatActivity {

    private String TAG = AdActivity.class.getSimpleName();
    InterstitialAd mInterstitialAd;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        mInterstitialAd = new InterstitialAd(this);

        // set the ad unit ID
        mInterstitialAd.setAdUnitId(getString(R.string.interstitial_full_screen));

        AdRequest adRequest = new AdRequest.Builder()
                .build();

        // Load ads into Interstitial Ads
        mInterstitialAd.loadAd(adRequest);

        mInterstitialAd.setAdListener(new AdListener() {
            public void onAdLoaded() {
                showInterstitial();
            }
        });
    }

    private void showInterstitial() {
        if (mInterstitialAd.isLoaded()) {
            mInterstitialAd.show();
        }
    }

}
```

This AdActivity will show a full screen ad now.

Section 241.2: Basic Ad Setup

You'll need to add the following to your dependencies:

```
compile 'com.google.firebase:firebase-ads:10.2.1'
```

and then put this in the same file.

```
apply plugin: 'com.google.gms.google-services'
```

Next you'll need to add relevant information into your strings.xml.

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

Next place an adview wherever you want it and style it just like any other view.

```
    <com.google.android.gms.ads.AdView
        android:id="@+id/adView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        ads:adSize="BANNER"
        ads:adUnitId="@string/banner_ad_unit_id">
    </com.google.android.gms.ads.AdView>
```

And last but not least, throw this in your onCreate.

```
MobileAds.initialize(getApplicationContext(), "ca-app-pub-YOUR_ID");
AdView mAdView = (AdView) findViewById(R.id.adView);
    AdRequest adRequest = new AdRequest.Builder().build();
    mAdView.loadAd(adRequest);
```

If you copy-pasted exactly you should now have a small banner ad. Simply place more AdViews wherever you need them for more.

Chapter 242: AdMob

Param	Details
ads:adUnitId="@string/main_screen_ad"	The ID of your ad. Get your ID from the admob site. *"While it's not a requirement, storing your ad unit ID values in a resource file is a good practice. As your app grows and your ad publishing needs mature, it may be necessary to change the ID values. If you keep them in a resource file, you never have to search through your code looking for them."*.[1]

Section 242.1: Implementing

Note: This example requires a valid Admob account and valid Admob ad code.

Build.gradle on app level

Change to the latest version if existing:

```
compile 'com.google.firebase:firebase-ads:10.2.1'
```

Manifest

Internet permission is required to access the ad data. Note that this permission does not have to be requested (using API 23+) as it is a normal permission and not dangerous:

```
<uses-permission android:name="android.permission.INTERNET" />
```

XML

The following XML example shows a banner ad:

```
<com.google.android.gms.ads.AdView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/adView"
    ads:adSize="BANNER"
    ads:adUnitId="@string/main_screen_ad" />
```

For the code of other types, refer to the Google AdMob Help.

Java

The following code is for the integration of banner ads. Note that other ad types may require different integration:

```
// Alternative for faster initialization.
// MobileAds.initialize(getApplicationContext(), "AD_UNIT_ID");

AdView mAdView = (AdView) findViewById(R.id.adView);
// Add your device test ID if you are doing testing before releasing.
// The device test ID can be found in the admob stacktrace.
AdRequest adRequest = new AdRequest.Builder().build();
mAdView.loadAd(adRequest);
```

Add the `AdView` life cycle methods in the `onResume()`, `onPause()`, and `onDestroy()` methods of your activity:

```
@Override
public void onPause() {
    if (mAdView != null) {
        mAdView.pause();
```

```
    }
    super.onPause();
}

@Override
public void onResume() {
    super.onResume();
    if (mAdView != null) {
        mAdView.resume();
    }
}

@Override
public void onDestroy() {
    if (mAdView != null) {
        mAdView.destroy();
    }
    super.onDestroy();
}
```

Chapter 243: Google Play Store

Section 243.1: Open Google Play Store Listing for your app

The following code snippet shows how to open the Google Play Store Listing of your app in a safe way. Usually you want to use it when asking the user to leave a review for your app.

```
private void openPlayStore() {
    String packageName = getPackageName();
    Intent playStoreIntent = new Intent(Intent.ACTION_VIEW,
            Uri.parse("market://details?id=" + packageName));
    setFlags(playStoreIntent);
    try {
        startActivity(playStoreIntent);
    } catch (Exception e) {
        Intent webIntent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("https://play.google.com/store/apps/details?id=" + packageName));
        setFlags(webIntent);
        startActivity(webIntent);
    }
}

@SuppressWarnings("deprecation")
private void setFlags(Intent intent) {
    intent.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
    else
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
}
```

Note: The code opens the Google Play Store if the app is installed. Otherwise it will just open the web browser.

Section 243.2: Open Google Play Store with the list of all applications from your publisher account

You can add a "Browse Our Other Apps" button in your app, listing all your(publisher) applications in the Google Play Store app.

```
String urlApp = "market://search?q=pub:Google+Inc.";
String urlWeb = "http://play.google.com/store/search?q=pub:Google+Inc.";
try {
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlApp));
    setFlags(i);
    startActivity(i);
} catch (android.content.ActivityNotFoundException anfe) {
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlWeb)));
    setFlags(i);
    startActivity(i);
}


@SuppressWarnings("deprecation")
public void setFlags(Intent i) {
    i.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)  {
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
    }
```

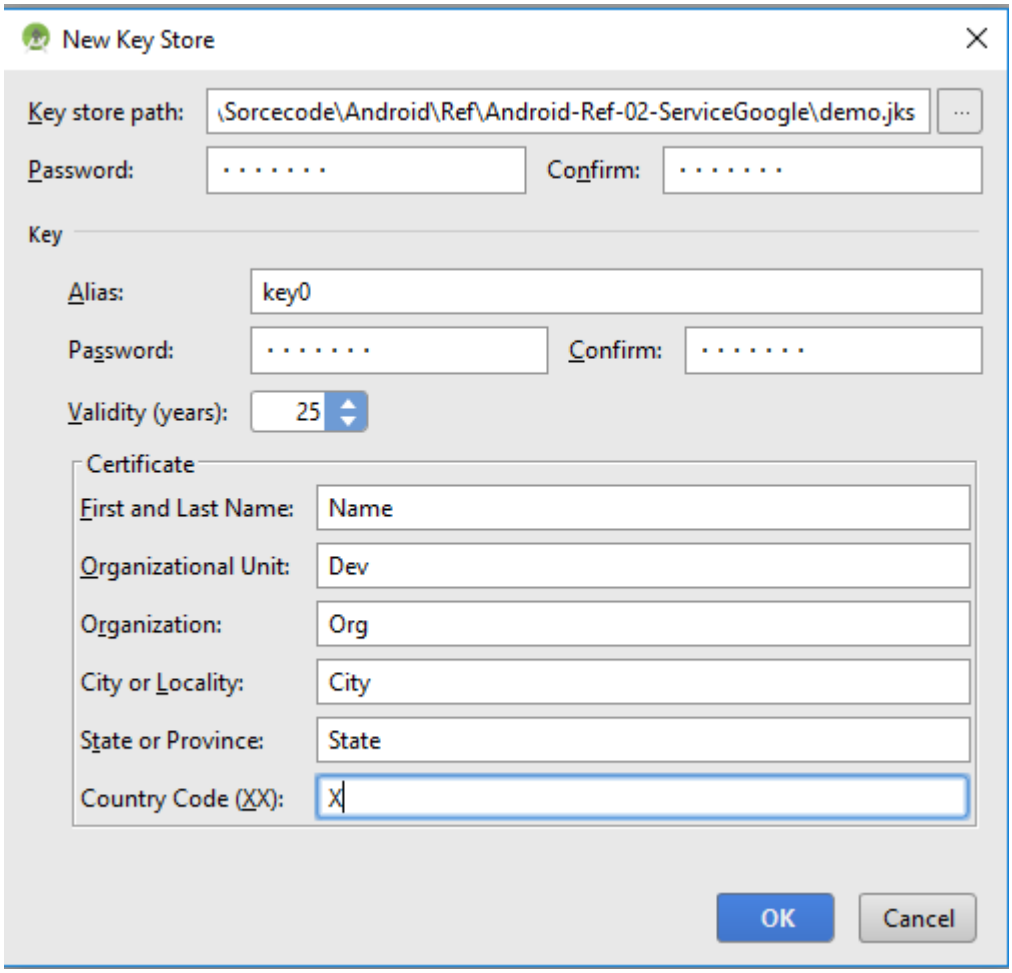
```
    else  {
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
    }
}
```

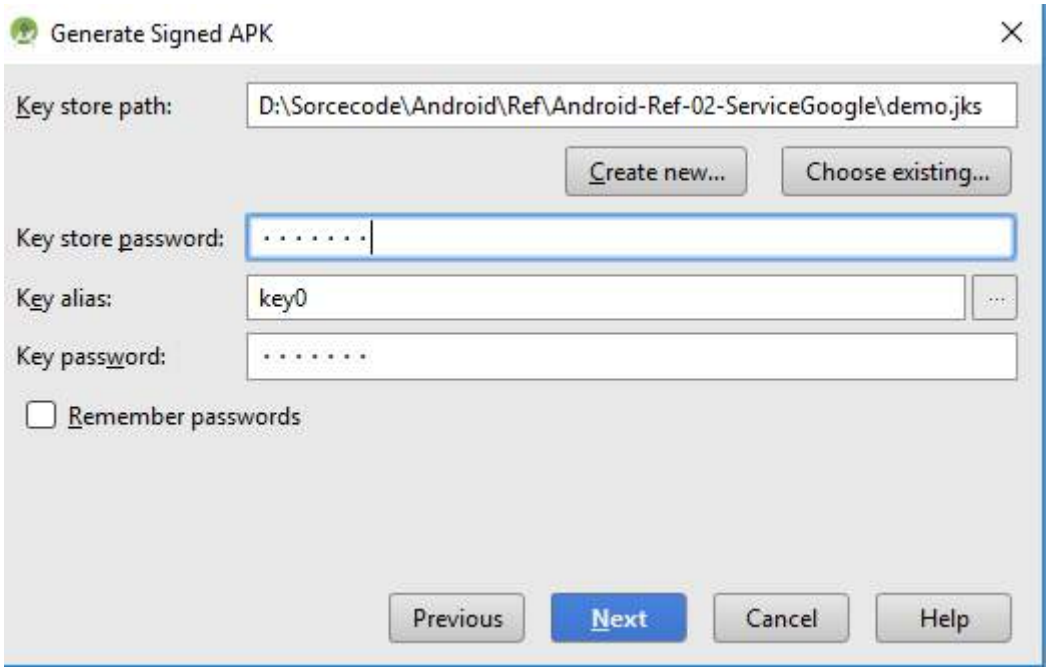
Chapter 244: Sign your Android App for Release

Android requires that all APKs be signed for release.

Section 244.1: Sign your App

1. In the menu bar, click Build > Generate Signed APK.
2. Select the module you would like to release from the drop down and click Next.
3. To Create a new keystore, click Create new. Now fill the required information and press ok in New Key Store.





4. On the Generate Signed APK Wizard fields are already populated for you if you just created new key store otherwise fill it and click next.
5. On the next window, select a destination for the signed APK, select the build type and click finish.

Section 244.2: Configure the build.gradle with signing configuration

You can define the signing configuration to sign the apk in the `build.gradle` file.

You can define:

- `storeFile` : the keystore file
- `storePassword`: the keystore password
- `keyAlias`: a key alias name
- `keyPassword`: A key alias password

You have to **define** the `signingConfigs` block to create a signing configuration:

```
android {
    signingConfigs {

        myConfig {
            storeFile file("myFile.keystore")
            storePassword "xxxx"
            keyAlias "xxxx"
            keyPassword "xxxx"
        }
    }
    //....
}
```

Then you can **assign** it to one or more build types.

```
android {

    buildTypes {
```

```
        release {
            signingConfig signingConfigs.myConfig
        }
    }
}
```

Chapter 245: TensorFlow

TensorFlow was designed with mobile and embedded platforms in mind. We have sample code and build support you can try now for these platforms:

Android iOS Raspberry Pi

Section 245.1: How to use

Install Bazel from here. Bazel is the primary build system for TensorFlow. Now, edit the WORKSPACE, we can find the WORKSPACE file in the root directory of the TensorFlow that we have cloned earlier.

```
# Uncomment and update the paths in these entries to build the Android demo.
#android_sdk_repository(
#    name = "androidsdk",
#    api_level = 23,
#    build_tools_version = "25.0.1",
#    # Replace with path to Android SDK on your system
#    path = "<PATH_TO_SDK>",
#)
#
#android_ndk_repository(
#    name="androidndk",
#    path="<PATH_TO_NDK>",
#    api_level=14)
```

Like below with our sdk and ndk path:

```
android_sdk_repository(
    name = "androidsdk",
    api_level = 23,
    build_tools_version = "25.0.1",
    # Replace with path to Android SDK on your system
    path = "/Users/amitshekhar/Library/Android/sdk/",
)
android_ndk_repository(
    name="androidndk",
    path="/Users/amitshekhar/Downloads/android-ndk-r13/",
    api_level=14)
```

Chapter 246: Android Vk Sdk

Section 246.1: Initialization and login

1. Create a new application here: create application
2. Choose standalone applicaton and confirm app creation via SMS.
3. Fill **Package namefor Android** as your current package name. *You can get your package name inside android manifest file, at the very begginning.*
4. Get your **Certificate fingerprint** by executing this command in your shell/cmd:

```
keytool -exportcert -alias androiddebugkey -keystore path-to-debug-or-production-keystore -list -v
```

You can also get this fingerprint by SDK itself:

```
String[] fingerprints = VKUtil.getCertificateFingerprint(this, this.getPackageName());
Log.d("MainActivity", fingerprints[0]);
```

5. Add recieved fingerprint into your **Signing certificate fingerprint for Android:** field in Vk app settings (where you entered your package name)

6. Then add this to your gradle file:

```
compile 'com.vk:androidsdk:1.6.5'
```

8. Initialize the SDK on startup using the following method. The best way is to call it in the Applications onCreate method.

```
private static final int VK_ID = your_vk_id;
public static final String VK_API_VERSION = "5.52"; //current version
@Override
    public void onCreate() {
        super.onCreate();
        VKSdk.customInitialize(this, VK_ID, VK_API_VERSION);
}
```

This is the best way to initizlize VKSdk. Don't use the methid where VK_ID should be placed inside strings.xml because api will not work correctly after it.

9. Final step is to login using vksdk.

```
    public static final String[] VK_SCOPES = new String[]{
            VKScope.FRIENDS,
            VKScope.MESSAGES,
            VKScope.NOTIFICATIONS,
            VKScope.OFFLINE,
            VKScope.STATUS,
            VKScope.STATS,
            VKScope.PHOTOS
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        someButtonForLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                VKSdk.login(this, VK_SCOPES);
            }
        });

    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        VKSdk.onActivityResult(requestCode, resultCode, data, new VKCallback<VKAccessToken>() {
            @Override
            public void onResult(VKAccessToken res) {
                res.accessToken; //getting our token here.
            }

            @Override
            public void onError(VKError error) {
                Toast.makeText(SocialNetworkChooseActivity.this,
                        "User didn't pass Authorization", Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Chapter 247: Project SDK versions

Parameter	Details
SDK Version	The SDK version for each field is the Android release's SDK API level integer. For example, Froyo (Android 2.2) corresponds to API level 8. These integers are also defined in `Build.VERSION_CODES`.

An Android application needs to run on all kinds of devices. Each device may have a different version on Android running on it.

Now, each Android version might not support all the features that your app requires, and so while building an app, you need to keep the minimum and maximum Android version in mind.

Section 247.1: Defining project SDK versions

In your `build.gradle` file of main module(**app**), define your minimum and target version number.

```
android {
    //the version of sdk source used to compile your project
    compileSdkVersion 23

    defaultConfig {
        //the minimum sdk version required by device to run your app
        minSdkVersion 19
        //you normally don't need to set max sdk limit so that your app can support future versions
of android without updating app
        //maxSdkVersion 23
        //
        //the latest sdk version of android on which you are targeting(building and testing) your
app, it should be same as compileSdkVersion
        targetSdkVersion 23
    }
}
```

Chapter 248: Facebook SDK for Android

Parameter	Details
TAG	A String used while logging
FacebookSignInHelper	A static reference to facebook helper
CallbackManager	A callback for facebook operations
Activity	A context
PERMISSION_LOGIN	An array that contains all permission required from facebook to login.
loginCallback	A callback for facebook login

Section 248.1: How to add Facebook Login in Android

Add below dependencies to your `build.gradle`

```
  // Facebook login
    compile 'com.facebook.android:facebook-android-sdk:4.21.1'
```

Add below helper class to your utility package:

```
/**
 * Created by Andy
 * An utility for Facebook
 */
public class FacebookSignInHelper {
    private static final String TAG = FacebookSignInHelper.class.getSimpleName();
    private static FacebookSignInHelper facebookSignInHelper = null;
    private CallbackManager callbackManager;
    private Activity mActivity;
    private static final Collection<String> PERMISSION_LOGIN = (Collection<String>)
Arrays.asList("public_profile", "user_friends","email");
    private FacebookCallback<LoginResult> loginCallback;



    public static FacebookSignInHelper newInstance(Activity context) {
        if (facebookSignInHelper == null)
            facebookSignInHelper = new FacebookSignInHelper(context);
        return facebookSignInHelper;
    }


    public FacebookSignInHelper(Activity mActivity) {
        try {
            this.mActivity = mActivity;
            // Initialize the SDK before executing any other operations,
            // especially, if you're using Facebook UI elements.
            FacebookSdk.sdkInitialize(this.mActivity);
            callbackManager = CallbackManager.Factory.create();
            loginCallback = new FacebookCallback<LoginResult>() {
                @Override
                public void onSuccess(LoginResult loginResult) {
                   // You are logged into Facebook
                }

                @Override
                public void onCancel() {
                    Log.d(TAG, "Facebook: Cancelled by user");
```

```
            }

            @Override
            public void onError(FacebookException error) {
                Log.d(TAG, "FacebookException: " + error.getMessage());
            }
        };
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * To login user on facebook without default Facebook button
 */
public void loginUser() {
    try {
        LoginManager.getInstance().registerCallback(callbackManager, loginCallback);
        LoginManager.getInstance().logInWithReadPermissions(this.mActivity, PERMISSION_LOGIN);
    } catch (Exception e) {
        e.printStackTrace();
    }
}


/**
 * To log out user from facebook
 */
public void signOut() {
    // Facebook sign out
    LoginManager.getInstance().logOut();
}

public CallbackManager getCallbackManager() {
    return callbackManager;
}

public FacebookCallback<LoginResult> getLoginCallback() {
    return loginCallback;
}

/**
 * Attempts to log debug key hash for facebook
 *
 * @param context : A reference to context
 * @return : A facebook debug key hash
 */
public static String getKeyHash(Context context) {
    String keyHash = null;
    try {
        PackageInfo info = context.getPackageManager().getPackageInfo(
                context.getPackageName(),
                PackageManager.GET_SIGNATURES);
        for (Signature signature : info.signatures) {
            MessageDigest md = MessageDigest.getInstance("SHA");
            md.update(signature.toByteArray());
            keyHash = Base64.encodeToString(md.digest(), Base64.DEFAULT);
            Log.d(TAG, "KeyHash:" + keyHash);
        }
    } catch (PackageManager.NameNotFoundException e) {
        e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
```

```
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return keyHash;
    }
}
```

Add below code in Your Activity:

```
FacebookSignInHelper facebookSignInHelper = FacebookSignInHelper.newInstance(LoginActivity.this,
fireBaseAuthHelper);
facebookSignInHelper.loginUser();
```

Add below code to your `OnActivityResult`:

```
facebookSignInHelper.getCallbackManager().onActivityResult(requestCode, resultCode, data);
```

Section 248.2: Create your own custom button for Facebook login

Once you first add the Facebook login/signup, the button looks something like:

Most of the times, it doesn't match with the design-specs of your app. And here's how you can customize it:

```
<FrameLayout
   android:layout_below="@+id/no_network_bar"
   android:id="@+id/FrameLayout1"
   android:layout_width="match_parent"
   android:layout_height="wrap_content">

    <com.facebook.login.widget.LoginButton
       android:id="@+id/login_button"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:visibility="gone" />

    <Button
       android:background="#3B5998"
       android:layout_width="match_parent"
       android:layout_height="60dp"
       android:id="@+id/fb"
       android:onClick="onClickFacebookButton"
       android:textAllCaps="false"
       android:text="Sign up with Facebook"
       android:textSize="22sp"
       android:textColor="#ffffff" />
</FrameLayout>
```

Just wrap the original `com.facebook.login.widget.LoginButton` into a `FrameLayout` and make its visibility gone.

Next, add your custom button in the same `FrameLayout`. I've added some sample specs. You can always make your own drawable background for the facebook button and set it as the background of the button.

The final thing we do is simply convert the click on my custom button to a click on the facecbook button:

```
//The original Facebook button
LoginButton loginButton = (LoginButton)findViewById(R.id.login_button);

//Our custom Facebook button
fb = (Button) findViewById(R.id.fb);

public void onClickFacebookButton(View view) {
    if (view == fb) {
        loginButton.performClick();
    }
}
```

Great! Now the button looks something like this:

Section 248.3: A minimalistic guide to Facebook login/signup implementation

1. You have to setup the prerequisites.
2. Add the Facebook activity to the *AndroidManifest.xml* file:

```
<activity
   android:name="com.facebook.FacebookActivity"
   android:configChanges= "keyboard|keyboardHidden|screenLayout|screenSize|orientation"
   android:theme="@android:style/Theme.Translucent.NoTitleBar"
   android:label="@string/app_name" />
```

3. Add the login button to your layout XML file:

```
<com.facebook.login.widget.LoginButton
    android:id="@+id/login_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

4. Now you have the Facebook button. If the user clicks on it, the Facebook login dialog will come up on top of the app's screen. Here the user can fill in their credentials and press the *Log In* button. If the credentials are correct, the dialog grants the corresponding permissions and a callback is sent to your original activity containing the button. The following code shows how you can receive that callback:

```
loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
    @Override
    public void onSuccess(LoginResult loginResult) {
        // Completed without error. You might want to use the retrieved data here.
    }

    @Override
    public void onCancel() {
        // The user either cancelled the Facebook login process or didn't authorize the app.
    }

    @Override
```

```
    public void onError(FacebookException exception) {
        // The dialog was closed with an error. The exception will help you recognize what
exactly went wrong.
    }
});
```

Section 248.4: Setting permissions to access data from the Facebook profile

If you want to retrieve the details of a user's Facebook profile, you need to set permissions for the same:

```
loginButton = (LoginButton)findViewById(R.id.login_button);

loginButton.setReadPermissions(Arrays.asList("email", "user_about_me"));
```

You can keep adding more permissions like friends-list, posts, photos etc. Just pick the right permission and add it the above list.

Note: You don't need to set any explicit permissions for accessing the public profile (first name, last name, id, gender etc).

Section 248.5: Logging out of Facebook

Facebook SDK 4.0 onwards, this is how we logout:

```
com.facebook.login.LoginManager.getInstance().logOut();
```

For versions before 4.0, the logging out is gone by explicitly clearing the access token:

```
Session session = Session.getActiveSession();
session.closeAndClearTokenInformation();
```

Chapter 249: Thread

Section 249.1: Thread Example with its description

While launching an application firstly main thread is executed. This Main thread handles all the UI concept of application. If we want to run long the task in which we don't need the UI then we use thread for running that task in background.

Here is the example of Thread which describes blow:

```
new Thread(new Runnable() {
    public void run() {
        for(int i = 1; i < 5;i++) {
            System.out.println(i);
        }
    }
}).start();
```

We can create thread by creating the object of Thread which have `Thread.run()` method for running the thread.Here, `run()` method is called by the `start()` method.

We can also run the the multiple threads independently, which is known as MultiThreading. This thread also have the functionality of sleep by which the currently executing thread to sleep (temporarily cease execution) for the specified number of time. But sleep throws the InterruptedException So, we have to handle it by using try/catch like this.

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

Section 249.2: Updating the UI from a Background Thread

It is common to use a background Thread for doing network operations or long running tasks, and then update the UI with the results when needed.

This poses a problem, as only the main thread can update the UI.

The solution is to use the `runOnUiThread()` method, as it allows you to initiate code execution on the UI thread from a background Thread.

In this simple example, a Thread is started when the Activity is created, runs until the magic number of `42` is randomly generated, and then uses the `runOnUiThread()` method to update the UI once this condition is met.

```
public class MainActivity extends AppCompatActivity {

    TextView mTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mTextView = (TextView) findViewById(R.id.my_text_view);

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (true) {
```

```
                    //do stuff....
                    Random r = new Random();
                    if (r.nextInt(100) == 42) {
                       break;
                    }
                }

                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        mTextView.setText("Ready Player One");
                    }
                });
            }
        }).start();
    }
}
```

Chapter 250: AsyncTask

Parameter	Details
Params	the type of the parameters sent to the task upon execution.
Progress	the type of the progress units published during the background computation
Result	the type of the result of the background computation.

Section 250.1: Basic Usage

In Android Activities and Services, most callbacks are run on the main thread. This makes it simple to update the UI, but running processor- or I/O-heavy tasks on the main thread can cause your UI to pause and become unresponsive (official documentation on what then happens).

You can remedy this by putting these heavier tasks on a background thread.

One way to do this is using an AsyncTask, which provides a framework to facilitate easy usage of a background Thread, and also perform UI Thread tasks before, during, and after the background Thread has completed its work.

Methods that can be overridden when extending `AsyncTask`:

- `onPreExecute()` : invoked on the **UI thread** before the task is executed
- `doInBackground()`: invoked on **the background thread** immediately after `onPreExecute()` finishes executing.
- `onProgressUpdate()`: invoked on the **UI thread** after a call to `publishProgress(Progress...)`.
- `onPostExecute()`: invoked on the **UI thread** after the background computation finishes

Example

```
public class MyCustomAsyncTask extends AsyncTask<File, Void, String> {


    @Override
    protected void onPreExecute(){
        // This runs on the UI thread before the background thread executes.
        super.onPreExecute();
        // Do pre-thread tasks such as initializing variables.
        Log.v("myBackgroundTask", "Starting Background Task");
    }

    @Override
    protected String doInBackground(File... params) {
        // Disk-intensive work. This runs on a background thread.
        // Search through a file for the first line that contains "Hello", and return
        // that line.
        try (Scanner scanner = new Scanner(params[0])) {
            while (scanner.hasNextLine()) {
                final String line = scanner.nextLine();
                publishProgress(); // tell the UI thread we made progress

                if (line.contains("Hello")) {
                    return line;
                }
            }
            return null;
        }
    }
```

```
    @Override
    protected void onProgressUpdate(Void...p) {
        // Runs on the UI thread after publishProgress is invoked
        Log.v("Read another line!")
    }

    @Override
    protected void onPostExecute(String s) {
        // This runs on the UI thread after complete execution of the doInBackground() method
        // This function receives result(String s) returned from the doInBackground() method.
        // Update UI with the found string.
        TextView view = (TextView) findViewById(R.id.found_string);
        if (s != null) {
            view.setText(s);
        } else {
            view.setText("Match not found.");
        }
    }

}
```

Usage:

```
MyCustomAsyncTask asyncTask = new MyCustomAsyncTask<File, Void, String>();
// Run the task with a user supplied filename.
asyncTask.execute(userSuppliedFilename);
```

or simply:

```
new MyCustomAsyncTask().execute(userSuppliedFilename);
```

Note

When defining an `AsyncTask` we can pass three types between `< >` brackets.
Defined as **`<Params, Progress, Result>`** (see **Parameters section**)

In the previous example we've used types **`<File, Void, String>`**:

```
AsyncTask<File, Void, String>
// Params has type File
// Progress has unused type
// Result has type String
```

`Void` is used when you want to mark a type as unused.

Note that you can't pass primitive types (i.e. **`int`**, **`float`** and 6 others) as parameters. In such cases, you should pass their wrapper classes, e.g. `Integer` instead of **`int`**, or `Float` instead of **`float`**.

The AsyncTask and Activity life cycle

AsyncTasks don't follow Activity instances' life cycle. If you start an AsyncTask inside an Activity and you rotate the device, the Activity will be destroyed and a new instance will be created. But the AsyncTask will not die. It will go on living until it completes.

Solution: AsyncTaskLoader

One subclass of Loaders is the AsyncTaskLoader. This class performs the same function as the AsyncTask, but much better. It can handle Activity configuration changes more easily, and it behaves within the life cycles of Fragments and Activities. The nice thing is that the AsyncTaskLoader can be used in any situation that the AsyncTask is being used. Anytime data needs to be loaded into memory for the Activity/Fragment to handle, The AsyncTaskLoader can

do the job better.

Section 250.2: Pass Activity as WeakReference to avoid memory leaks

It is common for an AsyncTask to require a reference to the Activity that called it.

If the AsyncTask is an inner class of the Activity, then you can reference it and any member variables/methods directly.

If, however, the AsyncTask is not an inner class of the Activity, you will need to pass an Activity reference to the AsyncTask. When you do this, one potential problem that may occur is that the AsyncTask will keep the reference of the Activity until the AsyncTask has completed its work in its background thread. If the Activity is finished or killed before the AsyncTask's background thread work is done, the AsyncTask will still have its reference to the Activity, and therefore it cannot be garbage collected.

As a result, this will cause a memory leak.

In order to prevent this from happening, make use of a WeakReference in the AsyncTask instead of having a direct reference to the Activity.

Here is an example AsyncTask that utilizes a WeakReference:

```
private class MyAsyncTask extends AsyncTask<String, Void, Void> {

    private WeakReference<Activity> mActivity;

    public MyAsyncTask(Activity activity) {
        mActivity = new WeakReference<Activity>(activity);
    }

    @Override
    protected void onPreExecute() {
        final Activity activity = mActivity.get();
        if (activity != null) {
            ....
        }
    }

    @Override
    protected Void doInBackground(String... params) {
        //Do something
        String param1 = params[0];
        String param2 = params[1];
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        final Activity activity = mActivity.get();
        if (activity != null) {
            activity.updateUI();
        }
    }
}
```

Calling the AsyncTask from an Activity:

```
new MyAsyncTask(this).execute("param1", "param2");
```

Calling the AsyncTask from a Fragment:

```
new MyAsyncTask(getActivity()).execute("param1", "param2");
```

Section 250.3: Download Image using AsyncTask in Android

This tutorial explains how to download Image using AsyncTask in Android. The example below download image while showing progress bar while during download. **Understanding Android AsyncTask**
Async task enables you to implement MultiThreading without get Hands dirty into threads. AsyncTask enables proper and easy use of the UI thread. It allows performing background operations and passing the results on the UI thread. If you are doing something isolated related to UI, for example downloading data to present in a list, go ahead and use AsyncTask.

- AsyncTasks should ideally be used for short operations (a few seconds at the most.)
- An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called `onPreExecute()`, `doInBackground()`, `onProgressUpdate()` and `onPostExecute()`.
- In `onPreExecute()` you can define code, which need to be executed before background processing starts.
- doInBackground have code which needs to be executed in background, here in `doInBackground()` we can send results to multiple times to event thread by publishProgress() method, to notify background processing has been completed we can return results simply.
- `onProgressUpdate()` method receives progress updates from `doInBackground()` method, which is published via `publishProgress()` method, and this method can use this progress update to update event thread
- `onPostExecute()` method handles results returned by `doInBackground()` method.
- The generic types used are
    - Params, the type of the parameters sent to the task upon execution
    - Progress, the type of the progress units published during the background computation.
    - Result, the type of the result of the background computation.
- If an async task not using any types, then it can be marked as Void type.
- An running async task can be cancelled by calling `cancel(boolean)` method.

Downloading image using Android AsyncTask

your .xml layout

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

<Button
   android:id="@+id/downloadButton"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="Click Here to Download" />

<ImageView
   android:id="@+id/imageView"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:contentDescription="Your image will appear here" />
```

```
</LinearLayout>
```

.java class

```
package com.javatechig.droid;

import java.io.InputStream;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import android.app.Activity;
import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;

public class ImageDownladerActivity extends Activity {

    private ImageView downloadedImg;
    private ProgressDialog simpleWaitDialog;
    private String downloadUrl = "http://www.9ori.com/store/media/images/8ab579a656.jpg";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.asynch);
        Button imageDownloaderBtn = (Button) findViewById(R.id.downloadButton);

        downloadedImg = (ImageView) findViewById(R.id.imageView);

        imageDownloaderBtn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                new ImageDownloader().execute(downloadUrl);
            }

        });
    }

    private class ImageDownloader extends AsyncTask {

        @Override
        protected Bitmap doInBackground(String... param) {
            // TODO Auto-generated method stub
            return downloadBitmap(param[0]);
        }

        @Override
        protected void onPreExecute() {
            Log.i("Async-Example", "onPreExecute Called");
            simpleWaitDialog = ProgressDialog.show(ImageDownladerActivity.this,
```

```
                    "Wait", "Downloading Image");

        }

        @Override
        protected void onPostExecute(Bitmap result) {
            Log.i("Async-Example", "onPostExecute Called");
            downloadedImg.setImageBitmap(result);
            simpleWaitDialog.dismiss();

        }

        private Bitmap downloadBitmap(String url) {
            // initilize the default HTTP client object
            final DefaultHttpClient client = new DefaultHttpClient();

            //forming a HttpGet request
            final HttpGet getRequest = new HttpGet(url);
            try {

                HttpResponse response = client.execute(getRequest);

                //check 200 OK for success
                final int statusCode = response.getStatusLine().getStatusCode();

                if (statusCode != HttpStatus.SC_OK) {
                    Log.w("ImageDownloader", "Error " + statusCode +
                            " while retrieving bitmap from " + url);
                    return null;

                }

                final HttpEntity entity = response.getEntity();
                if (entity != null) {
                    InputStream inputStream = null;
                    try {
                        // getting contents from the stream
                        inputStream = entity.getContent();

                        // decoding stream data back into image Bitmap that android understands
                        final Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

                        return bitmap;
                    } finally {
                        if (inputStream != null) {
                            inputStream.close();
                        }
                        entity.consumeContent();
                    }
                }
            } catch (Exception e) {
                // You Could provide a more explicit error message for IOException
                getRequest.abort();
                Log.e("ImageDownloader", "Something went wrong while" +
                        " retrieving bitmap from " + url + e.toString());
            }

            return null;
        }
    }
}
```

Since there is currently no comment field for examples (or I haven't found it or I haven't permission for it) here is some comment about this:

This is a good example what can be done with AsyncTask.

However the example currently has problems with

- possible memory leaks
- app crash if there was a screen rotation shortly before the async task finished.

For details see:

- Pass Activity as WeakReference to avoid memory leaks
- http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-with-inner-async-tasks
- Avoid leaking Activities with AsyncTask

Section 250.4: Canceling AsyncTask

```
YourAsyncTask task = new YourAsyncTask();
task.execute();
task.cancel();
```

This doesn't stop your task if it was in progress, it just sets the cancelled flag which can be checked by checking the return value of `isCancelled()` (assuming your code is currently running) by doing this:

```
class YourAsyncTask extends AsyncTask<Void, Void, Void> {
    @Override
    protected Void doInBackground(Void... params) {
        while(!isCancelled()) {
            ... doing long task stuff
            //Do something, you need, upload part of file, for example
            if (isCancelled()) {
                return null; // Task was detected as canceled
            }
            if (yourTaskCompleted) {
                return null;
            }
        }
    }
}
```

Note

If an AsyncTask is canceled while `doInBackground(Params... params)` is still executing then the method `onPostExecute(Result result)` will **NOT** be called after `doInBackground(Params... params)` returns. The AsyncTask will instead call the `onCancelled(Result result)` to indicate that the task was cancelled during execution.

Section 250.5: AsyncTask: Serial Execution and Parallel Execution of Task

AsyncTask is an abstract Class and does not inherit the `Thread` class. It has an **abstract** method `doInBackground(Params... params)`, which is overridden to perform the task. This method is called from `AsyncTask.call()`.

Executor are part of `java.util.concurrent` package.

Moreover, AsyncTask contains 2 `Executors`

`THREAD_POOL_EXECUTOR`

It uses worker threads to execute the tasks parallelly.

```
public static final Executor THREAD_POOL_EXECUTOR = new ThreadPoolExecutor(CORE_POOL_SIZE,
MAXIMUM_POOL_SIZE, KEEP_ALIVE, TimeUnit.SECONDS, sPoolWorkQueue, sThreadFactory);
```

`SERIAL_EXECUTOR`

It executes the task serially, i.e. one by one.

```
private static class SerialExecutor implements Executor { }
```

Both `Executors` are **static**, hence only one `THREAD_POOL_EXECUTOR` and one `SerialExecutor` objects exist, but you can create several `AsyncTask` objects.

Therefore, if you try to do multiple background task with the default Executor (`SerialExecutor`), these task will be queue and executed serially.

If you try to do multiple background task with `THREAD_POOL_EXECUTOR`, then they will be executed parallelly.

Example:

```
public class MainActivity extends Activity {
    private Button bt;
    private int CountTask = 0;
    private static final String TAG = "AsyncTaskExample";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bt = (Button) findViewById(R.id.button);
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                BackgroundTask backgroundTask = new BackgroundTask ();
                Integer data[] = { ++CountTask, null, null };

                // Task Executed in thread pool ( 1 )
                backgroundTask.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, data);

                // Task executed Serially ( 2 )
                // Uncomment the below code and comment the above code of Thread
                // pool Executor and check
                // backgroundTask.execute(data);
                Log.d(TAG, "Task = " + (int) CountTask + " Task Queued");

            }
        });

    }

    private class BackgroundTask extends AsyncTask<Integer, Integer, Integer> {
        int taskNumber;
```

```
        @Override
        protected Integer doInBackground(Integer... integers) {
            taskNumber = integers[0];

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            Log.d(TAG, "Task = " + taskNumber + " Task Running in Background");

            publishProgress(taskNumber);
            return null;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected void onPostExecute(Integer aLong) {
            super.onPostExecute(aLong);
        }

        @Override
        protected void onProgressUpdate(Integer... values) {
            super.onProgressUpdate(values);
            Log.d(TAG, "Task = " + (int) values[0]
                    + " Task Execution Completed");
        }
    }
}
```

Perform Click on button several times to start a task and see the result.

Task Executed in thread pool(1)

Each task takes 1000 ms to complete.

At t=36s, tasks 2, 3 and 4 are queued and started executing also because they are executing parallelly.

```
08-02 19:48:35.815: D/AsyncTaskExample(11693): Task = 1 Task Queued
08-02 19:48:35.815: D/AsyncTaskExample(11693): Task = 1 Task Running in Background
08-02 19:48:**36.025**: D/AsyncTaskExample(11693): Task = 2 Task Queued
08-02 19:48:**36.025**: D/AsyncTaskExample(11693): Task = 2 Task Running in Background
08-02 19:48:**36.165**: D/AsyncTaskExample(11693): Task = 3 Task Queued
08-02 19:48:**36.165**: D/AsyncTaskExample(11693): Task = 3 Task Running in Background
08-02 19:48:**36.325**: D/AsyncTaskExample(11693): Task = 4 Task Queued
08-02 19:48:**36.325**: D/AsyncTaskExample(11693): Task = 4 Task Running in Background
08-02 19:48:**36.815**: D/AsyncTaskExample(11693): Task = 1 Task Execution Completed
08-02 19:48:**36.915**: D/AsyncTaskExample(11693): Task = 5 Task Queued
08-02 19:48:**36.915**: D/AsyncTaskExample(11693): Task = 5 Task Running in Background
08-02 19:48:37.025: D/AsyncTaskExample(11693): Task = 2 Task Execution Completed
08-02 19:48:37.165: D/AsyncTaskExample(11693): Task = 3 Task Execution Completed
----------
```

Comment `Task Executed in thread pool` (1) and uncomment `Task executed Serially` (2).

Perform Click on button several times to start a task and see the result.

It is executing the task serially hence every task is started after the current task completed execution. Hence when Task 1's execution completes, only Task 2 starts running in background. Vice versa.

```
08-02 19:42:57.505: D/AsyncTaskExample(10299): Task = 1 Task Queued
08-02 19:42:57.505: D/AsyncTaskExample(10299): Task = 1 Task Running in Background
08-02 19:42:57.675: D/AsyncTaskExample(10299): Task = 2 Task Queued
08-02 19:42:57.835: D/AsyncTaskExample(10299): Task = 3 Task Queued
08-02 19:42:58.005: D/AsyncTaskExample(10299): Task = 4 Task Queued
08-02 19:42:58.155: D/AsyncTaskExample(10299): Task = 5 Task Queued
08-02 19:42:58.505: D/AsyncTaskExample(10299): Task = 1 Task Execution Completed
08-02 19:42:58.505: D/AsyncTaskExample(10299): Task = 2 Task Running in Background
08-02 19:42:58.755: D/AsyncTaskExample(10299): Task = 6 Task Queued
08-02 19:42:59.295: D/AsyncTaskExample(10299): Task = 7 Task Queued
08-02 19:42:59.505: D/AsyncTaskExample(10299): Task = 2 Task Execution Completed
08-02 19:42:59.505: D/AsyncTaskExample(10299): Task = 3 Task Running in Background
08-02 19:43:00.035: D/AsyncTaskExample(10299): Task = 8 Task Queued
08-02 19:43:00.505: D/AsyncTaskExample(10299): Task = 3 Task Execution Completed
08-02 19:43:**00.505**: D/AsyncTaskExample(10299): Task = 4 Task Running in Background
08-02 19:43:**01.505**: D/AsyncTaskExample(10299): Task = 4 Task Execution Completed
08-02 19:43:**01.515**: D/AsyncTaskExample(10299): Task = 5 Task Running in Background
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): Task = 5 Task Execution Completed
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): Task = 6 Task Running in Background
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): Task = 7 Task Running in Background
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): Task = 6 Task Execution Completed
08-02 19:43:04.515: D/AsyncTaskExample(10299): Task = 8 Task Running in Background
08-02 19:43:**04.515**: D/AsyncTaskExample(10299): Task = 7 Task Execution Completed
```

Section 250.6: Order of execution

When first introduced, `AsyncTasks` were executed serially on a single background thread. Starting with `DONUT`, this was changed to a pool of threads allowing multiple tasks to operate in parallel. Starting with `HONEYCOMB`, tasks are executed on a single thread to avoid common application errors caused by parallel execution.

If you truly want parallel execution, you can invoke `executeOnExecutor(java.util.concurrent.Executor, Object[])` with `THREAD_POOL_EXECUTOR`.

> SERIAL_EXECUTOR -> An Executor that executes tasks one at a time in serial order.
>
> THREAD_POOL_EXECUTOR -> An Executor that can be used to execute tasks in parallel.

sample :

```
Task task = new Task();
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB)
    task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR, data);
else
    task.execute(data);
```

Section 250.7: Publishing progress

Sometimes, we need to update the progress of the computation done by an `AsyncTask`. This progress could be represented by a string, an integer, etc. To do this, we have to use two functions. First, we need to set the `onProgressUpdate` function whose parameter type is the same as the second type parameter of our `AsyncTask`.

```
class YourAsyncTask extends AsyncTask<URL, Integer, Long> {
    @Override
    protected void onProgressUpdate(Integer... args) {
        setProgressPercent(args[0])
    }
}
```

Second, we have to use the function `publishProgress` necessarily on the `doInBackground` function, and that is all, the previous method will do all the job.

```
protected Long doInBackground(URL... urls) {
     int count = urls.length;
     long totalSize = 0;
     for (int i = 0; i < count; i++) {
         totalSize += Downloader.downloadFile(urls[i]);
         publishProgress((int) ((i / (float) count) * 100));
     }
     return totalSize;
 }
```

Chapter 251: Testing UI with Espresso

Section 251.1: Overall Espresso

Setup Espresso :

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

ViewMatchers – A collection of objects that implement `Matcher<? super View>` interface. You can pass one or more of these to the `onView` method to locate a view within the current view hierarchy.

ViewActions – A collection of `ViewActions` that can be passed to the `ViewInteraction.perform()` method (for example, `click()`).

ViewAssertions – A collection of `ViewAssertions` that can be passed the `ViewInteraction.check()` method. Most of the time, you will use the matches assertion, which uses a View matcher to assert the state of the currently selected view.

Espresso cheat sheet by google

CHEAT SHEET
espresso 2.1

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

```
onData(ObjectMatcher)
    .DataOptions
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Matchers

USER PROPERTIES
```
withId(...)
withText(...)
withTagKey(...)
withTagValue(...)
hasContentDescription(...)
withContentDescription(...)
withHint(...)
withSpinnerText(...)
hasLinks()
hasEllipsizedText()
hasMultilineTest()
```

HIERARCHY
```
withParent(Matcher)
withChild(Matcher)
hasDescendant(Matcher)
isDescendantOfA(Matcher)
hasSibling(Matcher)
isRoot()
```

INPUT
```
supportsInputMethods(...)
hasIMEAction(...)
```

UI PROPERTIES
```
isDisplayed()
isCompletelyDisplayed()
isEnabled()
hasFocus()
isClickable()
isChecked()
isNotChecked()
withEffectiveVisibility(...)
isSelected()
```

CLASS
```
isAssignableFrom(...)
withClassName(...)
```

ROOT MATCHERS
```
isFocusable()
isTouchable()
isDialog()
withDecorView()
isPlatformPopup()
```

OBJECT MATCHER
```
allOf(Matchers)
anyOf(Matchers)
is(...)
not(...)
endsWith(String)
startsWith(String)
instanceOf(Class)
```

SEE ALSO
```
Preference matchers
Cursor matchers
Layout matchers
```

Data Options

```
inAdapterView(Matcher)
atPosition(Integer)
onChildView(Matcher)
```

View Actions

CLICK/PRESS
```
click()
doubleClick()
longClick()
pressBack()
pressIMEActionButton()
pressKey([int/EspressoKey])
pressMenuKey()
closeSoftKeyboard()
openLink()
```

GESTURES
```
scrollTo()
swipeLeft()
swipeRight()
swipeUp()
swipeDown()
```

TEXT
```
clearText()
typeText(String)
typeTextIntoFocusedView(String)
replaceText(String)
```

View Assertions

```
matches(Matcher)
doesNotExist()
selectedDescendantsMatch(...)
```

POSITION ASSERTIONS
```
isLeftOf(Matcher)
isRightOf(Matcher)
isLeftAlignedWith(Matcher)
isRightAlignedWith(Matcher)
isAbove(Matcher)
isBelow(Matcher)
isBottomAlignedWith(Matcher)
isTopAlignedWith(Matcher)
```

LAYOUT ASSERTIONS
```
noEllipseizedText(Matcher)
noMultilineButtons()
noOverlaps([Matcher])
```

```
intended(IntentMatcher);
```

```
intending(IntentMatcher)
    .respondWith(ActivityResult);
```

Intent Matchers

INTENT
```
hasAction(...)
hasCategories(...)
hasData(...)
hasComponent(...)
hasExtra(...)
hasExtras(Matcher)
hasExtraWithKey(...)
hasType(...)
hasPackage()
toPackage(String)
hasFlag(int)
hasFlags(...)
isInternal()
```

URI
```
hasHost(...)
hasParamWithName(...)
hasPath(...)
hasParamWithValue(...)
hasScheme(...)
hasSchemeSpecificPart(...)
```

BUNDLE
```
hasEntry(...)
hasKey(...)
hasValue(...)
```

COMPONENT NAME
```
hasClassName(...)
hasPackageName(...)
hasShortClassName(...)
hasMyPackageName()
```

v2.1.0, 4/21/2015

Enter Text In EditText

```
onView(withId(R.id.edt_name)).perform(typeText("XYZ"));
        closeSoftKeyboard();
```

Perform Click on View

```
onView(withId(R.id.btn_id)).perform(click());
```

Checking View is Displayed

```
onView(withId(R.id.edt_pan_number)).check(ViewAssertions.matches((isDisplayed())));
```

Section 251.2: Espresso simple UI test

UI testing tools

Two main tools that are nowadays mostly used for UI testing are Appium and Espresso.

Appium	Espresso
blackbox test	white/gray box testing
what you see is what you can test	can change inner workings of the app and prepare it for testing, e.g. save some data to database or sharedpreferences before running the test
used mostly for integration end to end tests and entire user flows	testing the functionality of a screen and/or flow
can be abstracted so test written can be executed on iOS and Android	Android Only
well supported	well supported
supports parallel testing on multiple devices with selenium grid	Not out of the box parallel testing, plugins like Spoon exists until true Google support comes out

How to add espresso to the project

```
dependencies {
  // Set this dependency so you can use Android JUnit Runner
  androidTestCompile 'com.android.support.test:runner:0.5'
  // Set this dependency to use JUnit 4 rules
  androidTestCompile 'com.android.support.test:rules:0.5'
  // Set this dependency to build and run Espresso tests
  androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
  // Set this dependency to build and run UI Automator tests
  androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.2.2'
}
```

NOTE If you are using latest support libraries, annotations etc. you need to exclude the older versions from espresso to avoid collisions:

```
    // there is a conflict with the test support library (see
http://stackoverflow.com/questions/29857695)
    // so for now re exclude the support-annotations dependency from here to avoid clashes
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }
```

```
    // exclude a couple of more modules here because of <http://stackoverflow.com/questions/29216327>
and
    // more specifically of <https://code.google.com/p/android-test-kit/issues/detail?id=139>
    // otherwise you'll receive weird crashes on devices and dex exceptions on emulators
    // Espresso-contrib for DatePicker, RecyclerView, Drawer actions, Accessibility checks,
CountingIdlingResource
    androidTestCompile('com.android.support.test.espresso:espresso-contrib:2.2.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude group: 'com.android.support', module: 'design'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }
    //excluded specific packages due to https://code.google.com/p/android/issues/detail?id=183454
    androidTestCompile('com.android.support.test.espresso:espresso-intents:2.2.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }

    androidTestCompile('com.android.support.test.espresso:espresso-web:2.2.2') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }

    androidTestCompile('com.android.support.test:runner:0.5') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }
    androidTestCompile('com.android.support.test:rules:0.5') {
        exclude group: 'com.android.support', module: 'support-annotations'
        exclude module: 'support-annotations'
        exclude module: 'recyclerview-v7'
        exclude module: 'support-v4'
        exclude module: 'support-v7'
    }
```

Other than these imports it is necessary to add android instrumentation test runner to build.gradle android.defaultConfig:

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

Device setup

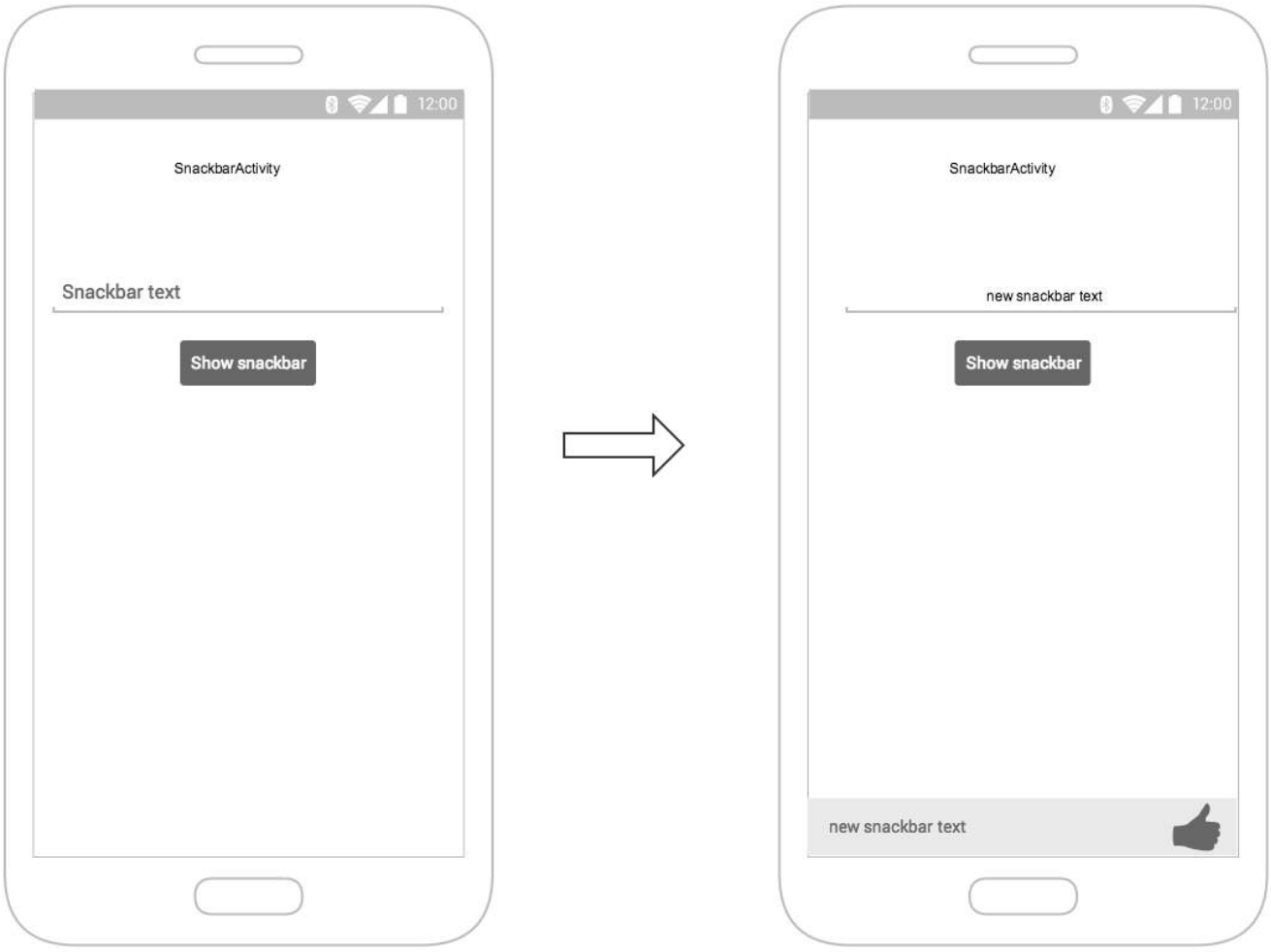
For non flaky test it is recommended to set following settings on your devices:

- Developer options / Disable Animations - reduces flakyness of tests
- Developer options / Stay awake - if you have dedicated devices for tests this is usefull
- Developer options / Logger buffer sizes - set to higher number if you run very big test suites on your phone
- Accessibility / Touch & Hold delay - long to avoid problems with tapping in espresso

Quite a setup from the real world ha? Well now when thats out of the way lets take a look how to setup a small test

Writing the test

Lets assume that we have the following screen:



The screen contains:

- text input field - **R.id.textEntry**
- button which shows snackbar with typed text when clicked - **R.id.shownSnackbarBtn**
- snackbar which should contain user typed text - **android.support.design.R.id.snackbar_text**

Now lets create a class that will test our flow:

```
/**
* Testing of the snackbar activity.
**/
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SnackbarActivityTest{
    //espresso rule which tells which activity to start
    @Rule
    public final ActivityTestRule<SnackbarActivity> mActivityRule =
        new ActivityTestRule<>(SnackbarActivity.class, true, false);


    @Override
    public void tearDown() throws Exception {
        super.tearDown();
        //just an example how tear down should cleanup after itself
```

```
        mDatabase.clear();
        mSharedPrefs.clear();
    }

    @Override
    public void setUp() throws Exception {
        super.setUp();
        //setting up your application, for example if you need to have a user in shared
        //preferences to stay logged in you can do that for all tests in your setup
        User mUser = new User();
        mUser.setToken("randomToken");
    }

    /**
    *Test methods should always start with "testXYZ" and it is a good idea to
    *name them after the intent what you want to test
    **/
    @Test
    public void testSnackbarIsShown() {
        //start our activity
        mActivityRule.launchActivity(null);
        //check is our text entry displayed and enter some text to it
        String textToType="new snackbar text";
        onView(withId(R.id.textEntry)).check(matches(isDisplayed()));
        onView(withId(R.id.textEntry)).perform(typeText(textToType));
        //click the button to show the snackbar
        onView(withId(R.id.shownSnackbarBtn)).perform(click());
        //assert that a view with snackbar_id with text which we typed and is displayed
        onView(allOf(withId(android.support.design.R.id.snackbar_text),
        withText(textToType))) .check(matches(isDisplayed()));
    }
}
```

As you noticed there are 3-4 things that you might notice come often:

onView(withXYZ) <-- viewMatchers with them you are able to find elements on screen

perform(click()) <-- viewActions, you can execute actions on elements you previously found

check(matches(isDisplayed())) <-- viewAssertions, checks you want to do on screens you previously found

All of these and many others can be found here:
https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html

Thats it, now you can run the test either with right clicking on the class name / test and selecting Run test or with command:

```
./gradlew connectedFLAVORNAMEAndroidTest
```

Section 251.3: Open Close DrawerLayout

```
public final class DrawerLayoutTest  {

  @Test public void Open_Close_Drawer_Layout() {
    onView(withId(R.id.drawer_layout)).perform(actionOpenDrawer());
    onView(withId(R.id.drawer_layout)).perform(actionCloseDrawer());
  }

  public static ViewAction actionOpenDrawer() {
```

```
    return new ViewAction() {
      @Override public Matcher<View> getConstraints() {
        return isAssignableFrom(DrawerLayout.class);
      }

      @Override public String getDescription() {
        return "open drawer";
      }

      @Override public void perform(UiController uiController, View view) {
        ((DrawerLayout) view).openDrawer(GravityCompat.START);
      }
    };
  }

  public static ViewAction actionCloseDrawer() {
    return new ViewAction() {
      @Override public Matcher<View> getConstraints() {
        return isAssignableFrom(DrawerLayout.class);
      }

      @Override public String getDescription() {
        return "close drawer";
      }

      @Override public void perform(UiController uiController, View view) {
        ((DrawerLayout) view).closeDrawer(GravityCompat.START);
      }
    };
  }

}
```

Section 251.4: Set Up Espresso

In the `build.gradle` file of your Android app module add next dependencies:

```
dependencies {
    // Android JUnit Runner
    androidTestCompile 'com.android.support.test:runner:0.5'
    // JUnit4 Rules
    androidTestCompile 'com.android.support.test:rules:0.5'
    // Espresso core
    androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
    // Espresso-contrib for DatePicker, RecyclerView, Drawer actions, Accessibility checks,
CountingIdlingResource
    androidTestCompile 'com.android.support.test.espresso:espresso-contrib:2.2.2'
    //UI Automator tests
    androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.2.2'
}
```

Specify the `AndroidJUnitRunner` for the `testInstrumentationRunner` parameter in the `build.gradle` file.

```
android {

  defaultConfig {
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }

}
```

Additionally, add this dependency for providing intent mocking support

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

And add this one for webview testing support

```
// Espresso-web for WebView support
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

Section 251.5: Performing an action on a view

It is possible to perform `ViewActions` on a view using the perform method.
The `ViewActions` class provides helper methods for the most common actions, like:

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

For example, to click on the view:

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

You can execute more than one action with one perform call:

```
onView(...).perform(typeText("Hello"), click());
```

If the view you are working with is located inside a `ScrollView` (vertical or horizontal), consider preceding actions that require the view to be displayed (like `click()` and `typeText()`) with `scrollTo()`. This ensures that the view is displayed before proceeding to the other action:

```
onView(...).perform(scrollTo(), click());
```

Section 251.6: Finding a view with onView

With the `ViewMatchers` you can find view in the current view hierarchy.

To find a view, use the `onView()` method with a view matcher which selects the correct view. The `onView()` methods return an object of type `ViewInteraction`.

For example, finding a view by its `R.id` is as simple as:

```
onView(withId(R.id.my_view))
```

Finding a view with a text:

```
onView(withText("Hello World"))
```

Section 251.7: Create Espresso Test Class

Place next java class in **src/androidTest/java** and run it.

```
public class UITest {
```

```
  @Test public void Simple_Test() {
    onView(withId(R.id.my_view))         // withId(R.id.my_view) is a ViewMatcher
        .perform(click())                // click() is a ViewAction
        .check(matches(isDisplayed()));  // matches(isDisplayed()) is a ViewAssertion
  }

}
```

Section 251.8: Up Navigation

```
@Test
public void testUpNavigation() {
    intending(hasComponent(ParentActivity.class.getName())).respondWith(new
Instrumentation.ActivityResult(0, null));

    onView(withContentDescription("Navigate up")).perform(click());

    intended(hasComponent(ParentActivity.class.getName()));
}
```

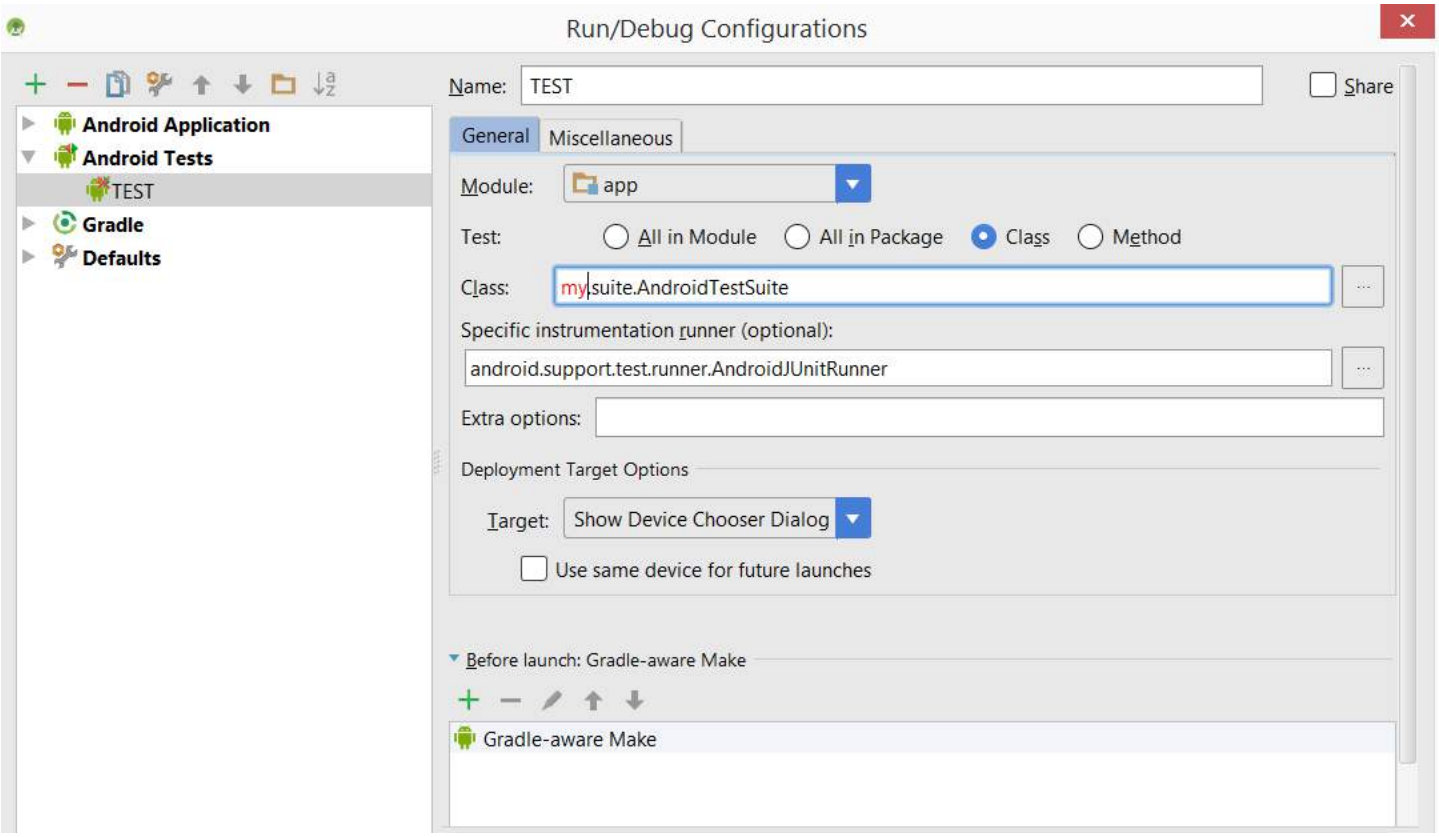
Note that this is a workaround and will collide with other Views that have the same content description.

Section 251.9: Group a collection of test classes in a test suite

You can organize the execution of your instrumented unit tests defining a Suite.

```
/**
 * Runs all unit tests.
 */
@RunWith(Suite.class)
@Suite.SuiteClasses({MyTest1.class ,
         MyTest2.class,
         MyTest3.class})
public class AndroidTestSuite {}
```

Then in AndroidStudio you can run with gradle or setting a new configuration like:



Test suites can be nested.

Section 251.10: Espresso custom matchers

Espresso by default has many matchers that help you find views that you need to do some checks or interactions with them.

Most important ones can be found in the following cheat sheet:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Some examples of matchers are:

- withId(R.id.ID_of_object_you_are_looking_for);
- withText("Some text you expect object to have");
- isDisplayed() <-- check is the view visible
- doesNotExist() <-- check that the view does not exist

All of these are very useful for everyday use, but if you have more complex views writing your custom matchers can make the tests more readable and you can reuse them in different places.

There are 2 most common type of matchers you can extend: **TypeSafeMatcher BoundedMatcher**

Implementing TypeSafeMatcher requires you to check the instanceOf the view you are asserting against, if its the correct type you match some of its properties against a value you provided to a matcher.

For example, type safe matcher that validates an image view has correct drawable:

```
public class DrawableMatcher extends TypeSafeMatcher<View> {

    private @DrawableRes final int expectedId;
    String resourceName;
```

```
    public DrawableMatcher(@DrawableRes int expectedId) {
        super(View.class);
        this.expectedId = expectedId;
    }

    @Override
    protected boolean matchesSafely(View target) {
        //Type check we need to do in TypeSafeMatcher
        if (!(target instanceof ImageView)) {
            return false;
        }
        //We fetch the image view from the focused view
        ImageView imageView = (ImageView) target;
        if (expectedId < 0) {
            return imageView.getDrawable() == null;
        }
        //We get the drawable from the resources that we are going to compare with image view source
        Resources resources = target.getContext().getResources();
        Drawable expectedDrawable = resources.getDrawable(expectedId);
        resourceName = resources.getResourceEntryName(expectedId);

        if (expectedDrawable == null) {
            return false;
        }
        //comparing the bitmaps should give results of the matcher if they are equal
        Bitmap bitmap = ((BitmapDrawable) imageView.getDrawable()).getBitmap();
        Bitmap otherBitmap = ((BitmapDrawable) expectedDrawable).getBitmap();
        return bitmap.sameAs(otherBitmap);
    }


    @Override
    public void describeTo(Description description) {
        description.appendText("with drawable from resource id: ");
        description.appendValue(expectedId);
        if (resourceName != null) {
            description.appendText("[");
            description.appendText(resourceName);
            description.appendText("]");
        }
    }
}
```

Usage of the matcher could be wrapped like this:

```
  public static Matcher<View> withDrawable(final int resourceId) {
    return new DrawableMatcher(resourceId);
}

 onView(withDrawable(R.drawable.someDrawable)).check(matches(isDisplayed()));
```

Bounded matchers are similar you just don't have to do the type check but, since that is done automagically for you:

```
/**
* Matches a {@link TextInputFormView}'s input hint with the given resource ID
*
* @param stringId
* @return
*/
```

```
public static Matcher<View> withTextInputHint(@StringRes final int stringId) {
    return new BoundedMatcher<View, TextInputFormView>(TextInputFormView.class) {
        private String mResourceName = null;

        @Override
        public void describeTo(final Description description) {
            //fill these out properly so your logging and error reporting is more clear
            description.appendText("with TextInputFormView that has hint ");
            description.appendValue(stringId);
            if (null != mResourceName) {
                description.appendText("[");
                description.appendText(mResourceName);
                description.appendText("]");
            }
        }

        @Override
        public boolean matchesSafely(final TextInputFormView view) {
            if (null == mResourceName) {
                try {
                    mResourceName = view.getResources().getResourceEntryName(stringId);
                } catch (Resources.NotFoundException e) {
                    throw new IllegalStateException("could not find string with ID " + stringId,
e);
                }
            }
            return view.getResources().getString(stringId).equals(view.getHint());
        }
    };
}
```

More on matchers can be read up on:

http://hamcrest.org/

https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html

Chapter 252: Writing UI tests - Android

Focus of this document is to represent goals and ways how to write android UI and integration tests. Espresso and UIAutomator are provided by Google so focus should be around these tools and their respective wrappers e.g. Appium, Spoon etc.

Section 252.1: MockWebServer example

In case your activities, fragments and UI require some background processing a good thing to use is a MockWebServer which runs localy on an android device which brings a closed and testable enviroment for your UI.

https://github.com/square/okhttp/tree/master/mockwebserver

First step is including the gradle dependency:

```
testCompile 'com.squareup.okhttp3:mockwebserver:(insert latest version)'
```

Now steps for running and using the mock server are:

- create mock server object
- start it at specific address and port (usually localhost:portnumber)
- enqueue responses for specific requests
- start the test

This is nicely explained in the github page of the mockwebserver but in our case we want something nicer and reusable for all tests, and JUnit rules will come nicely into play here:

```
/**
 *JUnit  rule that starts and stops a mock web server for test runner
*/
 public class MockServerRule extends UiThreadTestRule {

 private MockWebServer mServer;

 public static final int MOCK_WEBSERVER_PORT = 8000;

    @Override
    public Statement apply(final Statement base, Description description) {
        return new Statement() {
            @Override
            public void evaluate() throws Throwable {
                startServer();
                try {
                    base.evaluate();
                } finally {
                    stopServer();
                }
            }
        };
    }

    /**
     * Returns the started web server instance
     *
     * @return mock server
     */
    public MockWebServer server() {
```

```
        return mServer;
    }

    public void startServer() throws IOException, NoSuchAlgorithmException {
        mServer = new MockWebServer();
        try {
            mServer(MOCK_WEBSERVER_PORT);
        } catch (IOException e) {
            throw new IllegalStateException(e,"mock server start issue");
        }
    }

    public void stopServer() {
        try {
            mServer.shutdown();
        } catch (IOException e) {
            Timber.e(e, "mock server shutdown error");
        }
    }
}
```

Now lets assume that we have the exact same activity like in previous example, just in this case when we push the button app will fetch something from the network for example: https://someapi.com/name

This would return some text string which would be concatenated in the snackbar text e.g. NAME + text you typed in.

```
/**
* Testing of the snackbar activity with networking.
**/
@RunWith(AndroidJUnit4.class)
@LargeTest
public class SnackbarActivityTest{
    //espresso rule which tells which activity to start
    @Rule
    public final ActivityTestRule<SnackbarActivity> mActivityRule =
        new ActivityTestRule<>(SnackbarActivity.class, true, false);

    //start mock web server
    @Rule
    public final MockServerRule mMockServerRule = new MockServerRule();

    @Override
    public void tearDown() throws Exception {
       //same as previous example
    }

    @Override
    public void setUp() throws Exception {
       //same as previous example

       **//IMPORTANT:** point your application to your mockwebserver endpoint e.g.
       MyAppConfig.setEndpointURL("http://localhost:8000");
    }

    /**
    *Test methods should always start with "testXYZ" and it is a good idea to
    *name them after the intent what you want to test
    **/
    @Test
    public void testSnackbarIsShown() {
        //setup mockweb server
```

```
        mMockServerRule.server().setDispatcher(getDispatcher());

        mActivityRule.launchActivity(null);
        //check is our text entry displayed and enter some text to it
        String textToType="new snackbar text";
        onView(withId(R.id.textEntry)).check(matches(isDisplayed()));
        //we check is our snackbar showing text from mock webserver plus the one we typed
        onView(withId(R.id.textEntry)).perform(typeText("JazzJackTheRabbit" + textToType));
        //click the button to show the snackbar
        onView(withId(R.id.shownSnackbarBtn)).perform(click());
        //assert that a view with snackbar_id with text which we typed and is displayed
        onView(allOf(withId(android.support.design.R.id.snackbar_text),
        withText(textToType))) .check(matches(isDisplayed()));
    }

     /**
     *creates a mock web server dispatcher with prerecorded requests and responses
     **/
    private Dispatcher getDispatcher() {
        final Dispatcher dispatcher = new Dispatcher() {
            @Override
            public MockResponse dispatch(RecordedRequest request) throws InterruptedException {
                if (request.getPath().equals("/name")){
                    return new MockResponse().setResponseCode(200)
                            .setBody("JazzJackTheRabbit");
                }
                throw new IllegalStateException("no mock set up for " + request.getPath());
            }
        };
        return dispatcher;
    }
```

I would suggest wrapping the dispatcher in some sort of a Builder so you can easily chain and add new responses for your screens. e.g.

```
return newDispatcherBuilder()
            .withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
            .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
            .withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());
```

Section 252.2: IdlingResource

The power of idling resources lies in not having to wait for some app's processing (networking, calculations, animations, etc.) to finish with `sleep()`, which brings flakiness and/or prolongs the tests run. The official documentation can be found here.

Implementation

There are three things that you need to do when implementing `IdlingResource` interface:

- **`getName()`** - Returns the name of your idling resource.
- **`isIdleNow()`** - Checks whether your xyz object, operation, etc. is idle at the moment.
- **`registerIdleTransitionCallback`** (`IdlingResource.ResourceCallback` callback) - Provides a callback which you should call when your object transitions to idle.

Now you should create your own logic and determine when your app is idle and when not, since this is dependant on the app. Below you will find a simple example, just to show how it works. There are other examples online, but specific app implementation brings to specific idling resource implementations.

NOTES

- There have been some Google examples where they put `IdlingResources` in the code of the app. **Do not do this.** They presumably placed it there just to show how they work.
- Keeping your code clean and maintaining single principle of responsibility is up to you!

Example

Let us say that you have an activity which does weird stuff and takes a long time for the fragment to load and thus making your Espresso tests fail by not being able to find resources from your fragment (you should change how your activity is created and when to speed it up). But in any case to keep it simple, the following example shows how it should look like.

Our example idling resource would get two objects:

- The **tag** of the fragment which you need to find and waiting to get attached to the activity.
- A **FragmentManager** object which is used for finding the fragment.

```
/**
 * FragmentIdlingResource - idling resource which waits while Fragment has not been loaded.
 */
public class FragmentIdlingResource implements IdlingResource {
    private final FragmentManager mFragmentManager;
    private final String mTag;
    //resource callback you use when your activity transitions to idle
    private volatile ResourceCallback resourceCallback;

    public FragmentIdlingResource(FragmentManager fragmentManager, String tag) {
        mFragmentManager = fragmentManager;
        mTag = tag;
    }

    @Override
    public String getName() {
        return FragmentIdlingResource.class.getName() + ":" + mTag;
    }

    @Override
    public boolean isIdleNow() {
        //simple check, if your fragment is added, then your app has became idle
        boolean idle = (mFragmentManager.findFragmentByTag(mTag) != null);
        if (idle) {
            //IMPORTANT: make sure you call onTransitionToIdle
            resourceCallback.onTransitionToIdle();
        }
        return idle;
    }

    @Override
    public void registerIdleTransitionCallback(ResourceCallback resourceCallback) {
        this.resourceCallback = resourceCallback;
    }
}
```

Now that you have your `IdlingResource` written, you need to use it somewhere right?

Usage

Let us skip the entire test class setup and just look how a test case would look like:

```
@Test
public void testSomeFragmentText() {
    mActivityTestRule.launchActivity(null);

    //creating the idling resource
    IdlingResource fragmentLoadedIdlingResource = new
FragmentIdlingResource(mActivityTestRule.getActivity().getSupportFragmentManager(),
SomeFragmentText.TAG);
    //registering the idling resource so espresso waits for it
    Espresso.registerIdlingResources(idlingResource1);
    onView(withId(R.id.txtHelloWorld)).check(matches(withText(helloWorldText)));

    //lets cleanup after ourselves
    Espresso.unregisterIdlingResources(fragmentLoadedIdlingResource);
}
```

Combination with JUnit rule

This is not to hard; you can also apply the idling resource in form of a JUnit test rule. For example, let us say that you have some SDK that contains Volley in it and you want Espresso to wait for it. Instead of going through each test case or applying it in setup, you could create a JUnit rule and just write:

```
@Rule
public final SDKIdlingRule mSdkIdlingRule = new SDKIdlingRule(SDKInstanceHolder.getInstance());
```

Now since this is an example, don't take it for granted; all code here is imaginary and used only for demonstration purposes:

```
public class SDKIdlingRule implements TestRule {
    //idling resource you wrote to check is volley idle or not
    private VolleyIdlingResource mVolleyIdlingResource;
    //request queue that you need from volley to give it to idling resource
    private RequestQueue mRequestQueue;

    //when using the rule extract the request queue from your SDK
    public SDKIdlingRule(SDKClass sdkClass) {
        mRequestQueue = getVolleyRequestQueue(sdkClass);
    }

    private RequestQueue getVolleyRequestQueue(SDKClass sdkClass) {
        return sdkClass.getVolleyRequestQueue();
    }

    @Override
    public Statement apply(final Statement base, Description description) {
        return new Statement() {
            @Override
            public void evaluate() throws Throwable {
                //registering idling resource
                mVolleyIdlingResource = new VolleyIdlingResource(mRequestQueue);
                Espresso.registerIdlingResources(mVolleyIdlingResource);
                try {
                    base.evaluate();
                } finally {
                    if (mVolleyIdlingResource != null) {
                        //deregister the resource when test finishes
                        Espresso.unregisterIdlingResources(mVolleyIdlingResource);
                    }
                }
            }
        };
```

```
    }
}
```

Chapter 253: Unit testing in Android with JUnit

Section 253.1: Moving Business Logic Out of Android Componenets

A lot of the value from local JVM unit tests comes from the way you design your application. You have to design it in such a way where you can decouple your business logic from your Android Components. Here is an example of such a way using the Model-View-Presenter pattern. Lets practice this out by implementing a basic sign up screen that only takes a username and password. Our Android app is responsible for validating that the username the user supplies is not blank and that the password is at least eight characters long and contains at least one digit. If the username/password is valid we perform our sign up api call, otherwise we display an error message.

Example where business logic is highly coupled with Android Component.

```
public class LoginActivity extends Activity{
    ...
    private void onSubmitButtonClicked(){
        String username = findViewById(R.id.username).getText().toString();
        String password = findViewById(R.id.password).getText().toString();
        boolean isUsernameValid = username != null && username.trim().length() != 0;
        boolean isPasswordValid = password != null && password.trim().length() >= 8 &&
password.matches(".*\\d+.*");
        if(isUsernameValid && isPasswordValid){
            performSignUpApiCall(username, password);
        } else {
            displayInvalidCredentialsErrorMessage();
        }
    }
}
```

Example where business logic is decoupled from Android Component.

Here we define in a single class, LoginContract, that will house the various interactions between our various classes.

```
public interface LoginContract {
    public interface View {
        performSignUpApiCall(String username, String password);
        displayInvalidCredentialsErrorMessage();
    }
    public interface Presenter {
        void validateUserCredentials(String username, String password);
    }
}
```

Our LoginActivity is for the most part the same except that we have removed the responsibility of having to know how to validate a user's sign up form (our business logic). The LoginActivity will now rely on our new LoginPresenter to perform validation.

```
public class LoginActivity extends Activity implements LoginContract.View{
    private LoginContract.Presenter presenter;

    protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            presenter = new LoginPresenter(this);
            ....
```

```
        }
        ...

        private void onSubmitButtonClicked(){
            String username = findViewById(R.id.username).getText().toString();
            String password = findViewById(R.id.password).getText().toString();
            presenter.validateUserCredentials(username, password);
    }
    ...
}
```

Now your business logic will reside in your new LoginPresenter class.

```
public class LoginPresenter implements LoginContract.Presenter{
    private LoginContract.View view;

    public LoginPresenter(LoginContract.View view){
        this.view = view;
    }

    public void validateUserCredentials(String username, String password){
        boolean isUsernameValid = username != null && username.trim().length() != 0;
        boolean isPasswordValid = password != null && password.trim().length() >= 8 &&
password.matches(".*\\d+.*");
        if(isUsernameValid && isPasswordValid){
            view.performSignUpApiCall(username, password);
        } else {
            view.displayInvalidCredentialsErrorMessage();
        }
    }
}
```

And now we can create local JVM unit tests against your new LoginPresenter class.

```
public class LoginPresenterTest {

    @Mock
    LoginContract.View view;

    private LoginPresenter presenter;

    @Before
    public void setUp() throws Exception {
        MockitoAnnotations.initMocks(this);
        presenter = new LoginPresenter(view);
    }

    @Test
    public void test_validateUserCredentials_userDidNotEnterUsername_displayErrorMessage() throws
Exception {
        String username = "";
        String password = "kingslayer1";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
    }

    @Test
    public void
test_validateUserCredentials_userEnteredFourLettersAndOneDigitPassword_displayErrorMessage() throws
Exception {
        String username = "Jaime Lanninster";
```

```
        String password = "king1";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
    }

    @Test
    public void
test_validateUserCredentials_userEnteredNineLettersButNoDigitsPassword_displayErrorMessage() throws
Exception {
        String username = "Jaime Lanninster";
        String password = "kingslayer";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
    }

    @Test
    public void
test_validateUserCredentials_userEnteredNineLettersButOneDigitPassword_performApiCallToSignUpUser()
throws Exception {
        String username = "Jaime Lanninster";
        String password = "kingslayer1";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view).performSignUpApiCall(username, password);
    }
}
```

As you can see, when we extracted our business logic out of the LoginActivity and placed it in the LoginPresenter POJO. We can now create local JVM unit tests against our business logic.

It should be noted that there are various other implications from our change in architecture such as we are close to adhering to each class having a single responsibility, additional classes, etc. These are just side effects of the way I choose to go about performing this decoupling via the MVP style. MVP is just one way to go about this but there are other alternatives that you may want to look at such as MVVM. You just have to pick the best system that works for you.

Section 253.2: Creating Local unit tests

Place your test classes here: `/src/test/<pkg_name>/`

Example test class

```
public class ExampleUnitTest {
    @Test
    public void addition_isCorrect() throws Exception {
        int a=4, b=5, c;
        c = a + b;
        assertEquals(9, c); // This test passes
        assertEquals(10, c); //Test fails
    }
}
```

Breakdown

```
public class ExampleUnitTest {
  ...
}
```

The test class, you can create several test classes and place them inside the test package.

```
@Test
```

```
public void addition_isCorrect() {
    ...
}
```

The test method, several test methods can be created inside a test class.

Notice the annotation `@Test`.

> The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.

There are several other useful annotations like `@Before`, `@After` etc. This page would be a good place to start.

```
assertEquals(9, c); // This test passes
assertEquals(10, c); //Test fails
```

These methods are member of the **`Assert`** class. Some other useful methods are `assertFalse()`, `assertNotNull()`, `assertTrue` etc. Here's an elaborate Explanation.

Annotation Information for JUnit Test:

@Test: The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case. To run the method, JUnit first constructs a fresh instance of the class then invokes the annotated method.

@Before: When writing tests, it is common to find that several tests need similar objects created before they can run. Annotating a public void method with `@Before` causes that method to be run before the Test method.

@After: If you allocate external resources in a Before method you need to release them after the test runs. Annotating a public void method with `@After` causes that method to be run after the Test method. All `@After` methods are guaranteed to run even if a Before or Test method throws an exception

Tip Quickly create test classes in Android Studio

- Place the cursor on the class name for which you want to create a test class.
- Press Alt + Enter (Windows).
- Select Create Test, hit Return.
- Select the methods for which you want to create test methods, click OK.
- Select the directory where you want to create the test class.
- You're done, this what you get is your first test.

Tip Easily execute tests in Android Studio

- Right click test the package.
- Select Run 'Tests in ...
- All the tests in the package will be executed at once.

Section 253.3: Getting started with JUnit

Setup

To start unit testing your Android project using JUnit you need to add the JUnit dependency to your project and you need to create a test source-set which is going to contain the source code for the unit tests. Projects created with Android Studio often already include the JUnit dependency and the test source-set

Add the following line to your module `build.gradle` file within the dependencies `Closure`:

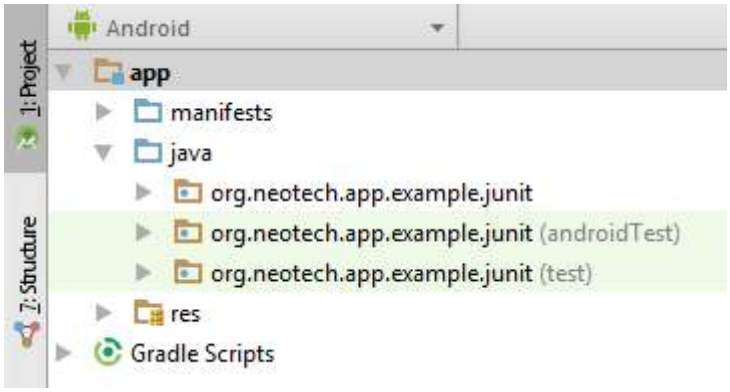
```
testCompile 'junit:junit:4.12'
```

JUnit test classes are located in a special source-set named `test`. If this source-set does not exist you need to create a new folder yourself. The folder structure of a default Android Studio (Gradle based) project looks like this:

```
<project-root-folder>
    /app (module root folder)
        /build
        /libs
        /src
            /main (source code)
            /test (unit test source code)
            /androidTest (instrumentation test source code)
        build.gradle (module gradle file)
    /build
    /gradle
    build.gradle (project gradle file)
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle (gradle settings)
```

If your project doesn't have the `/app/src/test` folder you need to create it yourself. Within the `test` folder you also need a `java` folder (create it if it doesn't exist). The java folder in the `test` source set should contains the same package structure as your `main` source-set.

If setup correctly your project structure (in the Android view in Android Studio) should look like this:



Note: You don't necessarily need to have the `androidTest` source-set, this source-set is often found in projects created by Android Studio and is included here for reference.

Writing a test

1. **Create a new class within the `test` source-set.**
   Right click the test source-set in the project view choose **`New`** > `Java` **`class`**.
   The most used naming pattern is to use the name of the class you're going to test with `Test` added to it. So `StringUtilities` becomes `StringUtilitiesTest`.

2. **Add the `@RunWith` annotation**
   The `@RunWith` annotation is needed in order to make JUnit run the tests we're going to define in our test class. The default JUnit runner (for JUnit 4) is the `BlockJUnit4ClassRunner` but instead of using this running

directly its more convenient to use the alias `JUnit4` which is a shorthand for the default JUnit runner.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

}
```

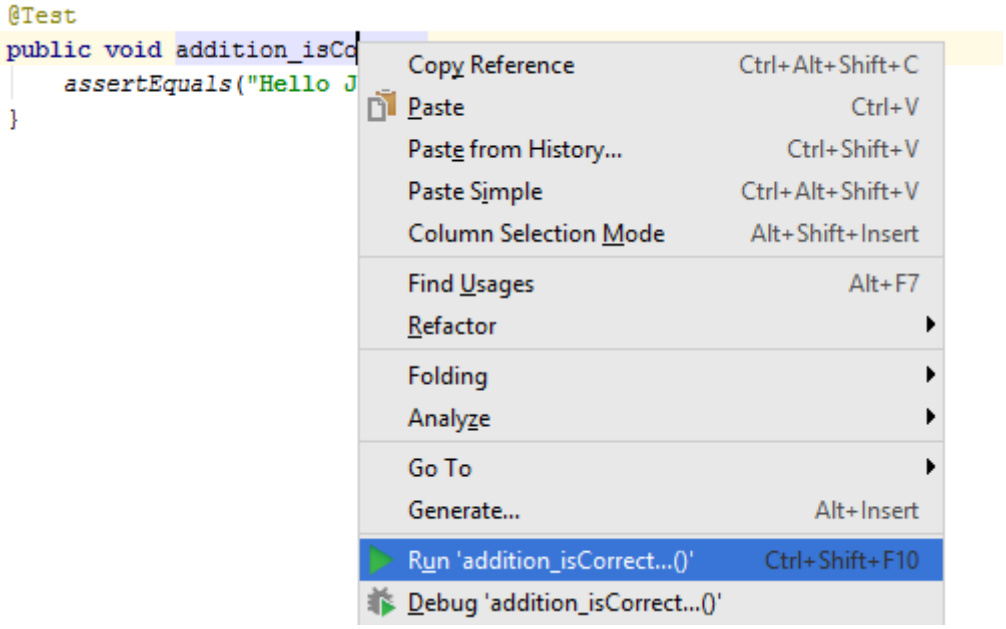
3. **Create a test**
   A unit test is essentially just a method which, in most cases, should not fail if run. In other words it should not throw an exception. Inside a test method you will almost always find assertions that check if specific conditions are met. If an assertion fails it throws an exception which causes the method/test to fail. A test method is always annotated with the `@Test` annotation. Without this annotation JUnit won't automatically run the test.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

    @Test
    public void addition_isCorrect() throws Exception {
        assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
    }
}
```

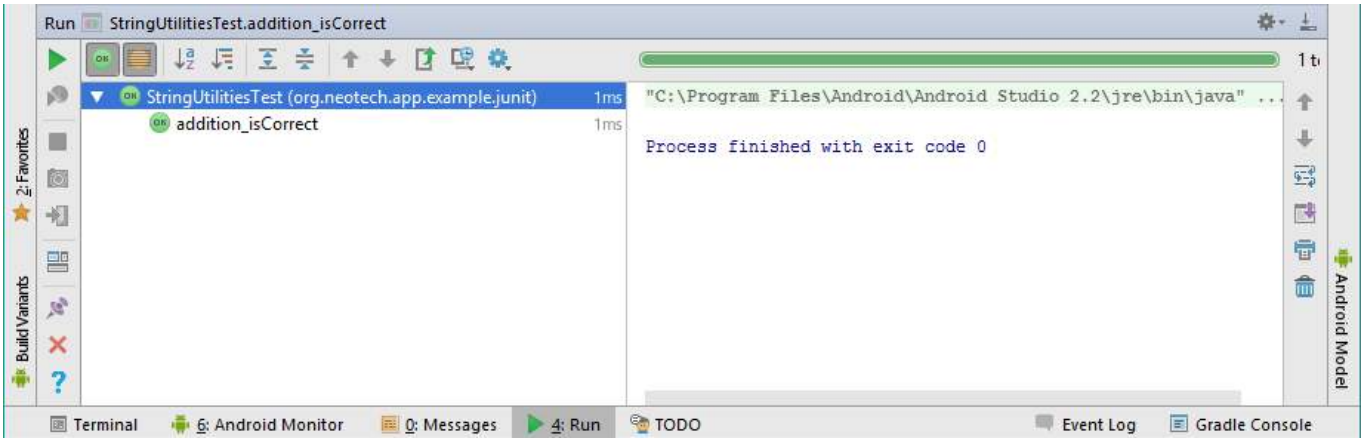
Note: unlike the standard Java method naming convention unit test method names do often contain underscores.

Running a test

1. **Method**
   To run a single test method you can right click the method and click `Run 'addition_isCorrect()'` or use the keyboard shortcut `ctrl+shift+f10`.



If everything is setup correctly JUnit starts running the method and you should see the following interface within Android Studio:



2. **Class**
   You can also run all the tests defined in a single class, by right clicking the class in the project view and clicking `Run 'StringUtilitiesTest '` or use the keyboard shortcut `ctrl+shift+f10` if you have selected the class in the project view.

3. **Package (everything)**
   If you wan't to run all the tests defined in the project or in a package you can just right click the package and click `Run ...` just like you would run all the tests defined in a single class.

Section 253.4: Exceptions

JUnit can also be used to test if a method throws a specific exception for a given input.

In this example we will test if the following method really throws an exception if the Boolean format (input) is not recognized/unknown:

```
public static boolean parseBoolean(@NonNull String raw) throws IllegalArgumentException{
    raw = raw.toLowerCase().trim();
    switch (raw) {
        case "t": case "yes": case "1": case "true":
            return true;
        case "f": case "no": case "0": case "false":
            return false;
        default:
            throw new IllegalArgumentException("Unknown boolean format: " + raw);
    }
}
```

By adding the `expected` parameter to the `@Test` annotation, one can define which exception is expected to be thrown. The unit test will fail if this exception does not occur, and succeed if the exception is indeed thrown:

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
    StringUtilities.parseBoolean("Hello JUnit");
}
```

This works well, however, it does limit you to just a single test case within the method. Sometimes you might want to test multiple cases within a single method. A technique often used to overcome this limitation is using `try-catch` blocks and the `Assert.fail()` method:

```
@Test
public void parseBoolean_parsesInvalidFormats_throwsException(){
```

```
    try {
        StringUtilities.parseBoolean("Hello!");
        fail("Expected IllegalArgumentException");
    } catch(IllegalArgumentException e){
    }

    try {
        StringUtilities.parseBoolean("JUnit!");
        fail("Expected IllegalArgumentException");
    } catch(IllegalArgumentException e){
    }
}
```

Note: Some people consider it to be bad practice to test more than a single case inside a unit test.

Section 253.5: Static import

JUnit defines quite some `assertEquals` methods at least one for each primitive type and one for Objects is available. These methods are by default not directly available to call and should be called like this: `Assert.assertEquals`. But because these methods are used so often people almost always use a static import so that the method can be directly used as if it is part of the class itself.

To add a static import for the `assertEquals` method use the following import statement:

```
import static org.junit.Assert.assertEquals;
```

You can also static import all assert methods including the `assertArrayEquals`, `assertNotNull` and `assertFalse` etc. using the following static import:

```
import static org.junit.Assert.*;
```

Without static import:

```
@Test
public void addition_isCorrect(){
    Assert.assertEquals(4 , 2 + 2);
}
```

With static import:

```
@Test
public void addition_isCorrect(){
    assertEquals(4 , 2 + 2);
}
```

Chapter 254: Inter-app UI testing with UIAutomator

Section 254.1: Prepare your project and write the first UIAutomator test

Add the required libraries into the dependencies section of your Android module's build.gradle:

```
android {
...
  defaultConfig {
    ...
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }
}

dependencies {
  ...
  androidTestCompile 'com.android.support.test:runner:0.5'
  androidTestCompile 'com.android.support.test:rules:0.5'
  androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.2'
  androidTestCompile 'com.android.support:support-annotations:23.4.0'
}
```

☐ Note that of course the versions may differ in the mean time.

After this sync with the changes.

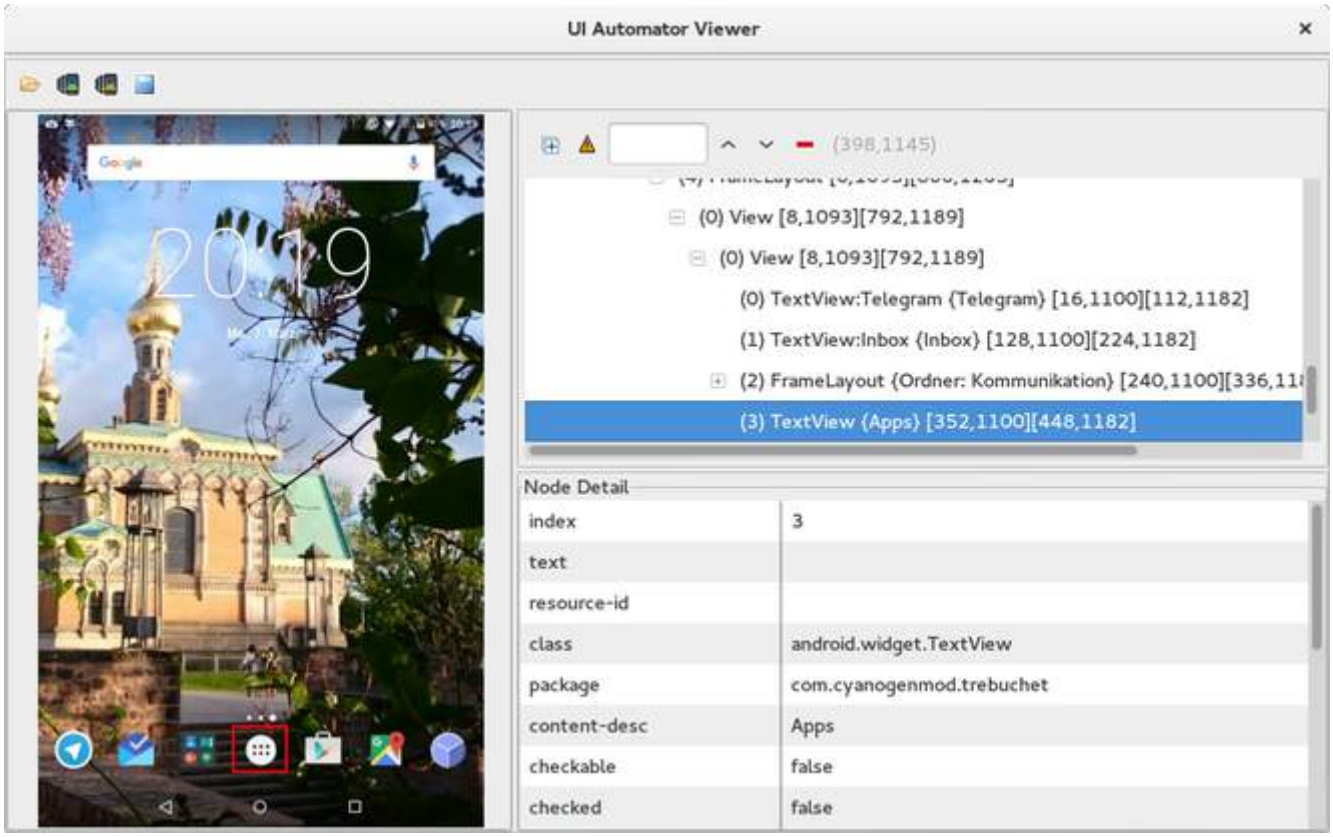
Then add a new Java class inside the androidTest folder:

```
public class InterAppTest extends InstrumentationTestCase {

  private UiDevice device;

  @Override
  public void setUp() throws Exception {
     device = UiDevice.getInstance(getInstrumentation());
  }

  public void testPressHome() throws Exception {
    device.pressHome();
  }
}
```

By making a right click on the class tab and on "Run "InterAppTest" executes this test.

Section 254.2: Writing more complex tests using the UIAutomatorViewer

In order to enable writing more complex UI tests the *UIAutomatorViewer* is needed. The tool located at */tools/* makes a fullscreen screenshot including the layouts of the currently displayed views. See the subsequent picture to get an idea of what is shown:



For the UI tests we are looking for *resource-id*, *content-desc* or something else to identify a view and use it inside our tests.

The *uiautomatorviewer* is executed via terminal.

If we now for instance want to click on the applications button and then open some app and swipe around, this is how the test method can look like:

```
public void testOpenMyApp() throws Exception {
    // wake up your device
    device.wakeUp();

    // switch to launcher (hide the previous application, if some is opened)
    device.pressHome();

    // enter applications menu (timeout=200ms)
    device.wait(Until.hasObject(By.desc(("Apps"))), 200);
    UiObject2 appsButton = device.findObject(By.desc(("Apps")));
    assertNotNull(appsButton);
    appsButton.click();

    // enter some application (timeout=200ms)
    device.wait(Until.hasObject(By.desc(("MyApplication"))), 200);
    UiObject2 someAppIcon = device.findObject(By.desc(("MyApplication")));
    assertNotNull(someAppIcon);
    someAppIcon.click();

    // do a swipe (steps=20 is 0.1 sec.)
    device.swipe(200, 1200, 1300, 1200, 20);
    assertTrue(isSomeConditionTrue)
}
```

Section 254.3: Creating a test suite of UIAutomator tests

Putting UIAutomator tests together to a test suite is a quick thing:

```
package de.androidtest.myapplication;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({InterAppTest1.class, InterAppTest2.class})
public class AppTestSuite {}
```

Execute similar to a single test by clicking right and run the suite.

Chapter 255: Lint Warnings

Section 255.1: Using tools:ignore in xml files

The attribute `tools:ignore` can be used in xml files to dismiss lint warnings.

BUT dismissing lint warnings with this technique is most of the time the wrong way to proceed.

A lint warning must be understood and fixed... it can be ignored if and only if you have a full understanding of it's meaning and a strong reason to ignore it.

Here is a use case where it legitimate to ignore a lint warning:

- You are developing a system-app (signed with the device manufacturer key)
- Your app need to change the device date (or any other protected action)

Then you can do this in your manifest : (i.e. requesting the protected permission and ignoring the lint warning because you know that in your case the permission will be granted)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:tools="http://schemas.android.com/tools"
     ...>
    <uses-permission android:name="android.permission.SET_TIME"
       tools:ignore="ProtectedPermissions"/>
```

Section 255.2: Configure LintOptions with gradle

You can configure lint by adding a `lintOptions` section in the `build.gradle` file:

```
android {

    //.....

    lintOptions {
        // turn off checking the given issue id's
        disable 'TypographyFractions','TypographyQuotes'

        // turn on the given issue id's
        enable 'RtlHardcoded','RtlCompat', 'RtlEnabled'

        // check *only* the given issue id's
        check 'NewApi', 'InlinedApi'

       // set to true to turn off analysis progress reporting by lint
       quiet true

       // if true, stop the gradle build if errors are found
       abortOnError false

       // if true, only report errors
       ignoreWarnings true
    }
}
```

You can run lint for a specific variant (see below), e.g. `./gradlew lintRelease`, or for all variants (`./gradlew lint`), in which case it produces a report which describes which specific variants a given issue applies to.

Check here for the DSL reference for all available options.

Section 255.3: Configuring lint checking in Java and XML source files

You can disable Lint checking from your Java and XML source files.

Configuring lint checking in Java

To disable Lint checking specifically for a **Java class** or method in your Android project, add the `@SuppressLint` **annotation** to that Java code.

Example:

```
@SuppressLint("NewApi")
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
```

To disable checking for all Lint issues:

```
@SuppressLint("all")
```

Configuring lint checking in XML

You can use the `tools:ignore` attribute to disable Lint checking for specific sections of your **XML files**.

For example:

```
tools:ignore="NewApi,StringFormatInvalid"
```

To suppress checking for all Lint issues in the XML element, use

```
tools:ignore="all"
```

Section 255.4: How to configure the lint.xml file

You can specify your Lint checking preferences in the `lint.xml` file. If you are creating this file manually, place it in the root directory of your Android project. If you are configuring Lint preferences in Android Studio, the lint.xml file is automatically created and added to your Android project for you.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
    <lint>
        <!-- list of issues to configure -->
</lint>
```

By setting the severity attribute value in the tag, you can disable Lint checking for an issue or change the severity level for an issue.

The following example shows the contents of a `lint.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<lint>
```

```
    <!-- Disable the given check in this project -->
    <issue id="IconMissingDensityFolder" severity="ignore" />

    <!-- Ignore the ObsoleteLayoutParam issue in the specified files -->
    <issue id="ObsoleteLayoutParam">
        <ignore path="res/layout/activation.xml" />
        <ignore path="res/layout-xlarge/activation.xml" />
    </issue>

    <!-- Ignore the UselessLeaf issue in the specified file -->
    <issue id="UselessLeaf">
        <ignore path="res/layout/main.xml" />
    </issue>

    <!-- Change the severity of hardcoded strings to "error" -->
    <issue id="HardcodedText" severity="error" />
</lint>
```

Section 255.5: Mark Suppress Warnings

It's good practice to mark some warnings in your code. For example, some deprecated methods is need for your testing, or old support version. But Lint checking will mark that code with warnings. For avoiding this problem, you need use annotation @SuppressWarnings.

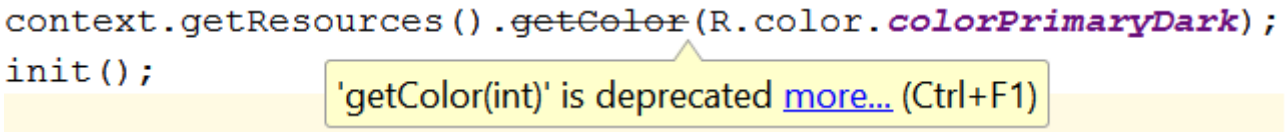
For example, add ignoring to warnings to deprecated methods. You need to put warnings description in annotation also:

```
@SuppressWarnings("deprecated");
public void setAnotherColor (int newColor) {
    getApplicationContext().getResources().getColor(newColor)
}
```

Using this annotation you can ignore all warnings, including Lint, Android, and other. Using Suppress Warnings, helps to understand code correctly!

Section 255.6: Importing resources without "Deprecated" error

Using the Android API 23 or higher, very often such situation can be seen:



This situation is caused by the structural change of the Android API regarding getting the resources. Now the function:

```
public int getColor(@ColorRes int id, @Nullable Theme theme) throws NotFoundException
```

should be used. But the android.support.v4 library has another solution.

Add the following dependency to the build.gradle file:

```
com.android.support:support-v4:24.0.0
```

Then all methods from support library are available:

```
ContextCompat.getColor(context, R.color.colorPrimaryDark);
ContextCompat.getDrawable(context, R.drawable.btn_check);
ContextCompat.getColorStateList(context, R.color.colorPrimary);
DrawableCompat.setTint(drawable);
ContextCompat.getColor(context,R.color.colorPrimaryDark));
```

Moreover more methods from support library can be used:

```
ViewCompat.setElevation(textView, 1F);
ViewCompat.animate(textView);
TextViewCompat.setTextAppearance(textView, R.style.AppThemeTextStyle);
...
```

Chapter 256: Performance Optimization

Your Apps performance is a crucial element of the user experience. Try to avoid bad performing patterns like doing work on the UI thread and learn how to write fast and responsive apps.

Section 256.1: Save View lookups with the ViewHolder pattern

Especially in a `ListView`, you can run into performance problems by doing too many `findViewById()` calls during scrolling. By using the `ViewHolder` pattern, you can save these lookups and improve your `ListView` performance.

If your list item contains a single `TextView`, create a `ViewHolder` class to store the instance:

```
static class ViewHolder {
    TextView myTextView;
}
```

While creating your list item, attach a `ViewHolder` object to the list item:

```
public View getView(int position, View convertView, ViewGroup parent) {
    Item i = getItem(position);
    if(convertView == null) {
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.list_item, parent, false);

        // Create a new ViewHolder and save the TextView instance
        ViewHolder holder = new ViewHolder();
        holder.myTextView = (TextView)convertView.findViewById(R.id.my_text_view);
        convertView.setTag(holder);
    }

    // Retrieve the ViewHolder and use the TextView
    ViewHolder holder = (ViewHolder)convertView.getTag();
    holder.myTextView.setText(i.getText());

    return convertView;
}
```

Using this pattern, `findViewById()` will only be called when a new `View` is being created and the `ListView` can recycle your views much more efficiently.

Chapter 257: Android Kernel Optimization

Section 257.1: Low RAM Configuration

Android now supports devices with 512MB of RAM. This documentation is intended to help OEMs optimize and configure Android 4.4 for low-memory devices. Several of these optimizations are generic enough that they can be applied to previous releases as well.

Enable Low Ram Device flag

We are introducing a new API called ActivityManager.isLowRamDevice() for applications to determine if they should turn off specific memory-intensive features that work poorly on low-memory devices.

For 512MB devices, this API is expected to return: "true" It can be enabled by the following system property in the device makefile.

```
PRODUCT_PROPERTY_OVERRIDES += ro.config.low_ram=true
```

Disable JIT

System-wide JIT memory usage is dependent on the number of applications running and the code footprint of those applications. The JIT establishes a maximum translated code cache size and touches the pages within it as needed. JIT costs somewhere between 3M and 6M across a typical running system.

The large apps tend to max out the code cache fairly quickly (which by default has been 1M). On average, JIT cache usage runs somewhere between 100K and 200K bytes per app. Reducing the max size of the cache can help somewhat with memory usage, but if set too low will send the JIT into a thrashing mode. For the really low-memory devices, we recommend the JIT be disabled entirely.

This can be achieved by adding the following line to the product makefile:

```
PRODUCT_PROPERTY_OVERRIDES += dalvik.vm.jit.codecachesize=0
```

Section 257.2: How to add a CPU Governor

The CPU governor itself is just 1 C file, which is located in kernel_source/drivers/cpufreq/, for example: cpufreq_smartass2.c. You are responsible yourself for find the governor (look in an existing kernel repo for your device) But in order to successfully call and compile this file into your kernel you will have to make the following changes:

1. Copy your governor file (cpufreq_govname.c) and browse to kernel_source/drivers/cpufreq, now paste it.
2. and open Kconfig (this is the interface of the config menu layout) when adding a kernel, you want it to show up in your config. You can do that by adding the choice of governor.

```
config CPU_FREQ_GOV_GOVNAMEHERE
tristate "'gov_name_lowercase' cpufreq governor"
depends on CPU_FREQ
help
governor' - a custom governor!
```

for example, for smartassV2.

```
config CPU_FREQ_GOV_SMARTASS2
 tristate "'smartassV2' cpufreq governor"
```

```
depends on CPU_FREQ
help
'smartassV2' - a "smart" optimized governor!
```

next to adding the choice, you also must declare the possibility that the governor gets chosen as default governor.

```
config CPU_FREQ_DEFAULT_GOV_GOVNAMEHERE
bool "gov_name_lowercase"
select CPU_FREQ_GOV_GOVNAMEHERE
help
Use the CPUFreq governor 'govname' as default.
```

for example, for smartassV2.

```
config CPU_FREQ_DEFAULT_GOV_SMARTASS2
 bool "smartass2"
 select CPU_FREQ_GOV_SMARTASS2
 help
 Use the CPUFreq governor 'smartassV2' as default.
```

– can't find the right place to put it? Just search for "`CPU_FREQ_GOV_CONSERVATIVE`", and place the code beneath, same thing counts for "`CPU_FREQ_DEFAULT_GOV_CONSERVATIVE`"

Now that Kconfig is finished you can save and close the file.

3. While still in the `/drivers/cpufreq` folder, open Makefile. In Makefile, add the line corresponding to your CPU Governor. for example:

```
obj-$(CONFIG_CPU_FREQ_GOV_SMARTASS2)    += cpufreq_smartass2.o
```

Be ware that you do not call the native C file, but the O file! which is the compiled C file. Save the file.

4. Move to: `kernel_source/includes/linux`. Now open `cpufreq.h` Scroll down until you see something like:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_ONDEMAND)
 extern struct cpufreq_governor cpufreq_gov_ondemand;
 #define CPUFREQ_DEFAULT_GOVERNOR    (&amp;cpufreq_gov_ondemand)
```

(other cpu governors are also listed there)

Now add your entry with the selected CPU Governor, example:

```
#elif defined(CONFIG_CPU_FREQ_DEFAULT_GOV_SMARTASS2)
 extern struct cpufreq_governor cpufreq_gov_smartass2;
 #define CPUFREQ_DEFAULT_GOVERNOR (&amp;cpufreq_gov_smartass2)
```

Save the file and close it.

The initial CPU Governor setup is now complete. when you've done all steps successfully, you should be able to choose your governor from the menu (`menuconfig`, `xconfig`, `gconfig`, `nconfig`). Once checked in the menu it will be included to the kernel.

Commit that is nearly the same as above instructions: Add smartassV2 and lulzactive governor commit

Section 257.3: I/O Schedulers

You can enhance your kernel by adding new I/O schedulers if needed. Globally, governors and schedulers are the same; they both provide a way how the system should work. However, for the schedulers it is all about the input/output datastream except for the CPU settings. I/O schedulers decide how an upcoming I/O activity will be scheduled. The standard schedulers such as *noop* or *cfq* are performing very reasonably.

I/O schedulers can be found in *kernel_source/block*.

1. Copy your I/O scheduler file (for example, *sio-iosched.c*) and browse to *kernel_source/block*. Paste the scheduler file there.
2. Now open *Kconfig.iosched* and add your choice to the *Kconfig*, for example for *SIO*:

```
config IOSCHED_SIO
  tristate "Simple I/O scheduler"
  default y
  ---help---
    The Simple I/O scheduler is an extremely simple scheduler,
    based on noop and deadline, that relies on deadlines to
    ensure fairness. The algorithm does not do any sorting but
    basic merging, trying to keep a minimum overhead. It is aimed
    mainly for aleatory access devices (eg: flash devices).
```

3. Then set the default choice option as follows:

```
default "sio" if DEFAULT_SIO
```

   Save the file.

4. Open the *Makefile* in *kernel_source/block/* and simply add the following line for *SIO*:

```
obj-$(CONFIG_IOSCHED_SIO)    += sio-iosched.o
```

   Save the file and you are done! The I/O schedulers should now pop up at the menu config.

Similar commit on GitHub: added Simple I/O scheduler.

Chapter 258: Memory Leaks

Section 258.1: Avoid leaking Activities with AsyncTask

> **A word of caution**: AsyncTask has many gotcha's apart from the memory leak described here. So be careful with this API, or avoid it altogether if you don't fully understand the implications. There are many alternatives (Thread, EventBus, RxAndroid, etc).

One common mistake with `AsyncTask` is to capture a strong reference to the host `Activity` (or `Fragment`):

```
class MyActivity extends Activity {
  private AsyncTask<Void, Void, Void> myTask = new AsyncTask<Void, Void, Void>() {
    // Don't do this! Inner classes implicitly keep a pointer to their
    // parent, which in this case is the Activity!
  }
}
```

This is a problem because `AsyncTask` can easily outlive the parent `Activity`, for example if a configuration change happens while the task is running.

The right way to do this is to make your task a **`static`** class, which does not capture the parent, and holding a weak reference to the host `Activity`:

```
class MyActivity extends Activity {
  static class MyTask extends AsyncTask<Void, Void, Void> {
    // Weak references will still allow the Activity to be garbage-collected
    private final WeakReference<MyActivity> weakActivity;

    MyTask(MyActivity myActivity) {
      this.weakActivity = new WeakReference<>(myActivity);
    }

    @Override
    public Void doInBackground(Void... params) {
      // do async stuff here
    }

    @Override
    public void onPostExecute(Void result) {
      // Re-acquire a strong reference to the activity, and verify
      // that it still exists and is active.
      MyActivity activity = weakActivity.get();
      if (activity == null
          || activity.isFinishing()
          || activity.isDestroyed()) {
        // activity is no longer valid, don't do anything!
        return;
      }

      // The activity is still valid, do main-thread stuff here
    }
  }
}
```

Section 258.2: Common memory leaks and how to fix them

1. Fix your contexts:

Try using the appropriate context: For example since a Toast can be seen in many activities instead of in just one, use `getApplicationContext()` for toasts, and since services can keep running even though an activity has ended start a service with:

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

Use this table as a quick guide for what context is appropriate:

	Application	Activity	Service	ContentProvider	BroadcastReceiver
Show a Dialog	NO	YES	NO	NO	NO
Start an Activity	NO[1]	YES	NO[1]	NO[1]	NO[1]
Layout Inflation	NO[2]	YES	NO[2]	NO[2]	NO[2]
Start a Service	YES	YES	YES	YES	YES
Bind to a Service	YES	YES	YES	YES	NO
Send a Broadcast	YES	YES	YES	YES	YES
Register BroadcastReceiver	YES	YES	YES	YES	NO[3]
Load Resource Values	YES	YES	YES	YES	YES

Original article on context here.

2. Static reference to Context

A serious memory leak mistake is keeping a static reference to `View`. Every `View` has an inner reference to the `Context`. Which means an old Activity with its whole view hierarchy will not be garbage collected until the app is terminated. You will have your app twice in memory when rotating the screen.

Make sure there is absolutely no static reference to View, Context or any of their descendants.

3. Check that you're actually finishing your services.

For example, I have an intentService that uses the Google location service API. And I forgot to call `googleApiClient.disconnect();`:

```
//Disconnect from API onDestroy()
if (googleApiClient.isConnected()) {
    LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
GoogleLocationService.this);
    googleApiClient.disconnect();
}
```

4. Check image and bitmaps usage:

If you are using Square's **library Picasso** I found I was leaking memory by not using the `.fit()`, that drastically reduced my memory footprint from 50MB in average to less than 19MB:

```
Picasso.with(ActivityExample.this)                    //Activity context
                .load(object.getImageUrl())
                .fit()                                 //This avoided the OutOfMemoryError
```

```
                .centerCrop()                           //makes image to not stretch
                .into(imageView);
```

5. If you are using broadcast receivers unregister them.
6. If you are using `java.util.Observer` (Observer pattern):

Make sure to use `deleteObserver(observer);`

Section 258.3: Detect memory leaks with the LeakCanary library

LeakCanary is an Open Source Java library to detect memory leaks in your debug builds.

Just add the dependencies in the `build.gradle`:

```
dependencies {
  debugCompile 'com.squareup.leakcanary:leakcanary-android:1.5.1'
  releaseCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
  testCompile 'com.squareup.leakcanary:leakcanary-android-no-op:1.5.1'
}
```

Then in your `Application` class:

```
public class ExampleApplication extends Application {

  @Override public void onCreate() {
    super.onCreate();

    if (LeakCanary.isInAnalyzerProcess(this)) {
      // This process is dedicated to LeakCanary for heap analysis.
      // You should not init your app in this process.
      return;
    }

    LeakCanary.install(this);
  }
}
```

Now LeakCanary will automatically show a notification when an activity memory leak is detected in your **debug** build.

NOTE: Release code will contain no reference to LeakCanary other than the two empty classes that exist in the `leakcanary-android-no-op` dependency.

Section 258.4: Anonymous callback in activities

Every Time you create an anonymous class, it retains an implicit reference to its parent class. So when you write:

```
public class LeakyActivity extends Activity
{

...

    foo.registerCallback(new BarCallback()
    {
        @Override
        public void onBar()
```

```
        {
            // do something
        }
    });
}
```

You are in fact sending a reference to your LeakyActivity instance to foo. When the user navigates away from your LeakyActivity, this reference can prevent the LeakyActivity instance from being garbage collected. This is a serious leak as activities hold a reference to their entire view hierarchy and are therefore rather large objects in memory.

How to avoid this leak:

You can of course avoid using anonymous callbacks in activities entirely. You can also unregister all of your callbacks with respect to the activity lifecycle. like so:

```
public class NonLeakyActivity extends Activity
{
    private final BarCallback mBarCallback = new BarCallback()
    {
        @Override
        public void onBar()
        {
            // do something
        }
    });

    @Override
    protected void onResume()
    {
        super.onResume();
        foo.registerCallback(mBarCallback);
    }

    @Override
    protected void onPause()
    {
        super.onPause();
        foo.unregisterCallback(mBarCallback);
    }
}
```

Section 258.5: Activity Context in static classes

Often you will want to wrap some of Android's classes in easier to use utility classes. Those utility classes often require a context to access the android OS or your apps' resources. A common example of this is a wrapper for the SharedPreferences class. In order to access Androids shared preferences one must write:

```
context.getSharedPreferences(prefsName, mode);
```

And so one may be tempted to create the following class:

```
public class LeakySharedPrefsWrapper
{
    private static Context sContext;

    public static void init(Context context)
    {
        sContext = context;
    }
```

```
    public int getInt(String name,int defValue)
    {
        return sContext.getSharedPreferences("a name", Context.MODE_PRIVATE).getInt(name,defValue);
    }
}
```

now, if you call `init()` with your activity context, the LeakySharedPrefsWrapper will retain a reference to your activity, preventing it from being garbage collected.

How to avoid:

When calling static helper functions, you can send in the application context using `context.getApplicationContext();`

When creating static helper functions, you can extract the application context from the context you are given (Calling getApplicationContext() on the application context returns the application context). So the fix to our wrapper is simple:

```
public static void init(Context context)
{
    sContext = context.getApplicationContext();
}
```

If the application context is not appropriate for your use case, you can include a Context parameter in each utility function, you should avoid keeping references to these context parameters. In this case the solution would look like so:

```
public int getInt(Context context,String name,int defValue)
{
    // do not keep a reference of context to avoid potential leaks.
    return context.getSharedPreferences("a name", Context.MODE_PRIVATE).getInt(name,defValue);
}
```

Section 258.6: Avoid leaking Activities with Listeners

If you implement or create a listener in an Activity, always pay attention to the lifecycle of the object that has the listener registered.

Consider an application where we have several different activities/fragments interested in when a user is logged in or out. One way of doing this would be to have a singleton instance of a `UserController` that can be subscribed to in order to get notified when the state of the user changes:

```
public class UserController {
    private static UserController instance;
    private List<StateListener> listeners;

    public static synchronized UserController getInstance() {
        if (instance == null) {
            instance = new UserController();
        }
        return instance;
    }

    private UserController() {
        // Init
    }
```

```
    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(listener);
    }

    public void logout() {
        for (StateListener listener : listeners) {
            listener.userLoggedOut();
        }
    }

    public void login() {
        for (StateListener listener : listeners) {
            listener.userLoggedIn();
        }
    }

    public interface StateListener {
        void userLoggedIn();
        void userLoggedOut();
    }
}
```

Then there are two activities, `SignInActivity`:

```
public class SignInActivity extends Activity implements UserController.StateListener{

    UserController userController;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        this.userController = UserController.getInstance();
        this.userController.registerUserStateChangeListener(this);
    }

    @Override
    public void userLoggedIn() {
        startMainActivity();
    }

    @Override
    public void userLoggedOut() {
        showLoginForm();
    }

    ...

    public void onLoginClicked(View v) {
        userController.login();
    }
}
```

And `MainActivity`:

```
public class MainActivity extends Activity implements UserController.StateListener{
    UserController userController;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        this.userController = UserController.getInstance();
        this.userController.registerUserStateChangeListener(this);
    }

    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
        finish();
    }

    ...

    public void onLogoutClicked(View v) {
        userController.logout();
    }
}
```

What happens with this example is that every time the user logs in and then logs out again, a `MainActivity` instance is leaked. The leak occurs because there is a reference to the activity in `UserController#listeners`.

Please note: Even if we use an anonymous inner class as a listener, the activity would still leak:

```
...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
        finish();
    }
});
...
```

The activity would still leak, because the anonymous inner class has an implicit reference to the outer class (in this case the activity). This is why it is possible to call instance methods in the outer class from the inner class. In fact, the only type of inner classes that do not have a reference to the outer class are **static** inner classes.

In short, all instances of non-static inner classes hold an implicit reference to the instance of the outer class that created them.

There are two main approaches to solving this, either by adding a method to remove a listener from `UserController#listeners` or using a `WeakReference` to hold the reference of the listeners.

Alternative 1: Removing listeners

Let us start by creating a new method `removeUserStateChangeListener(StateListener listener)`:

```
public class UserController {

    ...

    public void registerUserStateChangeListener(StateListener listener) {
```

```
        listeners.add(listener);
    }

    public void removeUserStateChangeListener(StateListener listener) {
        listeners.remove(listener);
    }

    ...
}
```

Then let us call this method in the activity's `onDestroy` method:

```
public class MainActivity extends Activity implements UserController.StateListener{
    ...

    @Override
    protected void onDestroy() {
        super.onDestroy();
        userController.removeUserStateChangeListener(this);
    }
}
```

With this modification the instances of `MainActivity` are no longer leaked when the user logs in and out. However, if the documentation isn't clear, chances are that the next developer that starts using `UserController` might miss that it is required to unregister the listener when the activity is destroyed, which leads us to the second method of avoiding these types of leaks.

Alternative 2: Using weak references

First off, let us start by explaining what a weak reference is. A weak reference, as the name suggests, holds a weak reference to an object. Compared to a normal instance field, which is a strong reference, a weak references does not stop the garbage collector, GC, from removing the objects. In the example above this would allow `MainActivity` to be garbage-collected after it has been destroyed if the `UserController` used `WeakReference` to the reference the listeners.

In short, a weak reference is telling the GC that if no one else has a strong reference to this object, go ahead and remove it.

Let us modify the `UserController` to use a list of `WeakReference` to keep track of it's listeners:

```
public class UserController {

    ...
    private List<WeakReference<StateListener>> listeners;
    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(new WeakReference<>(listener));
    }

    public void removeUserStateChangeListener(StateListener listenerToRemove) {
        WeakReference referencesToRemove = null;
        for (WeakReference<StateListener> listenerRef : listeners) {
            StateListener listener = listenerRef.get();
            if (listener != null && listener == listenerToRemove) {
                referencesToRemove = listenerRef;
                break;
            }
```

```
        }
        listeners.remove(referencesToRemove);
    }

    public void logout() {
        List referencesToRemove = new LinkedList();
        for (WeakReference<StateListener> listenerRef : listeners) {
            StateListener listener = listenerRef.get();
            if (listener != null) {
                listener.userLoggedOut();
            } else {
                referencesToRemove.add(listenerRef);
            }
        }
    }

    public void login() {
        List referencesToRemove = new LinkedList();
        for (WeakReference<StateListener> listenerRef : listeners) {
            StateListener listener = listenerRef.get();
            if (listener != null) {
                listener.userLoggedIn();
            } else {
                referencesToRemove.add(listenerRef);
            }
        }
    }
    ...
}
```

With this modification it doesn't matter whether or not the listeners are removed, since `UserController` holds no strong references to any of the listeners. However, writing this boilerplate code every time is cumbersome. Therefore, let us create a generic class called `WeakCollection`:

```
public class WeakCollection<T> {
    private LinkedList<WeakReference<T>> list;

    public WeakCollection() {
        this.list = new LinkedList<>();
    }
    public void put(T item){
        //Make sure that we don't re add an item if we already have the reference.
        List<T> currentList = get();
        for(T oldItem : currentList){
            if(item == oldItem){
                return;
            }
        }
        list.add(new WeakReference<T>(item));
    }

    public List<T> get() {
        List<T> ret = new ArrayList<>(list.size());
        List<WeakReference<T>> itemsToRemove = new LinkedList<>();
        for (WeakReference<T> ref : list) {
            T item = ref.get();
            if (item == null) {
                itemsToRemove.add(ref);
            } else {
                ret.add(item);
            }
```

```
        }
        for (WeakReference ref : itemsToRemove) {
            this.list.remove(ref);
        }
        return ret;
    }

    public void remove(T listener) {
        WeakReference<T> refToRemove = null;
        for (WeakReference<T> ref : list) {
            T item = ref.get();
            if (item == listener) {
                refToRemove = ref;
            }
        }
        if(refToRemove != null){
            list.remove(refToRemove);
        }
    }
}
```

Now let us re-write `UserController` to use `WeakCollection<T>` instead:

```
public class UserController {
    ...
    private WeakCollection<StateListener> listenerRefs;
    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listenerRefs.put(listener);
    }

    public void removeUserStateChangeListener(StateListener listenerToRemove) {
        listenerRefs.remove(listenerToRemove);
    }

    public void logout() {
        for (StateListener listener : listenerRefs.get()) {
            listener.userLoggedOut();
        }
    }

    public void login() {
        for (StateListener listener : listenerRefs.get()) {
            listener.userLoggedIn();
        }
    }

    ...
}
```

As shown in the code example above, the `WeakCollection<T>` removes all of the boilerplate code needed to use `WeakReference` instead of a normal list. To top it all off: If a call to `UserController#removeUserStateChangeListener(StateListener)` is missed, the listener, and all the objects it is referencing, will not leak.

Section 258.7: Avoid memory leaks with Anonymous Class,

Handler, Timer Task, Thread

In android, every developer uses `Anonymous Class` (Runnable) at least once in a project. Any `Anonymous Class` has a reference to its parent (activity). If we perform a long-running task, the parent activity will not be destroyed until the task is ended.
Example uses handler and Anonymous `Runnable` class. The memory will be leak when we quit the activity before the `Runnable` is finished.

```
new Handler().postDelayed(new Runnable() {
      @Override
      public void run() {
          // do abc long 5s or so
      }
  }, 10000); // run "do abc" after 10s. It same as timer, thread...
```

How do we solve it?

1. Don't do any long operating with `Anonymous Class` or we need a `Static class` for it and pass `WeakReference` into it (such as activity, view...). `Thread` is the same with `Anonymous Class`.
2. Cancel the `Handler`, `Timer` when activity is destroyed.

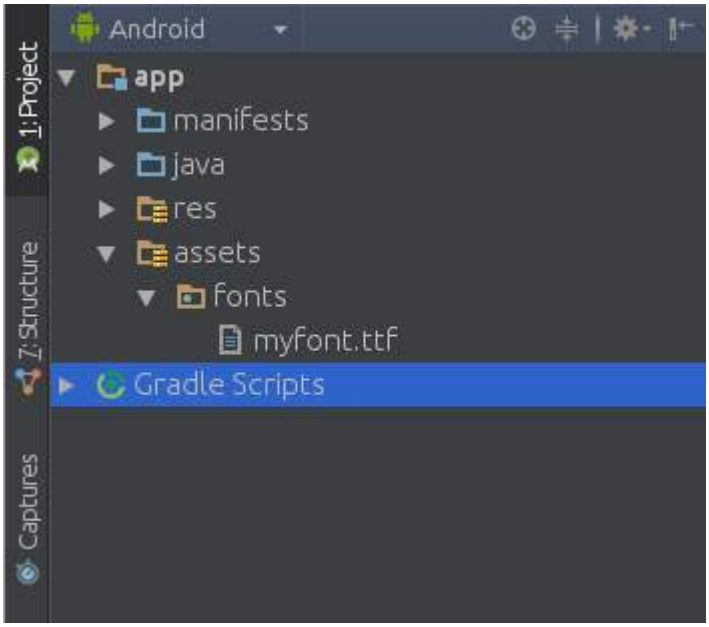
Chapter 259: Enhancing Android Performance Using Icon Fonts

Section 259.1: How to integrate Icon fonts

In order to use icon fonts, just follow the steps below:

- **Add the font file to your project**

  You may create your font icon file from online websites such as icomoon, where you can upload SVG files of the required icons and then download the created icon font. Then, place the *.ttf* font file into a folder named *fonts* (name it as you wish) in the assets folder:



- **Create a Helper Class**

  Now, create the following helper class, so that you can avoid repeating the initialisation code for the font:

```
public class FontManager {
  public static final String ROOT = "fonts/";
  FONT_AWESOME = ROOT + "myfont.ttf";
  public static Typeface getTypeface(Context context) {
    return Typeface.createFromAsset(context.getAssets(), FONT_AWESOME);
  }
}
```

  You may use the `Typeface` class in order to pick the font from the assets. This way you can set the typeface to various views, for example, to a button:

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

  Now, the button typeface has been changed to the newly created icon font.

- **Pick up the icons you want**

  Open the *styles.css* file attached to the icon font. There you will find the styles with Unicode characters of your icons:

```
.icon-arrow-circle-down:before {
  content: "\e001";
}
.icon-arrow-circle-left:before {
  content: "\e002";
}
.icon-arrow-circle-o-down:before {
  content: "\e003";
}
.icon-arrow-circle-o-left:before {
  content: "\e004";
}
```

  This resource file will serve as a dictionary, which maps the Unicode character associated with a specific icon to a human-readable name. Now, create the string resources as follows:

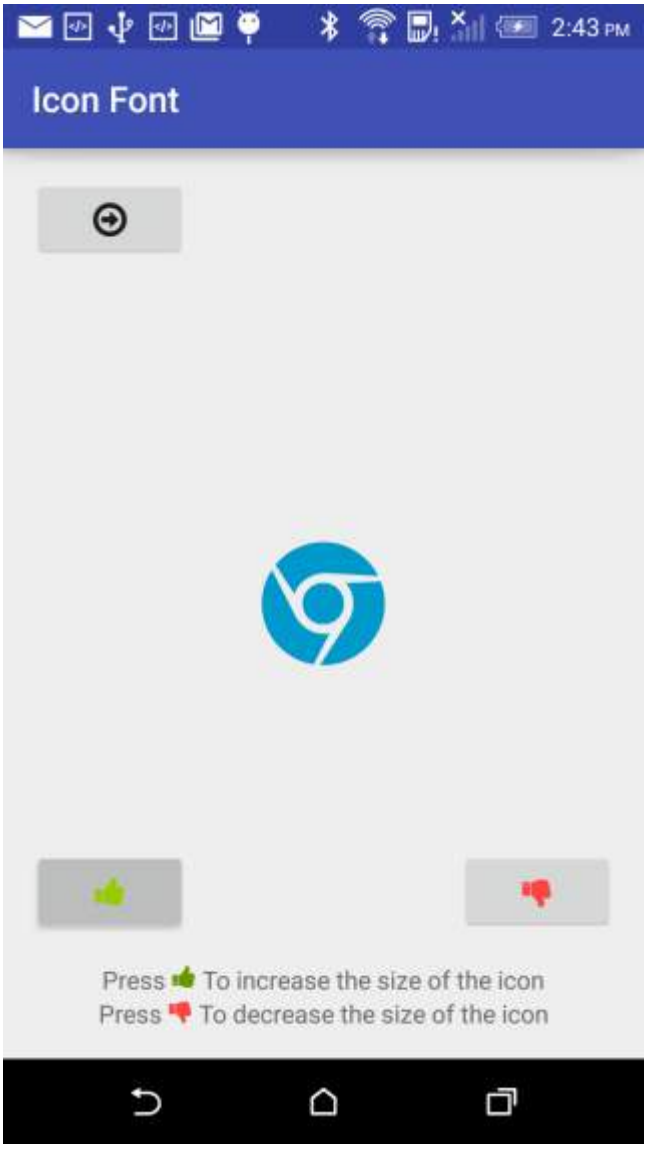
```
<resources>
  <! — Icon Fonts -->
  <string name="icon_arrow_circle_down">&#xe001; </string>
  <string name="icon_arrow_circle_left">&#xe002; </string>
  <string name="icon_arrow_circle-o_down">&#xe003; </string>
  <string name="icon_arrow_circle_o_left">&#xe004; </string>
</resources>
```

- **Use the icons in your code**

  Now, you may use your font in various views, for example, as follows:

```
button.setText(getString(R.string.icon_arrow_circle_left))
```

  You may also create button text views using icon fonts:



Section 259.2: TabLayout with icon fonts

```
public class TabAdapter extends FragmentPagerAdapter {

CustomTypefaceSpan fonte;
List<Fragment> fragments = new ArrayList<>(4);
private String[] icons = {"\ue001","\uE002","\uE003","\uE004"};

public TabAdapter(FragmentManager fm, CustomTypefaceSpan fonte) {
    super(fm);
    this.fonte = fonte
    for (int i = 0; i < 4; i++){
        fragments.add(MyFragment.newInstance());
    }
}

public List<Fragment> getFragments() {
    return fragments;
}

@Override
public Fragment getItem(int position) {
    return fragments.get(position);
}
```

```
@Override
public CharSequence getPageTitle(int position) {
    SpannableStringBuilder ss = new SpannableStringBuilder(icons[position]);
    ss.setSpan(fonte,0,ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE);
    ss.setSpan(new RelativeSizeSpan(1.5f),0,ss.length(), Spanned.SPAN_INCLUSIVE_INCLUSIVE );
    return ss;
}



@Override
public int getCount() {
    return 4;
}

}
```

- In this example, myfont.ttf is in Assets folder. Creating Assets folder
- In your activity class

```
//..
TabLayout tabs;
ViewPager tabs_pager;
public CustomTypefaceSpan fonte;
//..

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //...
    fm = getSupportFragmentManager();
    fonte = new CustomTypefaceSpan("icomoon",Typeface.createFromAsset(getAssets(),"myfont.ttf"));
    this.tabs = ((TabLayout) hasViews.findViewById(R.id.tabs));
    this.tabs_pager = ((ViewPager) hasViews.findViewById(R.id.tabs_pager));
    //...
}

@Override
protected void onStart() {
    super.onStart();
    //..
    tabs_pager.setAdapter(new TabAdapter(fm,fonte));
    tabs.setupWithViewPager(tabs_pager);
    //..
```

Chapter 260: Bitmap Cache

Parameter	Details
key	key to store bitmap in memory cache
bitmap	bitmap value which will cache into memory

Memory efficient bitmap caching: This is particularly important if your application uses animations as they will be stopped during GC cleanup and make your application appears sluggish to the user. A cache allows reusing objects which are expensive to create. If you load on object into memory, you can think of this as a cache for the object.Working with bitmap in android is tricky.It is more important to cache the bimap if you are going to use it repeatedly.

Section 260.1: Bitmap Cache Using LRU Cache

LRU Cache

The following example code demonstrates a possible implementation of the LruCache class for caching images.

```
private LruCache<String, Bitmap> mMemoryCache;
```

Here string value is key for bitmap value.

```
// Get max available VM memory, exceeding this amount will throw an
// OutOfMemory exception. Stored in kilobytes as LruCache takes an
// int in its constructor.
final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);

// Use 1/8th of the available memory for this memory cache.
final int cacheSize = maxMemory / 8;

mMemoryCache = new LruCache<String, Bitmap>(cacheSize) {
    @Override
    protected int sizeOf(String key, Bitmap bitmap) {
        // The cache size will be measured in kilobytes rather than
        // number of items.
        return bitmap.getByteCount() / 1024;
    }
};
```

For add bitmap to the memory cache

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }
}
```

For get bitmap from memory cache

```
public Bitmap getBitmapFromMemCache(String key) {
    return mMemoryCache.get(key);
}
```

For loading bitmap into imageview just use **getBitmapFromMemCache("Pass key").**

Chapter 261: Loading Bitmaps Effectively

This Topic Mainly Concentrate on Loading the Bitmaps Effectively in Android Devices.

When it comes to loading a bitmap, the question comes where it is loaded from. Here we are going to discuss about how to load the Bitmap from Resource with in the Android Device. i.e. eg from Gallery.

We will go through this by example which are discussed below.

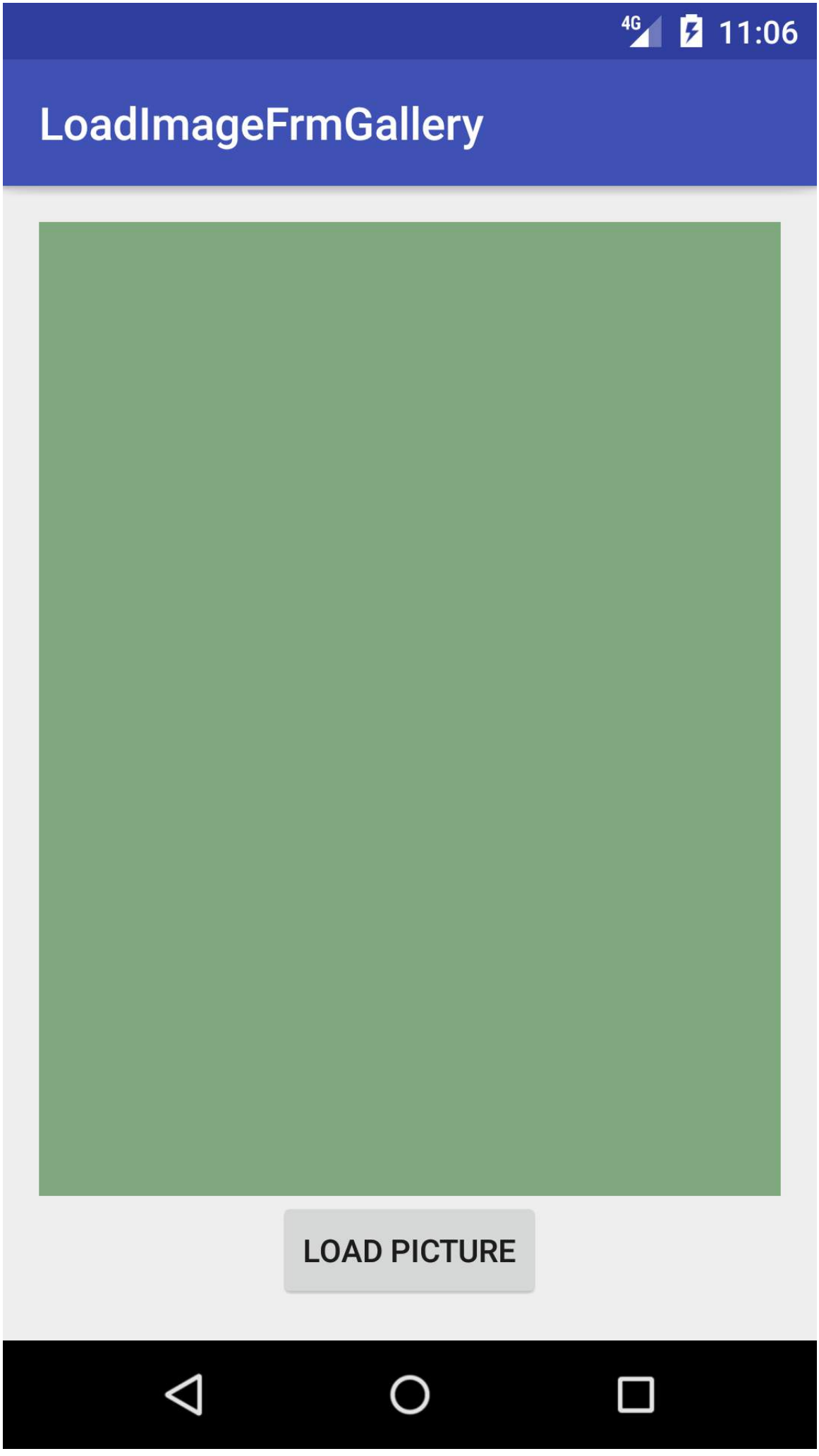
Section 261.1: Load the Image from Resource from Android Device. Using Intents

Using Intents to Load the Image from Gallery.

1. Initially you need to have the permission

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

2. Use the Following Code to have the layout as designed follows.



```
   <?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="androidexamples.idevroids.loadimagefrmgallery.MainActivity">

    <ImageView
        android:id="@+id/imgView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@color/abc_search_url_text_normal"></ImageView>

    <Button
        android:id="@+id/buttonLoadPicture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:text="Load Picture"
        android:layout_gravity="bottom|center"></Button>

</LinearLayout>
```

3. Use the Following code to Display the image with button Click.

Button Click will be

```
Button loadImg = (Button) this.findViewById(R.id.buttonLoadPicture);
loadImg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Intent i = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(i, RESULT_LOAD_IMAGE);
    }
});
```

3. Once you clicked on the button , it will open the gallery with help of intent.

You need to select image and send it back to main activity. Here with help of onActivityResult we can do that.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)  {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == RESULT_LOAD_IMAGE && resultCode == RESULT_OK && null != data) {
        Uri selectedImage = data.getData();
        String[] filePathColumn = { MediaStore.Images.Media.DATA };

        Cursor cursor = getContentResolver().query(selectedImage,
                filePathColumn, null, null, null);
        cursor.moveToFirst();

        int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
        String picturePath = cursor.getString(columnIndex);
        cursor.close();
```

```
        ImageView imageView = (ImageView) findViewById(R.id.imgView);
        imageView.setImageBitmap(BitmapFactory.decodeFile(picturePath));

    }
}
```

Chapter 262: Exceptions

Section 262.1: ActivityNotFoundException

This is a very common `Exception`. It causes your application to stop during the start or execution of your app. In the `LogCat` you see the message:

```
android.content.ActivityNotFoundException : Unable to find explicit activity class;
have you declared this activity in your AndroidManifest.xml?
```

In this case, check if you have declared your activity in the `AndroidManifest.xml` file.

The simplest way to declare your `Activity` in `AndroidManifest.xml` is:

```
<activity  android:name="com.yourdomain.YourStoppedActivity" />
```

Section 262.2: OutOfMemoryError

This is a runtime error that happens when you request a large amount of memory on the heap. This is common when loading a Bitmap into an ImageView.

You have some options:

1. Use a large application heap

Add the "largeHeap" option to the application tag in your AndroidManifest.xml. This will make more memory available to your app but will likely not fix the root issue.

```
<application largeHeap="true" ... >
```

2. Recycle your bitmaps

After loading a bitmap, be sure to recycle it and free up memory:

```
    if (bitmap != null && !bitmap.isRecycled())
       bitmap.recycle();
```

3. Load sampled bitmaps into memory

Avoid loading the entire bitmap into memory at once by sampling a reduced size, using BitmapOptions and inSampleSize.

See Android documentation for example

Section 262.3: Registering own Handler for unexpected exceptions

This is how you can react to exceptions which have not been catched, similar to the system's standard "Application XYZ has crashed"

```
import android.app.Application;
import android.util.Log;

import java.io.File;
```

```
import java.io.FileWriter;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

/**
 * Application class writing unexpected exceptions to a crash file before crashing.
 */
public class MyApplication extends Application {
    private static final String TAG = "ExceptionHandler";

    @Override
    public void onCreate() {
        super.onCreate();

        // Setup handler for uncaught exceptions.
        final Thread.UncaughtExceptionHandler defaultHandler =
Thread.getDefaultUncaughtExceptionHandler();
        Thread.setDefaultUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {
            @Override
            public void uncaughtException(Thread thread, Throwable e) {
                try {
                    handleUncaughtException(e);
                    System.exit(1);
                } catch (Throwable e2) {
                    Log.e(TAG, "Exception in custom exception handler", e2);
                    defaultHandler.uncaughtException(thread, e);
                }
            }
        });
    }

    private void handleUncaughtException(Throwable e) throws IOException {
        Log.e(TAG, "Uncaught exception logged to local file", e);

        // Create a new unique file
        final DateFormat dateFormat =  new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss", Locale.US);
        String timestamp;
        File file = null;
        while (file == null || file.exists()) {
            timestamp = dateFormat.format(new Date());
            file = new File(getFilesDir(), "crashLog_" + timestamp + ".txt");
        }
        Log.i(TAG, "Trying to create log file " + file.getPath());
        file.createNewFile();

        // Write the stacktrace to the file
        FileWriter writer = null;
        try {
            writer = new FileWriter(file, true);
            for (StackTraceElement element : e.getStackTrace()) {
                writer.write(element.toString());
            }
        } finally {
            if (writer != null) writer.close();
        }

        // You can (and probably should) also display a dialog to notify the user
    }
}
```

Then register this Application class in your AndroidManifest.xml:

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

Section 262.4: UncaughtException

If you want to handle uncaught exceptions try to catch them all in onCreate method of you Application class:

```
public class MyApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        try {
            Thread
                .setDefaultUncaughtExceptionHandler(
                            new Thread.UncaughtExceptionHandler() {

                @Override
                public void uncaughtException(Thread thread, Throwable e) {
                    Log.e(TAG,
                            "Uncaught Exception thread: "+thread.getName()+"
                             "+e.getStackTrace());
                    handleUncaughtException (thread, e);
                }
            });
        } catch (SecurityException e) {
            Log.e(TAG,
                    "Could not set the Default Uncaught Exception Handler:"
                    +e.getStackTrace());
        }
    }

    private void handleUncaughtException (Thread thread, Throwable e){
        Log.e(TAG, "uncaughtException:");
        e.printStackTrace();
    }
}
```

Section 262.5: NetworkOnMainThreadException

From the documentation:

> The exception that is thrown when an application attempts to perform a networking operation on its main thread.
>
> This is only thrown for applications targeting the Honeycomb SDK or higher. Applications targeting earlier SDK versions are allowed to do networking on their main event loop threads, but it's heavily discouraged.

Here's an example of a code fragment that may cause that exception:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        Uri.Builder builder = new Uri.Builder().scheme("http").authority("www.google.com");
        HttpURLConnection urlConnection = null;
        BufferedReader reader = null;
        URL url;
        try {
            url = new URL(builder.build().toString());
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();
        } catch (IOException e) {
            Log.e("TAG","Connection error", e);
        } finally{
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (reader != null) {
                try {
                    reader.close();
                } catch (final IOException e) {
                    Log.e("TAG", "Error closing stream", e);
                }
            }
        }
    }
}
```

Above code will throw `NetworkOnMainThreadException` for applications targeting Honeycomb SDK (Android v3.0) or higher as the application is trying to perform a network operation on the main thread.

To avoid this exception, your network operations must always run in a background task via an `AsyncTask`, `Thread`, `IntentService`, etc.

```
private class MyAsyncTask extends AsyncTask<String, Integer, Void> {

    @Override
    protected Void doInBackground(String[] params) {
        Uri.Builder builder = new Uri.Builder().scheme("http").authority("www.google.com");
        HttpURLConnection urlConnection = null;
        BufferedReader reader = null;
        URL url;
        try {
            url = new URL(builder.build().toString());
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();
        } catch (IOException e) {
            Log.e("TAG","Connection error", e);
        } finally{
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (reader != null) {
                try {
                    reader.close();
                } catch (final IOException e) {
                    Log.e("TAG", "Error closing stream", e);
                }
            }
        }

        return null;
```

```
    }
}
```

Section 262.6: DexException

```
com.android.dex.DexException: Multiple dex files define Lcom/example/lib/Class;
```

This error occurs because the app, when packaging, finds two `.dex` files that define the same set of methods.

Usually this happens because the app has accidentally acquired 2 separate dependencies on the same library.

For instance, say you have a project, and you want to rely on two libraries: `A` and `B`, which each have their own dependencies. If library `B` already has a dependency on library `A`, this error will be thrown if library `A` is added to the project by itself. Compiling library `B` already gave the set of code from `A`, so when the compiler goes to bundle library `A`, it finds library `A`'s methods already packaged.

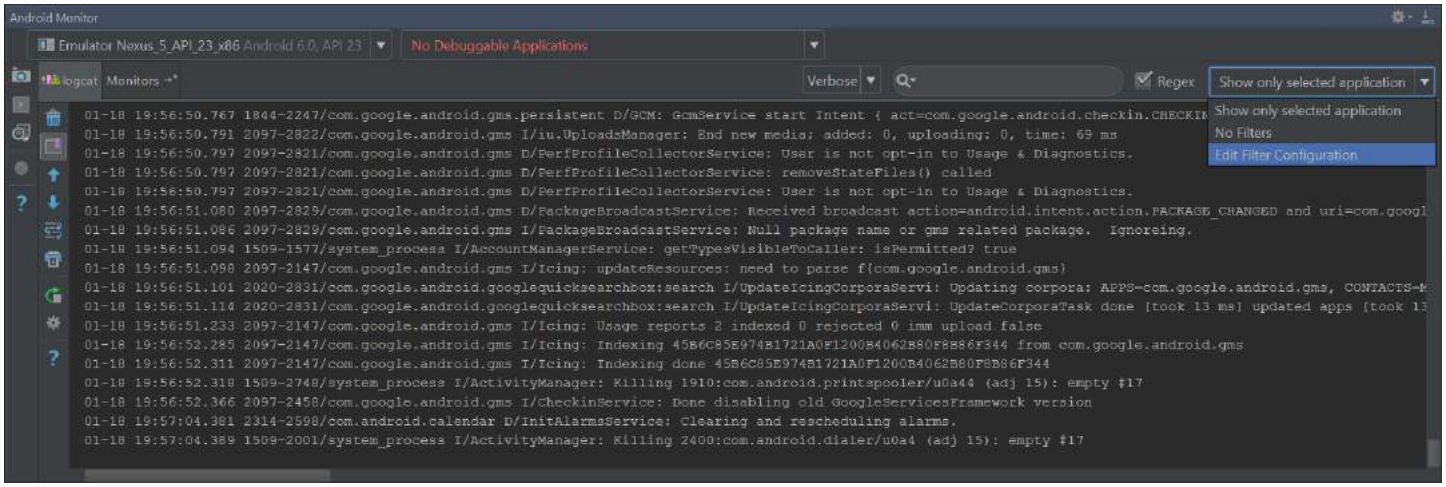
To resolve, make sure that none of your dependencies could accidentally be added twice in such a manner

Chapter 263: Logging and using Logcat

Option	Description
-b (buffer)	Loads an alternate log buffer for viewing, such as events or radio. The main buffer is used by default. See Viewing Alternative Log Buffers.
-c	Clears (flushes) the entire log and exits.
-d	Dumps the log to the screen and exits.
-f (filename)	Writes log message output to (filename). The default is stdout.
-g	Prints the size of the specified log buffer and exits.
-n (count)	Sets the maximum number of rotated logs to (count). The default value is 4. Requires the -r option.
-r (kbytes)	Rotates the log file every (kbytes) of output. The default value is 16. Requires the -f option.
-s	Sets the default filter spec to silent.
-v (format)	Sets the output format for log messages. The default is brief format.

Section 263.1: Filtering the logcat output

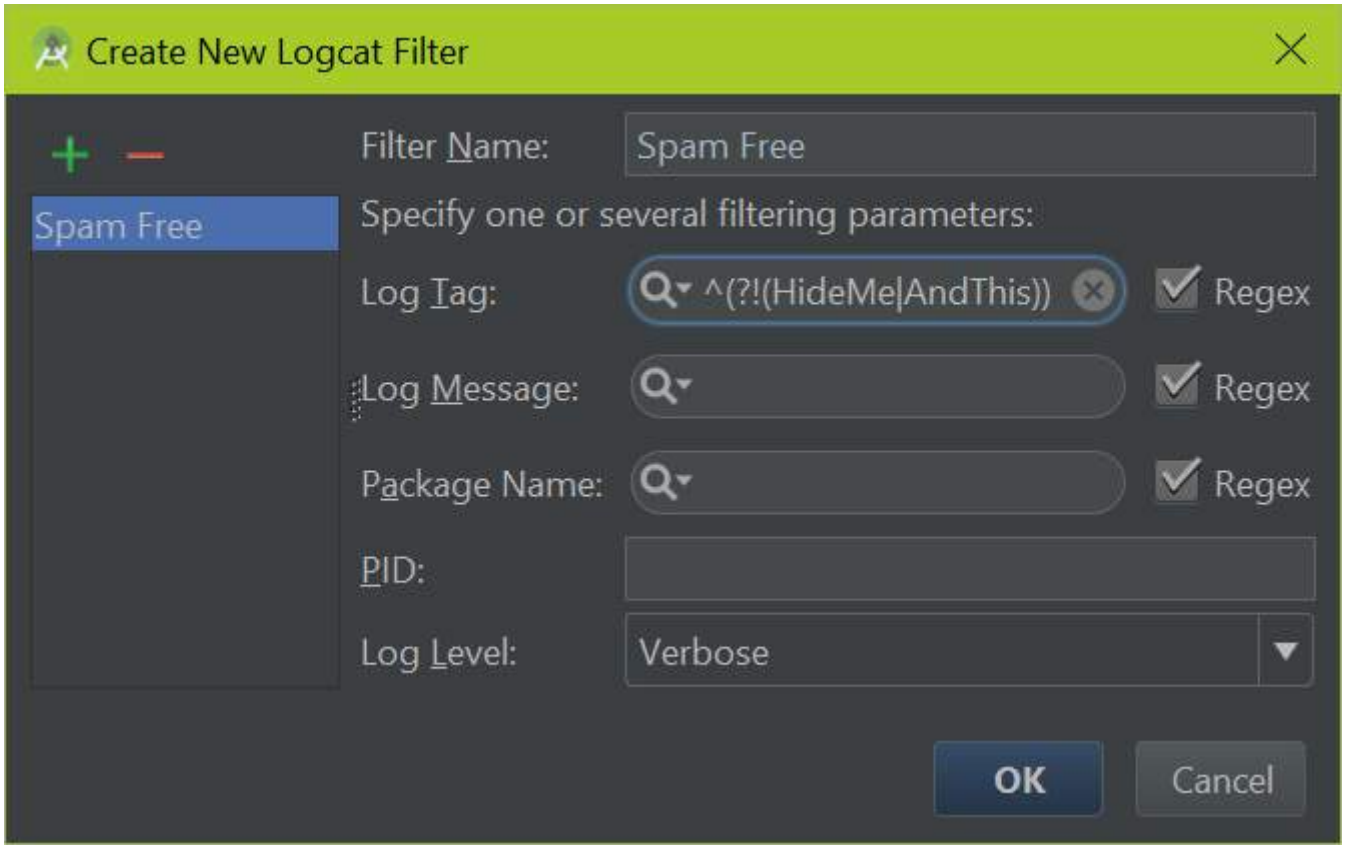
It is helpful to filter the logcat output because there are many messages which are not of interest. To filter the output, open the "Android Monitor" and click on the drop down on the top-right and select *Edit Filter Configuration*



Now you can add custom filters to show messages which are of interest, as well as filter out well-known log lines which can safely be ignored. To ignore a part of the output you may define a Regular Expression. Here is an example of excluding matching tags:

```
^(?!(HideMe|AndThis))
```

This can be entered by following this example:



The above is a regular expression which excludes inputs. If you wanted to add another tag to the *blacklist*, add it after a pipe `|` character. For example, if you wanted to blacklist "GC", you would use a filter like this:

```
^(?!(HideMe|AndThis|GC))
```

For more documentation and examples visit Logging and using Logcat

Section 263.2: Logging

Any quality Android application will keep track of what it's doing through application logs. These logs allow easy debugging help for the developer to diagnose what's going on with the application. Full Android Documentation can be found here, but a summary follows:

Basic Logging

The `Log` class is the main source of writing developer logs, by specifying a `tag` and a `message`. The tag is what you can use to filter log messages by to identify which lines come from your particular Activity. Simply call

```
Log.v(String tag, String msg);
```

And the Android system will write a message to the logcat:

```
07-28 12:00:00.759 24812-24839/my.packagename V/MyAnimator: Some log messages
 └ time stamp             |  app.package┘     |     └ any tag  |
    process & thread ids ┘          log level┘               └ the log message
```

> TIP:
> Notice the process id and the thread id. If they are the same - the log is coming from the main/UI thread!

Any tag can be used, but it is common to use the class name as a tag:

```
public static final String tag = MyAnimator.class.getSimpleName();
```

Log Levels
The Android logger has 6 different levels, each of which serve a certain purpose:

- `ERROR`: `Log.e()`
  - Used to indicate critical failure, this is the level printed at when throwing an `Exception`.
- `WARN`: `Log.w()`
  - Used to indicate a warning, mainly for recoverable failures
- `INFO`: `Log.i()`
  - Used to indicate higher-level information about the state of the application
- `DEBUG`: `Log.d()`
  - Used to log information that would be useful to know when debugging the application, but would get in the way when running the application
- `VERBOSE`: `Log.v()`
  - Used to log information that reflects the small details about the state of the application
- **`ASSERT`**: `Log.wtf()`
  - Used to log information about a condition that should never happen.
  - *wtf* stands for "What a Terrible Failure".

Motivation For Logging
The motivation for logging is to easily find errors, warnings, and other information by glancing at the chain of events from the application. For instance, imagine an application that reads lines from a text file, but incorrectly assumes that the file will never be empty. The log trace (of an app that doesn't log) would look something like this:

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

Followed by a bunch of stack traces that would eventually lead to the offending line, where stepping through with a debugger would eventually lead to the problem

However, the log trace of an application with logging enabled could look something like this:

```
V/MyApplication: Looking for file myFile.txt on the SD card
D/MyApplication: Found file myFile.txt at path <path>
V/MyApplication: Opening file myFile.txt
D/MyApplication: Finished reading myFile.txt, found 0 lines
V/MyApplication: Closing file myFile.txt
...
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

A quick glance at the logs and it is obvious that the file was empty.

Things To Considering When Logging:
Although logging is a powerful tool that allows Android developers to gain a greater insight into the inner working of their application, logging does have some drawbacks.

Log Readability:
It is common for Android Applications to have several logs running synchronously. As such, it is very important that each log is easily readable and only contains relevant, necessary information.

Performance:
Logging does require a small amount of system resources. In general, this does not warrant concern, however, if

overused, logging may have a negative impact on application performance.

Security:
Recently, several Android Applications have been added to the Google Play marketplace that allow the user to view logs of all running applications. This unintended display of data may allow users to view confidential information. As a rule of thumb, always remove logs that contain on non-public data *before* publishing your application to the marketplace.

Conclusion:
Logging is an essential part of an Android application, because of the power it gives to developers. The ability to create a useful log trace is one of the most challenging aspects of software development, but Android's Log class helps to make it much easier.

For more documentation and examples visit Logging and using Logcat

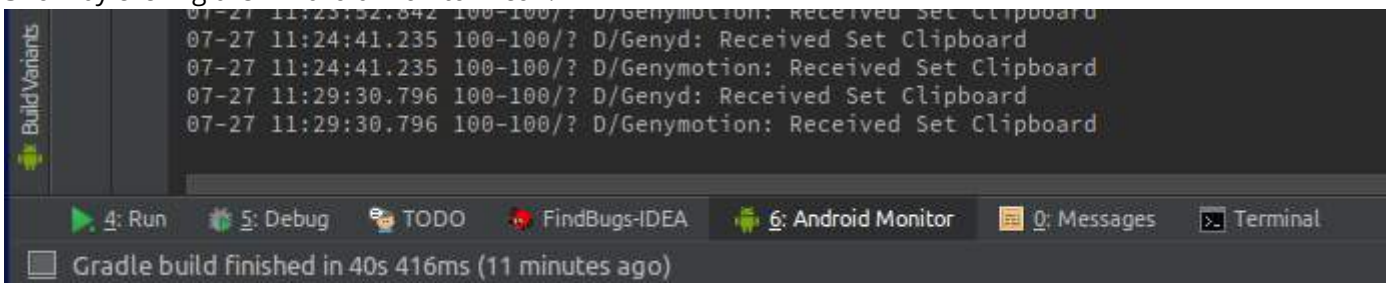
Section 263.3: Using the Logcat

Logcat is a command-line tool that dumps a log of system messages, including stack traces when the device throws an error and messages that you have written from your app with the Log class.

The Logcat output can be displayed within Android Studio's Android Monitor or with adb command line.

In Android Studio

Show by clicking the "Android Monitor" icon:



Or by pressing `Alt` + `6` on Windows/Linux or `CMD` + `6` on Mac.

via command line:

Simple usage:

```
$ adb logcat
```

With timestamps:

```
$ adb logcat -v time
```

Filter on specific text:

```
$ adb logcat -v time | grep 'searchtext'
```

There are many options and filters available to *command line logcat*, documented here.
A simple but useful example is the following filter expression that displays all log messages with priority level "error", on all tags:

```
$ adb logcat *:E
```

Section 263.4: Log with link to source directly from Logcat

This is a nice trick to add a link to code, so it will be easy to jump to the code that issued the log.

With the following code, this call:

```
MyLogger.logWithLink("MyTag","param="+param);
```

Will result in:

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3)  (MyFrag.java:2366) // << logcat converts
this to a link to source!
```

This is the code (inside a class called MyLogger):

```
static StringBuilder sb0 = new StringBuilder(); // reusable string object

public static void logWithLink(String TAG, Object param) {
    StackTraceElement stack = Thread.currentThread().getStackTrace()[3];
    sb0.setLength(0);
    String c = stack.getFileName().substring(0, stack.getFileName().length() - 5); // removes the
".java"
    sb0.append(c).append(":");
    sb0.append(stack.getMethodName()).append('(');
    if (param != null) {
        sb0.append(param);
    }
    sb0.append(") ");
    sb0.append("
(").append(stack.getFileName()).append(':').append(stack.getLineNumber()).append(')');
    Log.d(TAG, sb0.toString());
}
```

This is a basic example, it can be easily extended to issue a link to the caller (hint: the stack will be [4] instead of [3]), and you can also add other relevant information.

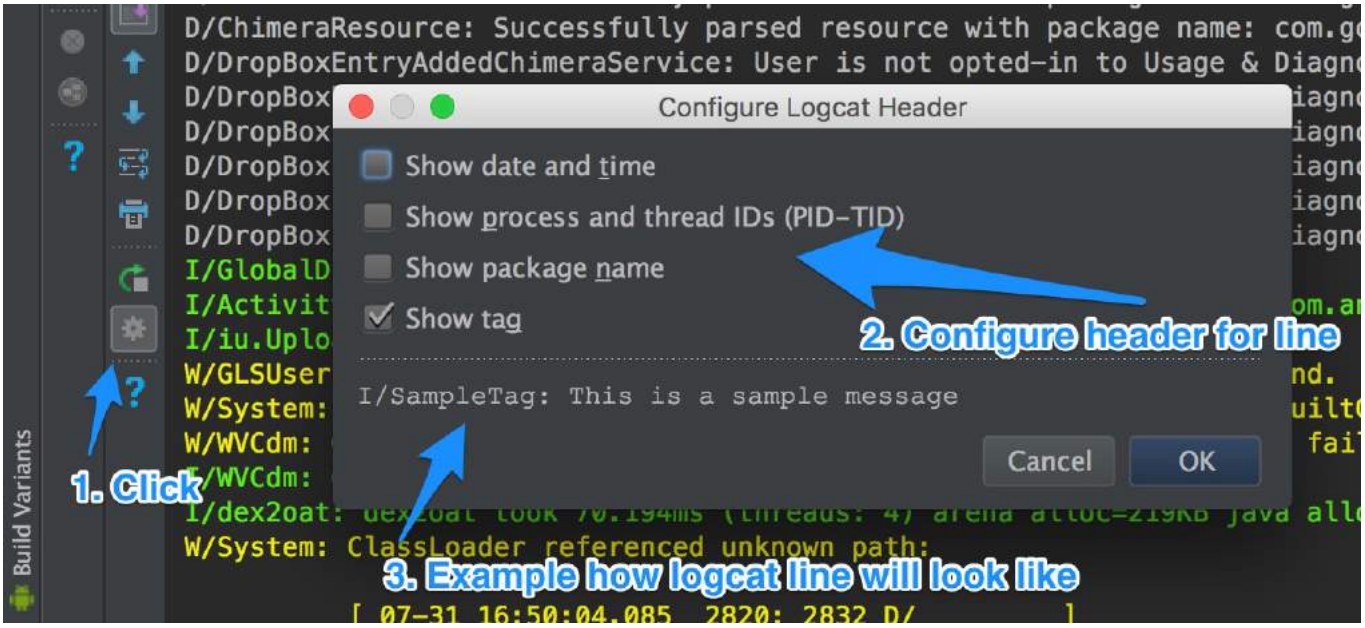
Section 263.5: Clear logs

In order to clear (flush) the entire log:

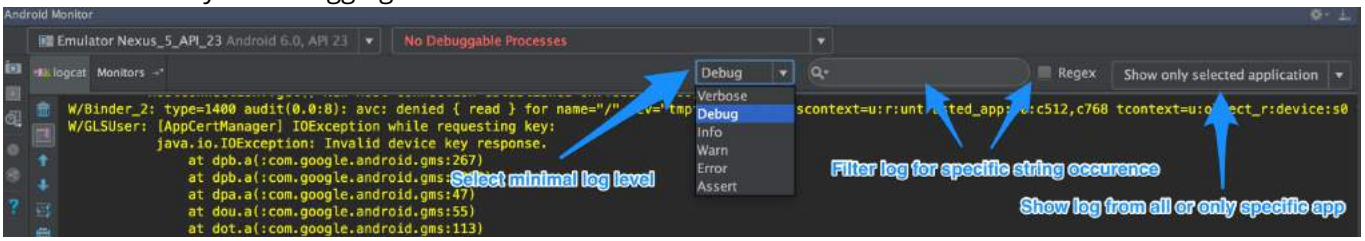
```
adb logcat -c
```

Section 263.6: Android Studio usage

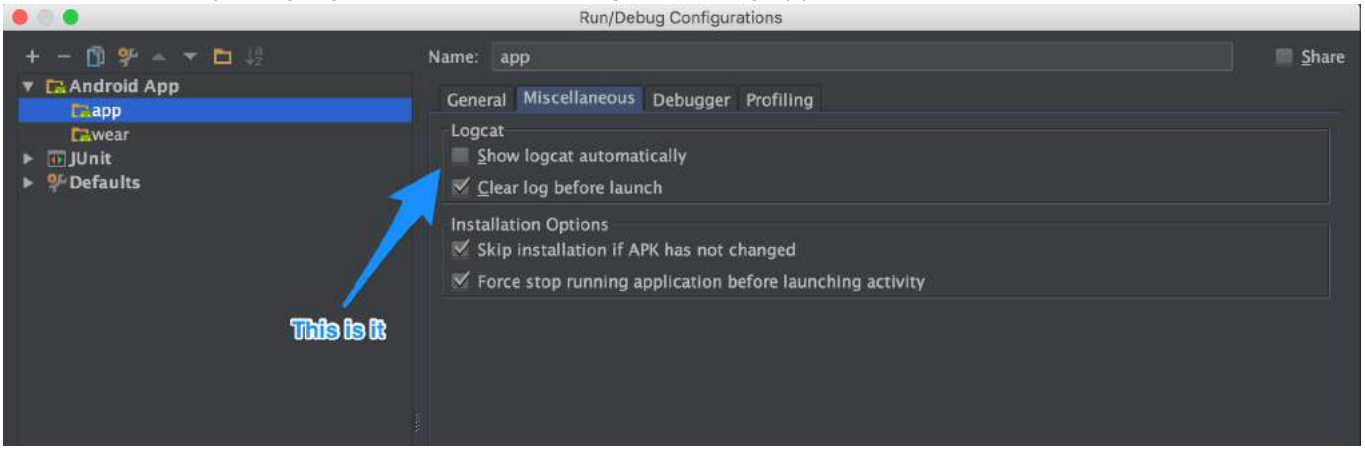
1. Hide/show printed information:



2. Control verbosity of the logging:



3. Disable/enable opening log window when starting run/debug application



Section 263.7: Generating Logging code

`Android Studio`'s `Live templates` can offer quite a few shortcuts for quick logging.
To use Live templates, all you need to do is to start typing the template name, and hit `TAB` or enter to insert the statement.

Examples:

- `logi` → turns into → `android.util.Log.i(TAG, "$METHOD_NAME$: $content$");`
  - `$METHOD_NAME$` will automatically be replaced with your method name, and the cursor will wait for the content to be filled.
- `loge` → same, for error
- etc. for the rest of the logging levels.

Full list of templates can be found in `Android Studio`'s settings (`ALT`+`s` and type "live"). And it is possible to

add your custom templates as well.

If you find `Android Studio`'s `Live templates` not enough for your needs, you can consider Android Postfix Plugin

This is a very useful library which helps you to avoid writing the logging line manually.

The syntax is absolutely simple:

.log - Logging. If there is constant variable "TAG", it use "TAG" . Else it use class name.

```
ublic class MyActivity extends Activity {
  static final String T= "test";
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      new View.OnClickListener() {
          @Override
          public void onClick(View view) {

          }
      };
  }
```

Chapter 264: ADB (Android Debug Bridge)

ADB (Android Debug Bridge) is a command line tool that used to communicate with an emulator instance or connected Android device.

Overview of ADB

A large portion of this topic was split out to adb shell

Section 264.1: Connect ADB to a device via WiFi

The standard ADB configuration involves a USB connection to a physical device.
If you prefer, you can switch over to TCP/IP mode, and connect ADB via WiFi instead.

Not rooted device

1. **Get on the same network:**
    - Make sure your device and your computer are on the same network.
2. **Connect the device to the host computer with a USB cable.**
3. **Connect `adb` to device over network:**

   While your device is connected to `adb` via USB, do the following command to listen for a TCP/IP connection on a port (default 5555):

    - Type `adb tcpip <port>` (switch to TCP/IP mode).
    - Disconnect the USB cable from the target device.
    - Type `adb connect <ip address>:<port>` (port is optional; default 5555).

   For example:

```
adb tcpip 5555
adb connect 192.168.0.101:5555
```

   If you don't know your device's IP you can:

    - check the IP in the WiFi settings of your device.
    - use ADB to discover IP (via USB):
        1. Connect the device to the computer via USB
        2. In a command line, type `adb shell ifconfig` and copy your device's IP address

   To **revert back** to debugging via **USB** use the following command:

```
adb usb
```

   You can also connect ADB via WiFi by installing a plugin to Android Studio. In order to do so, go to *Settings* > *Plugins* and Browse repositories, search for *ADB WiFi*, install it, and reopen Android Studio. You will see a new icon in your toolbar as shown in the following image. Connect the device to the host computer via USB and click on this *AndroidWiFiADB* icon. It will display a message whether your device is connected or not. Once it gets connected you can unplug your USB.

Rooted device

Note: Some devices which **are rooted** can use the ADB WiFi App from the Play Store to enable this in a simple way. Also, for certain devices (especially those with CyanogenMod ROMs) this option is present in the Developer Options among the Settings. Enabling it will give you the IP address and port number required to connect to `adb` by simply executing `adb connect <ip address>:<port>`.

When you have a rooted device but don't have access to a USB cable

The process is explained in detail in the following answer:
http://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727
The most important commands are shown below.

Open a terminal in the device and type the following:

```
su
setprop service.adb.tcp.port <a tcp port number>
stop adbd
start adbd
```

For example:

```
setprop service.adb.tcp.port 5555
```

And on your computer:

```
adb connect <ip address>:<a tcp port number>
```

For example:

```
adb connect 192.168.1.2:5555
```

To turn it off:

```
setprop service.adb.tcp.port -1
stop adbd
start adbd
```

Avoid timeout

By default `adb` will timeout after 5000 ms. This can happen in some cases such as slow WiFi or large APK.

A simple change in the Gradle configuration can do the trick:

```
android {
    adbOptions {
        timeOutInMs 10 * 1000
    }
}
```

Section 264.2: Direct ADB command to specific device in a multi-device setting

1. Target a device by serial number

Use the `-s` option followed by a device name to select on which device the `adb` command should run. The `-s` options should be first in line, **before** the command.

```
adb -s <device> <command>
```

Example:

```
adb devices

List of devices attached
emulator-5554       device
02157df2d1faeb33    device

adb -s emulator-5554 shell
```

Example#2:

```
adb devices -l

List of devices attached
06157df65c6b2633    device usb:1-3 product:zerofltexx model:SM_G920F device:zeroflte
LC62TB413962        device usb:1-5 product:a50mgp_dug_htc_emea model:HTC_Desire_820G_dual_sim
device:htc_a50mgp_dug

adb -s usb:1-3 shell
```

2. Target a device, when only one device type is connected

You can target the only running emulator with -e

```
adb -e <command>
```

Or you can target the only connected USB device with -d

```
adb -d <command>
```

Section 264.3: Taking a screenshot and video (for kitkat only) from a device display

Screen shot: Option 1 (pure adb)

The `shell` adb command allows us to execute commands using a device's built-in shell. The `screencap` shell command captures the content currently visible on a device and saves it into a given image file, e.g. `/sdcard/screen.png`:

```
adb shell screencap /sdcard/screen.png
```

You can then use the pull command to download the file from the device into the current directory on you computer:

```
adb pull /sdcard/screen.png
```

Screen shot:Option 2 (faster)

Execute the following one-liner:

(Marshmallow and earlier):

```
adb shell screencap -p | perl -pe 's/\x0D\x0A/\x0A/g' > screen.png
```

(Nougat and later):

```
adb shell screencap -p > screen.png
```

The `-p` flag redirects the output of the `screencap` command to stdout. The Perl expression this is piped into cleans up some end-of-line issues on Marshmallow and earlier. The stream is then written to a file named `screen.png` within the current directory. See this article and this article for more information.

Video

this only work in KitKat and via ADB only. This not Working below Kitkat To start recording your device's screen, run the following command:

`adb shell screenrecord /sdcard/example.mp4`, This command will start recording your device's screen using the default settings and save the resulting video to a file at `/sdcard/example.mp4` file on your device.

When you're done recording, press Ctrl+C (z in Linux) in the Command Prompt window to stop the screen recording. You can then find the screen recording file at the location you specified. Note that the screen recording is saved to your device's internal storage, not to your computer.

The default settings are to use your device's standard screen resolution, encode the video at a bitrate of 4Mbps, and set the maximum screen recording time to 180 seconds. For more information about the command-line options you can use, run the following command:

`adb shell screenrecord –help`, This works without rooting the device. Hope this helps.

Section 264.4: Pull (push) files from (to) the device

You may pull (download) files from the device by executing the following command:

```
adb pull <remote> <local>
```

For example:

```
adb pull /sdcard/ ~/
```

You may also push (upload) files from your computer to the device:

```
adb push <local> <remote>
```

For example:

```
adb push ~/image.jpg /sdcard/
```

Example to Retrieve Database from device

```
sudo adb -d shell "run-as com.example.name cat /data/da/com.example.name /databases/DATABASE_NAME
 > /sdcard/file
```

Section 264.5: Print verbose list of connected devices

To get a verbose list of all devices connected to `adb`, write the following command in your terminal:

```
adb devices -l
```

Example Output

```
List of devices attached
ZX1G425DC6             device usb:336592896X product:shamu model:Nexus_6 device:shamu
013e4e127e59a868       device usb:337641472X product:bullhead model:Nexus_5X device:bullhead
ZX1D229KCN             device usb:335592811X product:titan_retde model:XT1068 device:titan_umtsds
A50PL                  device usb:331592812X
```

- The first column is the serial number of the device. If it starts with `emulator-`, this device is an emulator.
- `usb:` the path of the device in the USB subsystem.
- `product:` the product code of the device. This is very manufacturer-specific, and as you can see in the case of the Archos device `A50PL` above, it can be blank.
- `model:` the device model. Like `product`, can be empty.
- `device:` the device code. This is also very manufacturer-specific, and can be empty.

Section 264.6: View logcat

You can run `logcat` as an adb command or directly in a shell prompt of your emulator or connected device. To view log output using `adb`, navigate to your SDK platform-tools/ directory and execute:

```
$ adb logcat
```

Alternatively, you can create a shell connection to a device and then execute:

```
$ adb shell
$ logcat
```

One useful command is:

```
adb logcat -v threadtime
```

This displays the date, invocation time, priority, tag, and the PID and TID of the thread issuing the message in a long message format.

Filtering

Logcat logs got so called log levels:

> **V** — Verbose, **D** — Debug, **I** — Info, **W** — Warning, **E** — Error, **F** — Fatal, **S** — Silent

You can filter logcat by log level as well. For instance if you want only to output Debug level:

```
adb logcat *:D
```

Logcat can be filtered by a package name, of course you can combine it with the log level filter:

```
adb logcat <package-name>:<log level>
```

You can also filter the log using grep (more on filtering logcat output here):

```
adb logcat | grep <some text>
```

In Windows, filter can be used using findstr, for example:

```
adb logcat | findstr <some text>
```

To view alternative log buffer [main|events|radio], run the `logcat` with the `-b` option:

```
adb logcat -b radio
```

Save output in file :

```
adb logcat > logcat.txt
```

Save output in file while also watching it:

```
adb logcat | tee logcat.txt
```

Cleaning the logs:

```
adb logcat -c
```

Section 264.7: View and pull cache files of an app

You may use this command for listing the files for your own debuggable apk:

```
adb shell run-as <sample.package.id> ls /data/data/sample.package.id/cache
```

And this script for pulling from cache, this copy the content to sdcard first, pull and then remove it at the end:

```
#!/bin/sh
adb shell "run-as <sample.package.id> cat '/data/data/<sample.package.id>/$1' > '/sdcard/$1'"
adb pull "/sdcard/$1"
adb shell "rm '/sdcard/$1'"
```

Then you can pull a file from cache like this:

```
./pull.sh cache/someCachedData.txt
```

Get Database file via ADB

```
sudo adb -d shell "run-as com.example.name cat /data/da/com.example.name
/databases/STUDENT_DATABASE  > /sdcard/file
```

Section 264.8: Clear application data

One can clear the user data of a specific app using `adb`:

```
adb shell pm clear <package>
```

This is the same as to browse the settings on the phone, select the app and press on the clear data button.

- `pm` invokes the package manager on the device
- `clear` deletes all data associated with a package

Section 264.9: View an app's internal data (data/data/<sample.package.id>) on a device

First, make sure your app can be backed up in `AndroidManifest.xml`, i.e. `android:allowBackup` is not **`false`**.

Backup command:

```
adb -s <device_id> backup -noapk <sample.package.id>
```

Create a tar with dd command:

```
dd if=backup.ab bs=1 skip=24 | python -c "import
zlib,sys;sys.stdout.write(zlib.decompress(sys.stdin.read()))" > backup.tar
```

Extract the tar:

```
tar -xvf backup.tar
```

You may then view the extracted content.

Section 264.10: Install and run an application

To install an APK file, use the following command:

```
adb install path/to/apk/file.apk
```

or if the app is existing and we want to reinstall

```
adb install -r path/to/apk/file.apk
```

To uninstall an application, we have to specify its package

```
adb uninstall application.package.name
```

Use the following command to start an app with a provided package name (or a specific activity in an app):

```
adb shell am start -n adb shell am start <package>/<activity>
```

For example, to start Waze:

```
adb shell am start -n adb shell am start com.waze/com.waze.FreeMapAppActivity
```

Section 264.11: Sending broadcast

It's possible to send broadcast to `BroadcastReceiver` with `adb`.

In this example we are sending broadcast with action `com.test.app.ACTION` and string extra in bundle `'foo'='bar'`:

```
adb shell am broadcast -a action com.test.app.ACTION --es foo "bar"
```

You can put any other supported type to bundle, not only strings:

> --ez - boolean
> --ei - integer
> --el - long
> --ef - float
> --eu - uri
> --eia - int array (separated by ',')
> --ela - long array (separated by ',')
> --efa - float array (separated by ',')
> --esa - string array (separated by ',')

To send intent to specific package/class `-n` or `-p` parameter can be used.
Sending to package:

```
-p com.test.app
```

Sending to a specific component (`SomeReceiver` class in `com.test.app` **package**):

```
-n com.test.app/.SomeReceiver
```

Useful examples:

- Sending a "boot complete" broadcast
- Sending a "time changed" broadcast after setting time via adb command

Section 264.12: Backup

You can use the `adb backup` command to backup your device.

```
adb backup [-f <file>] [-apk|-noapk] [-obb|-noobb] [-shared|-noshared] [-all]
           [-system|nosystem] [<packages...>]
```

`-f <filename>` specify filename **default:** *creates backup.ab in the current directory*

`-apk|noapk` enable/disable backup of .apks themself **default:** *-noapk*

`-obb|noobb` enable/disable backup of additional files **default:** *-noobb*

`-shared|noshared` backup device's shared storage / SD card contents **default:** *-noshared*

`-all` backup all installed apllications

`-system|nosystem` include system applications **default:** *-system*

<packages> a list of packages to be backed up (e.g. com.example.android.myapp) (not needed if `-all` is specified)

For a full device backup, including everything, use

```
adb backup -apk -obb -shared -all -system -f fullbackup.ab
```

> **Note:** Doing a full backup can take a long time.

In order to restore a backup, use

```
adb restore backup.ab
```

Section 264.13: View available devices

Command:

```
 adb devices
```

Result example:

```
List of devices attached
emulator-5554  device
PhoneRT45Fr54  offline
123.454.67.45  no device
```

First column - device serial number

Second column - connection status

Android documentation

Section 264.14: Connect device by IP

Enter these commands in Android device Terminal

```
su
setprop service.adb.tcp.port 5555
stop adbd
start adbd
```

After this, you can use **CMD** and **ADB** to connect using the following command

```
adb connect 192.168.0.101.5555
```

And you can disable it and return ADB to listening on USB with

```
setprop service.adb.tcp.port -1
stop adbd
start adbd
```

From a computer, if you have USB access already (no root required)

It is even easier to switch to using Wi-Fi, if you already have USB. From a command line on the computer that has the device connected via USB, issue the commands

```
adb tcpip 5555
adb connect 192.168.0.101:5555
```

Replace 192.168.0.101 with device IP

Section 264.15: Install ADB on Linux system

How to install the Android Debugging Bridge (ADB) to a Linux system with the terminal using your distro's repositories.

Install to Ubuntu/Debian system via apt:

```
sudo apt-get update
sudo apt-get install adb
```

Install to Fedora/CentOS system via yum:

```
sudo yum check-update
sudo yum install android-tools
```

Install to Gentoo system with portage:

```
sudo emerge --ask dev-util/android-tools
```

Install to openSUSE system with zypper:

```
sudo zypper refresh
sudo zypper install android-tools
```

Install to Arch system with pacman:

```
sudo pacman -Syyu
sudo pacman -S android-tools
```

Section 264.16: View activity stack

```
adb -s <serialNumber> shell dumpsys activity activities
```

Very useful when used together with the `watch` unix command:

```
watch -n 5 "adb -s <serialNumber> shell dumpsys activity activities | sed -En -e '/Stack #/p' -e
'/Running activities/,/Run #0/p'"
```

Section 264.17: Reboot device

You can reboot your device by executing the following command:

```
adb reboot
```

Perform this command to reboot into bootloader:

```
adb reboot bootloader
```

Reboot to recovery mode:

```
adb reboot recovery
```

Be aware that the device won't shutdown first!

Section 264.18: Read device information

Write the following command in your terminal:

```
adb shell getprop
```

This will print all available information in the form of key/value pairs.

You can just read specific information by appending the name of a specific key to the command. For example:

```
adb shell getprop ro.product.model
```

Here are a few interesting pieces of information that you cat get:

- `ro.product.model`: Model name of the device (e.g. Nexus 6P)
- `ro.build.version.sdk`: API Level of the device (e.g. 23)
- `ro.product.brand`: Branding of the device (e.g. Samsung)

Full Example Output

```
```

Section 264.19: List all permissions that require runtime grant from users on Android 6.0

```
adb shell pm list permissions -g -d
```

Section 264.20: Turn on/off Wifi

Turn on:

```
adb shell svc wifi enable
```

Turn off:

```
adb shell svc wifi disable
```

Section 264.21: Start/stop adb

Start ADB:

```
adb kill-server
```

Stop ADB:

```
adb start-server
```

Chapter 265: Localization with resources in Android

Section 265.1: Configuration types and qualifier names for each folder under the "res" directory

Each resource directory under the `res` folder (listed in the example above) can have different variations of the contained resources in similarly named directory suffixed with different `qualifier-values` for each `configuration-type`.

Example of variations of `` directory with different qualifier values suffixed which are often seen in our android projects:

- drawable/
- drawable-en/
- drawable-fr-rCA/
- drawable-en-port/
- drawable-en-notouch-12key/
- drawable-port-ldpi/
- drawable-port-notouch-12key/

Exhaustive list of all different configuration types and their qualifier values for android resources:

Configuration	Qualifier Values
MCC and MNC	Examples: mcc310 mcc310-mnc004 mcc208-mnc00 etc.
Language and region	Examples: en fr en-rUS fr-rFR fr-rCA
Layout Direction	ldrtl ldltr
smallestWidth	swdp Examples: sw320dp sw600dp sw720dp
Available width	wdp w720dp w1024dp
Available height	hdp h720dp h1024dp
Screen size	small

	normal
	large
	xlarge
Screen aspect	long
	notlong
Round screen	round
	notround
Screen orientation	port
	land
UI mode	car
	desk
	television
	appliancewatch
Night mode	night
	notnight
Screen pixel density (dpi)	ldpi
	mdpi
	hdpi
	xhdpi
	xxhdpi
	xxxhdpi
	nodpi
	tvdpi
	anydpi
Touchscreen type	notouch
	finger
Keyboard availability	keysexposed
	keyshidden
	keyssoft
Primary text input method	nokeys
	qwerty
	12key
Navigation key availability	navexposed
	navhidden
Primary non-touch navigation method	nonav
	dpad
	trackball
	wheel
Platform Version (API level)	Examples:
	v3
	v4
	v7

Section 265.2: Adding translation to your Android app

You have to create a different `strings.xml` file for every new language.

1. Right-click on the *res* folder
2. Choose *New → Values resource file*
3. Select a locale from the available qualifiers
4. Click on the *Next* button (>>)
5. Select a language
6. Name the file *strings.xml*

strings.xml

```
<resources>
    <string name="app_name">Testing Application</string>
    <string name="hello">Hello World</string>
</resources>
```

strings.xml(hi)

```
<resources>
    <string name="app_name">परीक्षण आवेदन</string>
    <string name="hello">नमस्ते दुनिया</string>
</resources>
```

Setting the language programmatically:

```
public void setLocale(String locale) // Pass "en","hi", etc.
{
    myLocale = new Locale(locale);
    // Saving selected locale to session - SharedPreferences.
    saveLocale(locale);
    // Changing locale.
    Locale.setDefault(myLocale);
    android.content.res.Configuration config = new android.content.res.Configuration();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        config.setLocale(myLocale);
    } else {
        config.locale = myLocale;
    }
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {
        getBaseContext().createConfigurationContext(config);
    } else {
        getBaseContext().getResources().updateConfiguration(config,
getBaseContext().getResources().getDisplayMetrics());
    }
}
```

The function above will change the text fields which are referenced from *strings.xml*. For example, assume that you have the following two text views:

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/app_name"/>
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/hello"/>
```

Then, after changing the locale, the language strings having the ids `app_name` and `hello` will be changed accordingly.

Section 265.3: Type of resource directories under the "res" folder

When localizing different types of resources are required, each of which has its own home in the android project structure. Following are the different directories that we can place under the `\res` directory. The resource types placed in each of these directories are explained in the table below:

Directory	Resource Type
animator/	XML files that define property animations.
anim/	XML files that define tween animations. (Property animations can also be saved in this directory, but the animator/ directory is preferred for property animations to distinguish between the two types.)
color/	XML files that define a state list of colors. See Color State List Resource
drawable/	"Bitmap files (.png, .9.png, .jpg, .gif) or XML files that are compiled into the following drawable resource subtypes: : `Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - "`
mipmap/	Drawable files for different launcher icon densities. For more information on managing launcher icons with mipmap/ folders, see Managing Projects Overview.
layout/	XML files that define a user interface layout. See Layout Resource.
menu/	XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu. See Menu Resource.
raw/	Arbitrary files to save in their raw form. To open these resources with a raw InputStream, call Resources.openRawResource() with the resource ID, which is R.raw.filename. However, if you need access to original file names and file hierarchy, you might consider saving some resources in the assets/ directory (instead ofres/raw/). Files in assets/ are not given a resource ID, so you can read them only using AssetManager.
values/	XML files that contain simple values, such as strings, integers, and colors, as well as styles and themes
xml/	Arbitrary XML files that can be read at runtime by calling Resources.getXML(). Various XML configuration files must be saved here, such as a searchable configuration.

Section 265.4: Change locale of android application programmatically

In above examples you understand how to localize resources of application. Following example explain how to change the application locale within application, not from device. In order to change Application locale only, you can use below locale util.

```
import android.app.Application;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.content.res.Resources;
import android.os.Build;
import android.preference.PreferenceManager;
import android.view.ContextThemeWrapper;

import java.util.Locale;

/**
 * Created by Umesh on 10/10/16.
 */
public class LocaleUtils {

    private static Locale mLocale;

    public static void setLocale(Locale locale){
```

```
        mLocale = locale;
        if(mLocale != null){
            Locale.setDefault(mLocale);
        }
    }

    public static void updateConfiguration(ContextThemeWrapper wrapper){
        if(mLocale != null && Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1){
            Configuration configuration = new Configuration();
            configuration.setLocale(mLocale);
            wrapper.applyOverrideConfiguration(configuration);
        }
    }

    public static void updateConfiguration(Application application, Configuration configuration){
        if(mLocale != null && Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN_MR1){
            Configuration config = new Configuration(configuration);
            config.locale = mLocale;
            Resources res = application.getBaseContext().getResources();
            res.updateConfiguration(configuration, res.getDisplayMetrics());
        }
    }

    public static void updateConfiguration(Context context, String language, String country){
        Locale locale = new Locale(language,country);
        setLocale(locale);
        if(mLocale != null){
            Resources res = context.getResources();
            Configuration configuration = res.getConfiguration();
            configuration.locale = mLocale;
            res.updateConfiguration(configuration,res.getDisplayMetrics());
        }
    }




    public static String getPrefLangCode(Context context) {
        return PreferenceManager.getDefaultSharedPreferences(context).getString("lang_code","en");
    }

    public static void setPrefLangCode(Context context, String mPrefLangCode) {

        SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
        editor.putString("lang_code",mPrefLangCode);
        editor.commit();
    }

    public static  String getPrefCountryCode(Context context) {
        return
PreferenceManager.getDefaultSharedPreferences(context).getString("country_code","US");
    }

    public static void setPrefCountryCode(Context context,String mPrefCountryCode) {

        SharedPreferences.Editor editor =
PreferenceManager.getDefaultSharedPreferences(context).edit();
        editor.putString("country_code",mPrefCountryCode);
        editor.commit();
    }
}
```

Initialize locale that user preferred, from Application class.

```
public class LocaleApp extends Application{

    @Override
    public void onCreate() {
        super.onCreate();

        LocaleUtils.setLocale(new Locale(LocaleUtils.getPrefLangCode(this),
LocaleUtils.getPrefCountryCode(this)));
        LocaleUtils.updateConfiguration(this, getResources().getConfiguration());
    }
}
```

You also need to create a base activity and extend this activity to all other activity so that you can change locale of application only one place as follows :

```
public abstract class LocalizationActivity extends AppCompatActivity {

    public LocalizationActivity() {
        LocaleUtils.updateConfiguration(this);
    }

    // We only override onCreate
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

}
```

> Note : Always initialize locale in constructor.

Now you can use LocalizationActivity as follow.

```
public class MainActivity extends LocalizationActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }
}
```

> Note: When you change locale of application programmatically, need to restart your activity to take the effect of locale change In order to work properly for this solution you and use locale from shared preferences on app startup you `android:name=".LocaleApp"` in you Manifest.xml.

Sometimes Lint checker prompt to create the release build. To solve such issue follow below options.

First:

If you want to disable translation for some strings only then add following attribute to default string.xml

```
<string name="developer" translatable="false">Developer Name</string>
```

Second:

Ignore all missing translation from resource file add following attribute It's the ignore attribute of the tools namespace in your strings file, as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<resources
  xmlns:tools="http://schemas.android.com/tools"
  tools:ignore="MissingTranslation" >
http://stackoverflow.com/documentation/android/3345/localization-with-resources-in-android#
  <!-- your strings here; no need now for the translatable attribute -->

</resources>
```

Third:

Another way to disable non-translatable string

http://tools.android.com/recent/non-translatablestrings

If you have a lot of resources that should not be translated, you can place them in a file named donottranslate.xml and lint will consider all of them non-translatable resources.

Fourth:

You can also add locale in resource file

```
<resources
xmlns:tools="http://schemas.android.com/tools"
    tools:locale="en" tools:ignore="MissingTranslation">
```

You can also disable missing translation check for lint from app/build.gradle

```
lintOptions {

        disable 'MissingTranslation'
    }
```

Section 265.5: Currency

```
Currency currency = Currency.getInstance("USD");
NumberFormat format = NumberFormat.getCurrencyInstance();
format.setCurrency(currency);
format.format(10.00);
```

Chapter 266: Convert vietnamese string to english string Android

Section 266.1: example:

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

converted:

```
"Le Minh Thoai la nguoi Viet Nam"
```

Section 266.2: Chuyển chuỗi Tiếng Việt thành chuỗi không dấu

```
  public static String convert(String str) {
    str = str.replaceAll("à|á|ạ|ả|ã|â|ầ|ấ|ậ|ẩ|ẫ|ă|ằ|ắ|ặ|ẳ|ẵ", "a");
    str = str.replaceAll("è|é|ẹ|ẻ|ẽ|ê|ề|ế|ệ|ể|ễ", "e");
    str = str.replaceAll("ì|í|ị|ỉ|ĩ", "i");
    str = str.replaceAll("ò|ó|ọ|ỏ|õ|ô|ồ|ố|ộ|ổ|ỗ|ơ|ờ|ớ|ợ|ở|ỡ", "o");
    str = str.replaceAll("ù|ú|ụ|ủ|ũ|ư|ừ|ứ|ự|ử|ữ", "u");
    str = str.replaceAll("ỳ|ý|ỵ|ỷ|ỹ", "y");
    str = str.replaceAll("đ", "d");

    str = str.replaceAll("À|Á|Ạ|Ả|Ã|Â|Ầ|Ấ|Ậ|Ẩ|Ẫ|Ă|Ằ|Ắ|Ặ|Ẳ|Ẵ", "A");
    str = str.replaceAll("È|É|Ẹ|Ẻ|Ẽ|Ê|Ề|Ế|Ệ|Ể|Ễ", "E");
    str = str.replaceAll("Ì|Í|Ị|Ỉ|Ĩ", "I");
    str = str.replaceAll("Ò|Ó|Ọ|Ỏ|Õ|Ô|Ồ|Ố|Ộ|Ổ|Ỗ|Ơ|Ờ|Ớ|Ợ|Ở|Ỡ", "O");
    str = str.replaceAll("Ù|Ú|Ụ|Ủ|Ũ|Ư|Ừ|Ứ|Ự|Ử|Ữ", "U");
    str = str.replaceAll("Ỳ|Ý|Ỵ|Ỷ|Ỹ", "Y");
    str = str.replaceAll("Đ", "D");
    return str;
}
```

Credits

Thank you greatly to all the people from Stack Overflow Documentation who helped provide this content, more changes can be sent to web@petercv.com for new content to be published or updated

0xalihn	Chapters 191 and 231
1SStorm	Chapter 266
3VYZkz7t	Chapter 205
A	Chapter 159
A.A.	Chapter 7
a.ch.	Chapter 8
Aawaz Gyawali	Chapter 231
Abdallah Alaraby	Chapter 41
Abdellah	Chapter 97
abhi	Chapters 41, 47 and 253
Abhishek Jain	Chapters 1, 16, 40, 47, 218, 227 and 258
abhishesh	Chapters 18, 235 and 262
Abilash	Chapter 16
Ab_	Chapter 62
adalPaRi	Chapters 62 and 99
Adam Ratzman	Chapters 78 and 263
adao7000	Chapters 45, 78 and 264
Adarsh Ashok	Chapter 10
Adhikari Bishwash	Chapters 33, 196 and 212
Adil Saiyad	Chapters 63, 119 and 228
Adnan	Chapters 51 and 96
Adrián Pérez	Chapter 13
AesSedai101	Chapter 8
Ahmad Aghazadeh	Chapters 40, 44, 47, 53, 57, 84, 120, 156, 205, 250 and 264
ahmadalibaloch	Chapter 41
Ajit Singh	Chapter 125
Akash Patel	Chapters 16 and 37
Akeshwar Jha	Chapter 248
AL.	Chapter 239
Ala Eddine JEBALI	Chapter 1
alanv	Chapters 83 and 160
Aleks G	Chapter 74
Aleksandar Stefanović	Chapters 1, 8, 37, 43, 83 and 163
Alex	Chapter 59
Alex Bonel	Chapter 16
Alex Chengalan	Chapters 37 and 82
Alex Ershov	Chapter 77
Alex Sullivan	Chapter 95
Alexander Oprisnik	Chapter 143
alexey polusov	Chapters 69, 246 and 263
Ali Sherafat	Chapter 21
Aman Anguralla	Chapter 131
Amit	Chapter 42
Amit Thakkar	Chapter 264
Amod Gokhale	Chapter 24
Anand Singh	Chapter 61
anatoli	Chapter 16

biddulph.r	Chapter 38
Blackbelt	Chapters 2, 40 and 264
BlitzKraig	Chapter 43
Blundell	Chapters 187 and 188
Blundering Philosopher	Chapters 81 and 147
bpoiss	Chapter 41
Braj Bhushan Singh	Chapter 18
Brenden Kromhout	Chapter 47
bricklore	Chapter 69
BrickTop	Chapter 10
Bryan	Chapters 9, 16 and 18
Bryan Bryce	Chapter 39
Buddy	Chapters 40 and 57
Burak Day	Chapters 205 and 264
Burhanuddin Rashid	Chapter 101
busradeniz	Chapter 58
Cabezas	Chapter 112
Caique Oliveira	Chapter 39
Carl Poole	Chapters 65 and 206
Carlos	Chapter 14
Carlos Borau	Chapters 43 and 242
carvaq	Chapter 53
CaseyB	Chapters 96 and 108
Cassio Landim	Chapter 96
cdeange	Chapters 47, 88, 110, 111 and 219
Charu□	Chapters 1, 2, 8, 37, 41, 76, 78, 81, 103, 125, 239 and 264
Chintan Soni	Chapter 229
Chip	Chapters 21, 23 and 91
Chirag SolankI	Chapter 16
Chol	Chapter 61
Chris Stratton	Chapter 264
Christlin Panneer	Chapters 225, 226 and 240
Code.IT	Chapters 34 and 41
Code_Life	Chapter 219
Cold Fire	Chapter 41
commonSenseCode	Chapter 258
CptEric	Chapter 239
cricket_007	Chapters 42 and 47
dakshbhatt21	Chapters 37, 227 and 243
Dale	Chapter 264
Dalija Prasnikar	Chapters 38 and 41
Damian Kozlak	Chapters 40, 42 and 227
Dan	Chapters 14, 96 and 113
Dan Hulme	Chapters 37 and 40
Daniel Käfer	Chapter 41
Daniel Nugent	Chapters 3, 8, 9, 12, 13, 16, 24, 38, 40, 41, 42, 45, 61, 62, 69, 70, 72, 74, 82, 88, 92, 96, 103, 114, 115, 143, 146, 218, 227, 229, 239, 243, 249, 250, 251, 258 and 263
Daniel W.	Chapter 174
DanielDiSu	Chapters 21, 41 and 219
Daniele Segato	Chapter 1
David Argyle Thacker	Chapter 39
David Cheung	Chapter 264

David Medenjak	Chapters 17 and 112
davidgiga1993	Chapter 27
DeKaNszn	Chapter 220
dev.mi	Chapter 37
devnull69	Chapters 70 and 219
Dhaval Solanki	Chapter 96
Dima Rostopira	Chapter 250
Dinesh Choudhary	Chapters 73 and 111
Disk Crasher	Chapter 86
Dmide	Chapter 25
Don Chakkappan	Chapter 77
Doron Behar	Chapter 1
Doron Yakovlev	Chapters 55 and 185
Douglas Drumond	Chapter 7
drulabs	Chapter 231
Duan Bressan	Chapters 74 and 219
Dus	Chapters 69 and 238
Ege Kuzubasioglu	Chapter 241
Eixx	Chapter 81
Ekin	Chapter 125
EKN	Chapters 41 and 131
EmmanuelMess	Chapter 135
Endzeit	Chapter 98
Enrique de Miguel	Chapter 210
EpicPandaForce	Chapters 112, 113 and 127
Er. Kaushik Kajavadara	Chapters 14 and 195
Erik Minarini	Chapters 1, 41, 42 and 57
Eugen Martynov	Chapters 184 and 263
Fabian Tamp	Chapter 250
Fabio	Chapters 47, 139, 148, 194, 205, 264 and 265
Farid	Chapter 204
Felix Edelmann	Chapter 19
Flayn	Chapter 59
Floern	Chapters 8, 34, 38, 41, 47 and 71
Florent Spahiu	Chapters 9, 47 and 218
FlyingPumba	Chapter 158
Franck Dernoncourt	Chapter 41
FredMaggiowski	Chapters 137 and 250
Freek Nortier	Chapter 250
FromTheSeventhSky	Chapter 18
Froyo	Chapter 142
fuwaneko	Chapter 18
fyfyone Google	Chapters 44, 156, 205 and 264
g4s8	Chapters 24, 28, 34, 41, 42, 72, 130, 187 and 264
gaara87	Chapters 4, 39, 163, 172 and 176
Gabe Sechan	Chapter 262
Gabriele Mariotti	Chapters 1, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 19, 26, 37, 40, 47, 50, 61, 66, 76, 78, 83, 88, 92, 103, 117, 118, 132, 144, 153, 154, 218, 219, 227, 229, 230, 231, 233, 244, 250, 251, 252, 255, 258 and 264
Gaket	Chapter 1
Gal Yedidovich	Chapter 37
gattsbr	Chapter 263
Gaurav Jindal	Chapter 8

gbansal	Chapters 40, 69 and 126
Geert	Chapter 100
GensaGames	Chapters 16 and 255
gerard	Chapter 218
Ghanshyam Sharma	Chapter 199
Gian Patrick Quintana	Chapter 209
Giannis	Chapter 235
Ginandi	Chapter 219
Gokhan Arik	Chapter 180
Gorg	Chapter 227
Graham Smith	Chapter 219
grebulon	Chapter 264
Greg T	Chapters 40, 67, 70, 96, 149, 160 and 262
Guilherme Torres Castro	Chapter 31
Guillaume Imbert	Chapter 39
Guillermo García	Chapter 37
GurpreetSK95	Chapter 176
gus27	Chapters 109 and 115
H. Pauwelyn	Chapter 114
h22	Chapter 48
Hamed Gh	Chapter 166
Hamed Momeni	Chapter 108
hankide	Chapters 1 and 47
Hannoun Yassir	Chapters 205 and 264
Harish Gyanani	Chapters 1, 5, 28, 38, 41, 99, 134, 201, 219 and 265
Harsh Pandey	Chapter 76
Harsh Sharma	Chapter 66
Hasif Seyd	Chapter 71
HDehghani	Chapter 34
hello_world	Chapter 84
herrmartell	Chapter 98
Hi I'm Frogatto	Chapters 41, 42, 113, 137 and 250
Hiren Patel	Chapters 2, 4, 25, 34, 42, 53, 81, 82, 117, 118, 128, 141, 150, 152, 160 and 239
Hitesh Sahu	Chapter 155
honk	Chapters 4, 11, 18, 33, 37, 38, 44, 57, 58, 63, 74, 75, 86, 87, 91, 98, 100, 112, 116, 118, 119, 122, 133, 134, 139, 145, 149, 152, 155, 163, 165, 179, 194, 195, 200, 224, 228, 231, 233, 235, 238, 240, 242, 248, 252, 257, 258, 259, 264 and 265
Hussein El Feky	Chapter 102
Ic2h	Chapter 218
Ichigo Kurosaki	Chapters 37, 62, 126 and 250
Ichthyocentaurs	Chapters 1, 47, 72, 92 and 179
iDevRoids	Chapter 261
Ilya Blokh	Chapter 227
Ilya Krol	Chapter 61
Iman Hamidi	Chapter 91
Imdad	Chapter 77
IncrediApp	Chapters 40 and 218
inetphantom	Chapter 1
insomniac	Chapter 41
Inzimam Tariq IT	Chapter 2
iravul	Chapters 22 and 152
Irfan Raza	Chapter 41

Kiran Benny Joseph	Chapters 1 and 145
Kirill Kulakov	Chapter 72
kit	Chapter 118
Kling Klang	Chapter 38
Knossos	Chapter 151
krishan	Chapter 215
Krishnakanth	Chapter 207
kRiZ	Chapter 168
krunal patel	Chapters 64 and 229
KuroObi	Chapter 86
L. Swifter	Chapter 219
Laurel	Chapter 156
Lazy Ninja	Chapters 72 and 227
Leo	Chapter 229
Leo.Han	Chapter 72
Lewis McGeary	Chapters 8, 61 and 163
Lokesh Desai	Chapters 186 and 260
Long Ranger	Chapter 39
LordSidious	Chapters 40, 78 and 178
Lucas Paolillo	Chapters 45, 61 and 227
Lukas	Chapters 78, 120 and 176
M D P	Chapters 62 and 262
M M	Chapter 244
Madhukar Hebbar	Chapter 125
Magesh Pandian	Chapter 134
Mahmoud Ibrahim	Chapters 82 and 234
Malek Hijazi	Chapter 144
Manos	Chapter 175
Marco Marchiori	Chapters 40 and 84
Marcus Becker	Chapter 130
MarGenDo	Chapter 182
Mario Lenci	Chapter 263
Mark Ormesher	Chapter 41
Mark Yisri	Chapter 1
marshmallow	Chapter 76
MashukKhan	Chapter 235
Matas Vaitkevicius	Chapter 1
MathaN	Chapters 1, 16, 37, 41 and 250
mattfred	Chapter 112
Mauker	Chapters 41, 218 and 219
Max	Chapters 37, 41, 61, 97 and 131
Max McKinney	Chapter 40
mayojava	Chapter 46
Medusalix	Chapters 72 and 136
Menasheh	Chapters 38, 46 and 264
mhenryk	Chapter 61
Michael Allan	Chapter 1
Michael Spitsin	Chapters 43, 45 and 218
Michael Vescovo	Chapter 251
Michele	Chapter 4
MidasLefko	Chapters 19 and 82
MiguelHincapieC	Chapters 65 and 153
Mikael Ohlson	Chapter 258

Omar Aflak	Chapter 229
Omar Al Halabi	Chapter 96
once2go	Chapter 92
Onik	Chapter 59
Onur	Chapter 31
orelzion	Chapter 73
Oren	Chapter 258
originx	Chapter 252
oshurmamadov	Chapters 37 and 92
Pablo Baxter	Chapters 16, 35, 82, 98 and 239
Pankaj Kumar	Chapter 26
Paresh Mayani	Chapter 97
Parsania Hardik	Chapter 42
Patrick Dattilio	Chapters 8, 76 and 253
Paul Lammertsma	Chapter 41
Pavel Durov	Chapters 56, 205 and 264
Pavel Strelchenko	Chapter 47
Pavneet_Singh	Chapters 1, 41 and 47
Pawel Cala	Chapter 40
Pedro Varela	Chapter 197
Peter Taylor	Chapter 120
Phan Van Linh	Chapters 2 and 8
Phil	Chapter 72
PhilLab	Chapters 90 and 262
Pinaki Acharya	Chapter 96
piotrek1543	Chapter 69
Piyush	Chapters 76 and 83
Pongpat	Chapters 88 and 132
Pradumn Kumar Mahanta	Chapter 200
Prakash Bala	Chapter 24
pRaNaY	Chapters 92, 125 and 263
Pratik Butani	Chapters 9, 21, 26, 147, 202 and 245
privatestaticint	Chapter 157
PRIYA PARASHAR	Chapter 249
Priyank Patel	Chapters 6, 26 and 162
Pro Mode	Chapters 1 and 102
Prownage	Chapter 219
PSN	Chapter 1
R. Zagórski	Chapters 2, 38, 47, 56, 65, 73, 83, 84, 88, 131, 176, 205, 255 and 264
rajan ks	Chapter 40
Rajesh	Chapters 8, 37, 41, 62, 72 and 93
Rakshit Nawani	Chapter 211
Raman	Chapter 250
Ramzy Hassan	Chapter 61
Ranveer	Chapter 247
Rasoul Miri	Chapter 76
Ravi Rupareliya	Chapters 39 and 62
rciovati	Chapters 41 and 47
Reaz Murshed	Chapters 16, 41, 47 and 56
RediOne1	Chapters 13, 41, 53, 62 and 80
Redman	Chapter 84
reflective_mind	Chapter 171
rekire	Chapters 8, 15, 41, 44, 47, 218, 255 and 263

Revanth Gopi	Chapter 47
reVerse	Chapter 243
ReverseCold	Chapter 241
Ricardo Vieira	Chapters 43, 131 and 135
ridsatrio	Chapters 37 and 97
Risch	Chapter 258
RishbhSharma	Chapter 264
Robert	Chapter 122
Robert Banyai	Chapter 192
Roberto Betancourt	Chapter 213
Robin Dijkhof	Chapter 227
Rohan Arora	Chapter 99
Rohit Arya	Chapters 40, 61, 78 and 123
Rolf ツ	Chapter 253
Romu Dizzy	Chapter 181
Rosário Pereira Fernandes	Chapters 76 and 139
Rubin Nellikunnathu	Chapters 44, 98 and 154
Rupali	Chapters 69, 139 and 165
russjr08	Chapter 41
russt	Chapters 1 and 263
S.D.	Chapter 90
S.R	Chapters 14, 68 and 161
Saeed	Chapters 7 and 229
Sagar Chavada	Chapters 16 and 37
Sam Judd	Chapter 61
sameera lakshitha	Chapter 98
samgak	Chapter 185
Sammy T	Chapter 78
SANAT	Chapter 15
Sanket Berde	Chapters 16, 92 and 229
Sanoop	Chapters 37 and 117
Sasank Sunkavalli	Chapter 16
Sashabrava	Chapter 22
saul	Chapter 1
saurav	Chapter 78
Saveen	Chapter 167
Segun Famisa	Chapter 39
Sevle	Chapter 15
shadygoneinsane	Chapter 222
shahharshil46	Chapter 229
ShahiM	Chapter 162
shalini	Chapter 232
Shantanu Paul	Chapter 74
Shashanth	Chapter 250
Shekhar	Chapter 189
shikhar bansal	Chapter 230
Shinil M S	Chapters 40, 92 and 164
Shirane85	Chapter 88
ShivBuyya	Chapters 40 and 62
shtolik	Chapters 24 and 117
Shubham Shukla	Chapter 230
Siddharth Venu	Chapter 1
Simon	Chapters 41 and 116

Simon Schubert	Chapter 118
Simone Carletti	Chapters 74 and 262
Simplans	Chapters 1 and 41
SimplyProgrammer	Chapter 37
Sir SC	Chapters 8 and 97
Smit.Satodia	Chapter 18
Sneh Pandya	Chapters 1, 9, 14, 16, 37, 61, 96 and 257
Sohail Zahid	Chapter 4
SoroushA	Chapters 41, 129 and 147
spaceplane	Chapter 41
ssimm	Chapter 173
Stanojkovic	Chapter 118
Stephane Mathis	Chapter 41
Steve.P	Chapter 33
still_learning	Chapter 59
stkent	Chapters 29 and 264
sud007	Chapters 14, 15, 25, 26, 37 and 153
Sujith Niraikulathan	Chapters 2, 4, 28, 118, 140 and 165
Sukrit Kumar	Chapters 177 and 242
sukumar	Chapters 61, 183, 205, 239 and 264
Sup	Chapter 21
Suragch	Chapter 28
Suresh Kumar	Chapters 67, 183 and 239
Sweeper	Chapter 71
Täg	Chapter 59
tainy	Chapter 41
Talha Mir	Chapter 8
TameHog	Chapter 126
Tanis.7x	Chapters 38, 39, 124 and 247
TDG	Chapters 116 and 263
Tejas Pawar	Chapter 250
theFunkyEngineer	Chapters 41 and 264
THelper	Chapter 262
thetonrifles	Chapter 16
thiagolr	Chapters 59, 263 and 264
Thomas Easo	Chapter 64
ThomasThiebaud	Chapters 2, 8, 38, 41, 44, 47, 218 and 262
Tien	Chapter 264
Tim Kranen	Chapters 8 and 219
Timo Bähr	Chapter 254
Tomik	Chapter 151
Tot Zam	Chapter 16
tpk	Chapter 98
TR4Android	Chapters 28, 69 and 70
TRINADH KOYA	Chapter 146
Tudor Luca	Chapter 30
tynn	Chapters 106, 109, 114, 211 and 214
ubuntudroid	Chapter 8
Ufkoku	Chapter 92
Uriel Carrillo	Chapters 81 and 227
user1506104	Chapter 250
USKMobility	Chapter 265
Uttam Panchasara	Chapters 50 and 87

You may also like