1. Introduction to Android

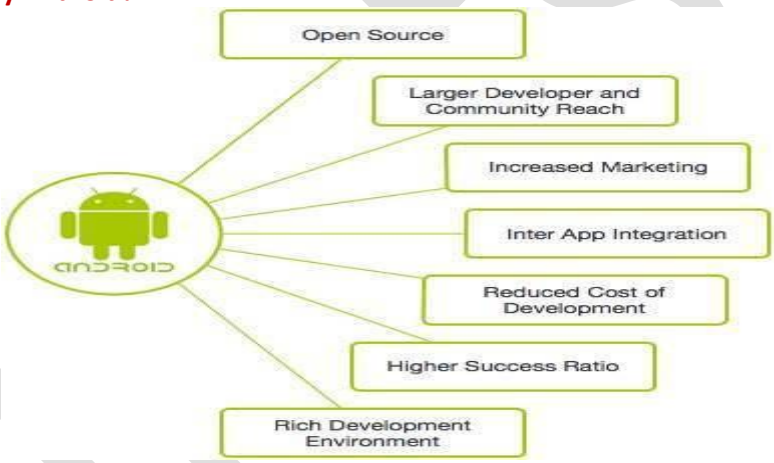
1.1. Introduction to Mobile Application Development

Mobile OS are keep on introducing from Palm OS in 1996 to Windows pocket PC in 2000 then to Blackberry OS and Android. One of the most popular used mobile OS now days are ANDROID. There are around 2.0 lack+ games, application and widgets available on the market for Android users. Android is a powerful Operating System supporting a large number of applications in Smart Phones. These applications make life more comfortable and advanced for the users. Android is a Linux-based operating system designed for mobile devices such as smartphones and tablet computers. The Android is a light weight OS and can easily embedded into different hardware devices like networking equipment, smart TV systems including set top boxes and built in systems and various devices like as house hold appliances and wrist watches.

1.2. What is Android

- It is a platform that provides tools and technologies which can used to develop and build mobile Applications. The Android platform is open and free software stack that includes an operating system, middleware services and also key applications for use on mobile devices.The Android environment uses the Linux operating system at its core.
- Middleware Services makes it easier for software developers to perform communication and input/output, so that developers can concentrate on the specific purpose of their application without need of concentrating on input output resource coding of device. Android provides a middleware layer including libraries that provide services such as data storage, screen display, multimedia, and web browsing. Because the middleware libraries are compiled to machine language, services execute quickly.
- Android also provides an application framework that developers integrate into their applications for application development.

1.3. Why Android?



1.4. Features

1) Open source framework, Android OS is open source and it is part of the Open Handset Alliance, most of the leading handset manufacturers in the world have Android phones.
2) Uses of tools are very simple.
3) Availability of Apps, majority of the apps in Google Play are free as compared to the paid apps on iPhone.
4) Inbuilt support for the Flash.
5) Social networking integration like with Twitter, Facebook, ..etc
6)Integrated Applications & Features, Eg: Android allows an option to share, after taking a photo with the Camera.
7) Free to customizes, we can customize widgets as our wish with new properties.
8) Better Notification System (comprises emails, updates from various widgets.)
9) Updated user interface design
10) It has a better App Market, to easily upload and download.
11) Different Resolutions for Different Screen Sizes.

12) System wide copy and paste functionalities.
13) Multi touch interfacing
14) Java Support so can develop Robust applications.
15) Multilanguage support
16) Multitask Support, i.e Android phone can run different applications simultaneously, which obviously makes it easier to multitask and improves the overall functionality of the phone.

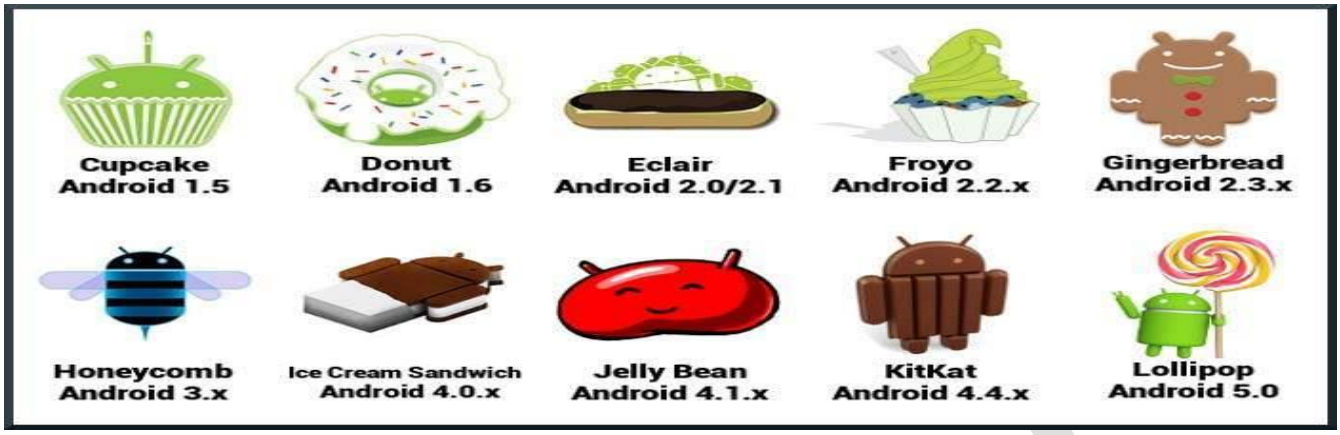
What is Open Source Project ?

Key features of Android

1. Storage: SQLite, a lightweight relational database, is used for data storage purposes.
2. Connectivity: Android supports connectivity technologies including GSM/CDMA/ Blue tooth, Wi-Fi, 3G, 4G, ..etc
3. Messaging : SMS and MMS are available forms of messaging
4. Multiple language support : Android supports multiple languages
5. Web browser : The web browser available in Android is based on the open-source WebKit layout engine
6. Java support : Java classes are compiled into Dalvik executables and run on Dalvik, a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU
7. Media support : Android supports the following audio/video/still media formats, like WAV, JPEG, PNG, GIF, BMP, 3GP or MP4 container
8. Streaming media support : HTTP Live Streaming is supported by RealPlayer and others for Android
9. Additional hardware support : Android can use video/still cameras, touchscreens, GPS, accelerometers, gyroscopes, barometers, magnetometers, gaming Controllers, sensors.
10. Multi-touch : Android has native support for multi-touch
11. Bluetooth : Supports sending files, accessing the phone book, voice dialing and sending contacts between phones.
12. Video calling : Skype 2.1 offers video calling in Android 2.3, including front camera support.
13. Multitasking : Multitasking of applications
14. Tethering : Android supports tethering, which allows a phone to be used as a wireless/wired Wi-Fi
15. External storage : Most Android devices include MicroSD slot and can read MicroSD cards formatted with FAT32, Ext3 or Ext4 file system.

1.5. History & Versions

The code names of android ranges from A to M currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat and Lollipop. Let's understand the android history in a sequence.



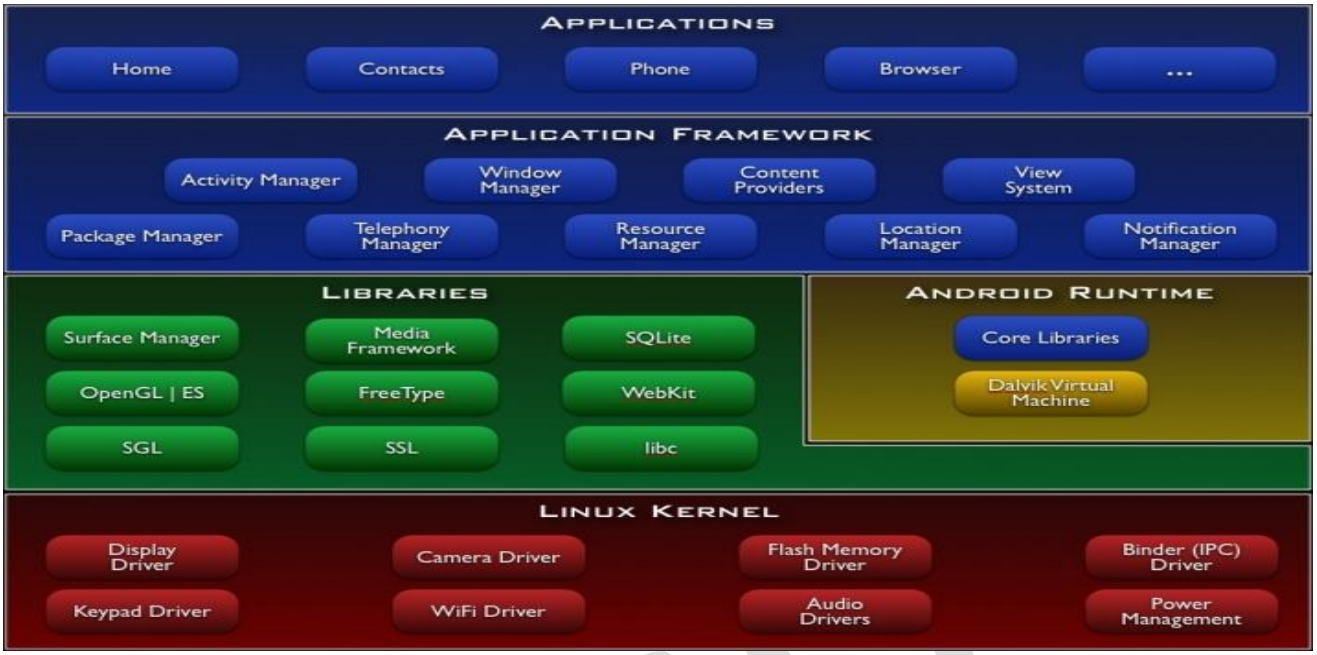
What is API level?

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

Platform Version	API Level	VERSION_CODE
Android 6.0	23	Marshmallow
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP
Android 4.4W	20	KITKAT_WATCH
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1
Android 4.1, 4.1.1	16	JELLY_BEAN
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 & Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2, Android 2.3.1 & Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ÉCLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

2. Android Architecture

2.1. Architecture



The Android Architecture Elements:

2.1 Applications :

- ✓ Predefined standard applications are pre-installed with every device, such as:
  - SMS app
  - Dialer
  - Web browser
  - Contact manager
  - Email App
- ✓ As a developer we are able to write an app which replaces any existing system app.

2.2. Android Frameworks

- ✓ Applications are the top layer in the Android architecture.

This is used in our application development to run

Block	Explanation
Activity Manager	Manages the activity life cycle of applications. To understand the Activity component
Window Manager	Manage different activities in Android
Content Provider	Manage the data sharing between applications. Our Post on Content Provider component describes this in greater detail
View System	An extensible set of views used to create application user interfaces
Package Manager	Manages the different packages in Application
Telephony Manage	Manages all voice calls. We use telephony manager if we want to access voice calls in our application.
Resource Manager	Manage the various types of resources we use in our Application
Location Manager	Location management, using GPS or cell tower
Notification Manager	Allows applications to display alerts and notifications to the user.

2.3. Linux Kernel

- ✓ It is the basic layer of Android. The complete Android OS is built on top of the Linux.
- ✓ It interacts with the hardware and contains all the essential hardware drivers.
- ✓ Drivers are programs that control and communicate with the hardware. For example, consider the Bluetooth function, all devices has a Bluetooth hardware in it. So that kernel must include Bluetooth driver to communicate with the Bluetooth hardware.
- ✓ The Linux kernel acts as an abstraction layer between the hardware and other software layers. Android uses the Linux for all its core functionality such as Memory management, process management, networking, security settings etc.

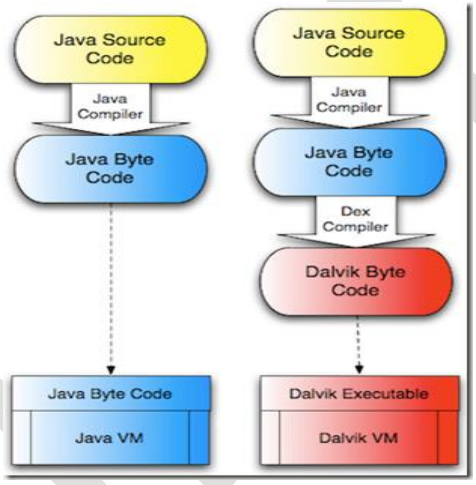
2.4. Core Libraries

- ✓ The next layer is the Android's native libraries, this is the layer that enables the device to handle different types of data. These libraries are written in C or C++ language and are specific for a particular hardware.
- ✓ Some of the Libraries are ,

  Media framework: Media framework provides different media codec's allowing the recording and playback of different media formats and static image files.
  SQLite: It is light weight relational database engine used in android for data storage purposes.
  WebKit: It is the browser engine used to display HTML content.
  OpenGL: Used to deliver 2D or 3D graphics content to the screen.
  SSL : (Secure Socket Layer) Used for Internet Security.
  FreeType : Bit map and font interpretation
  SurfaceManager : Manages access to the display subsystem and provides support for 2D and 3G graphic layers from multiple applications.

2.5. Android Runtime

Every android application runs in its own process, with its own instance of the Dalvik Virtual Machine.



2.6. Dalvik virtual Machine

- ✓ The DVM operates on bytecodes that are transformed from the Java Class files compiled by a Java compiler into another class file format called the .dex format using a dx tool which is included in the Android SDK (Software Development Kit).
- ✓ A virtual machine optimized for mobile devices that was designed and written by Dan Bornstein and other Google engineers.
- ✓ The intention of the DVM is creation of platform neutral dex files, this process seems like as Java Virtual Machine (JVM), which processes the Java bytecodes while using Java Class files or Jar files.
- ✓ The processes of the DVM is optimized instructions for the low memory requirements needed for development and implementation of applications for the mobile phone platform.
- ✓ The DVM is register-based, and it can run classes compiled by a Java language compiler that have been transformed into its native format.

3. Setup of Android Development Environment

What is Android SDK ?

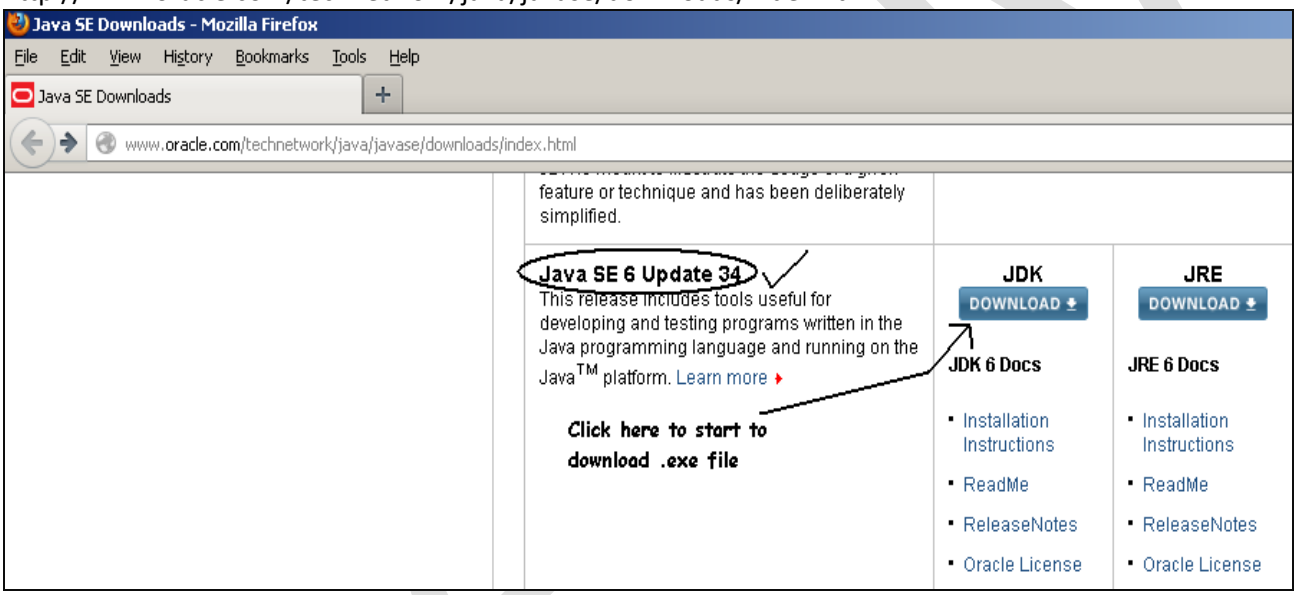
- ✓ The Android SDK includes a complete set of development tools. It includes a debugger, libraries, a handset emulator.
- ✓ Software written in Java can be compiled to be executed in the Dalvik virtual machine, which is a specialized VM implementation designed for mobile device use.

Required Software to Install Android ?

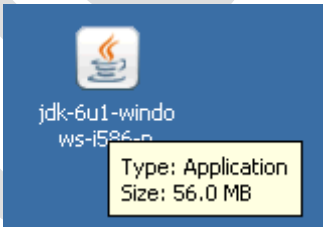
1. JDK6 Version
2. Android SDK for Windows (if you have Windows OS)
3. Eclipse IDE

Installing JDK6 (Java Development Kit)

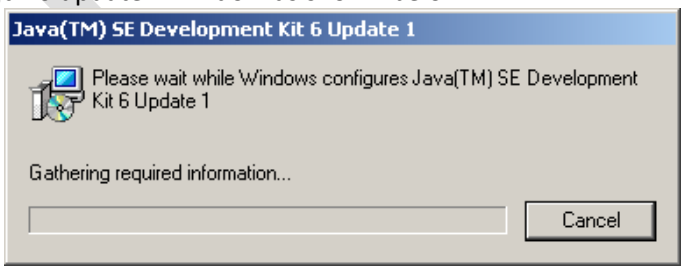
The latest version of jdk is 6, at the time of writing my java tutorial. We can download the jdk from: http://www.oracle.com/technetwork/java/javase/downloads/index.html.

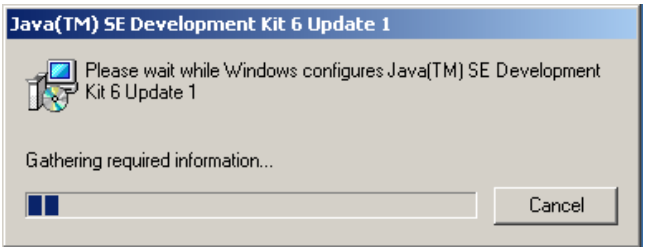


Step 1. Once we download the exe file we can now install it. Just follow what i mentioned below: To install the jdk, double click on the downloaded exe file (jdk-6u1-windows-i586-p.exe) Double Click the icon of downloaded exe from the downloaded location.

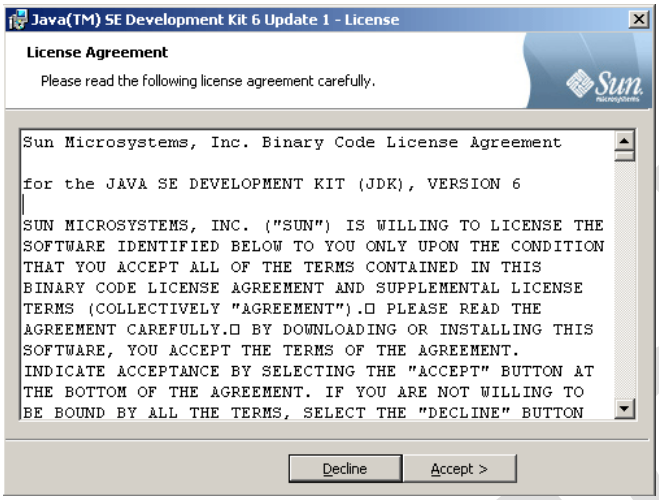


You will see jdk 6 update 1 window as shown below.

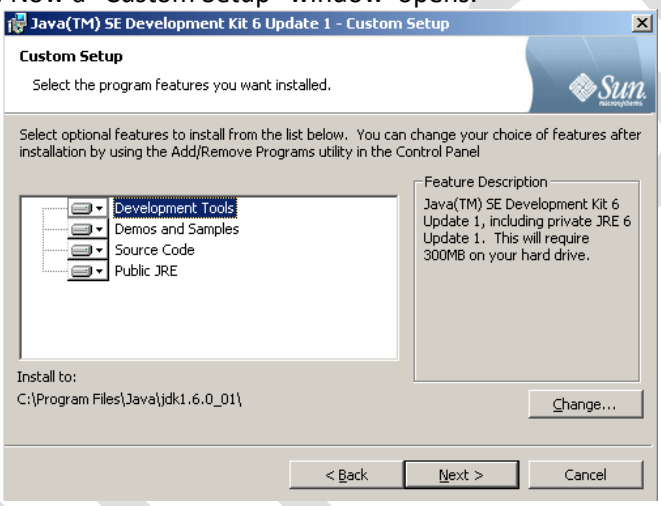




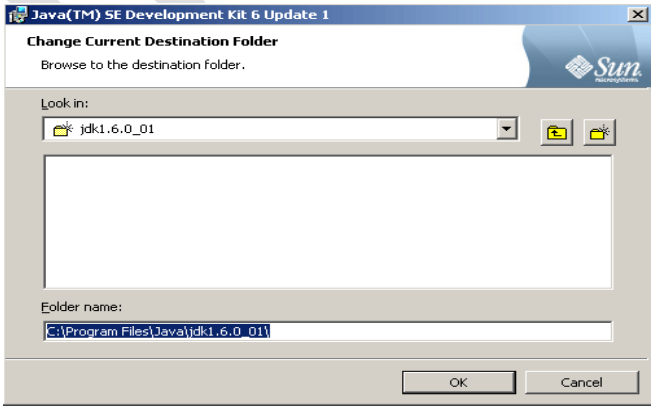
Step 2: Now a "License Agreement" window opens. Just read the agreement and click "Accept" button to accept and go further.



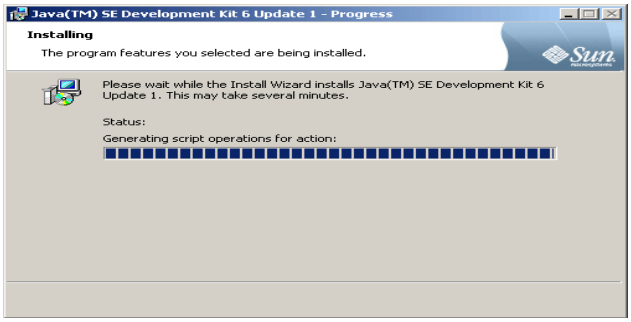
Step 3: Now a "Custom Setup" window opens.



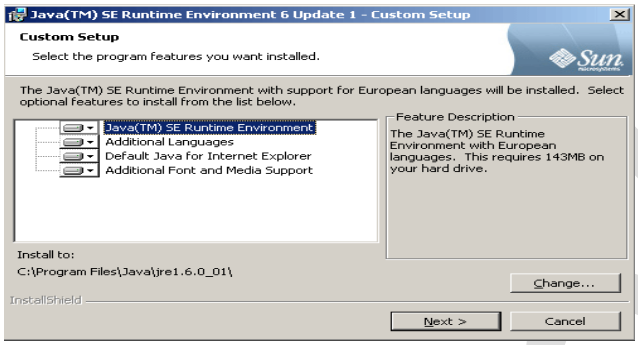
Step 4: Click on "Change" button to choose the installation directory. Here it is "C:\Program Files\ Java\jdk1.6.0_01". Now click on "OK" button.



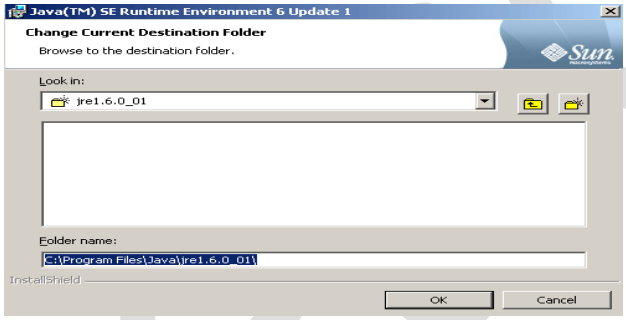
Clicking the "OK" button starts the installation. It is shown in the following figure.



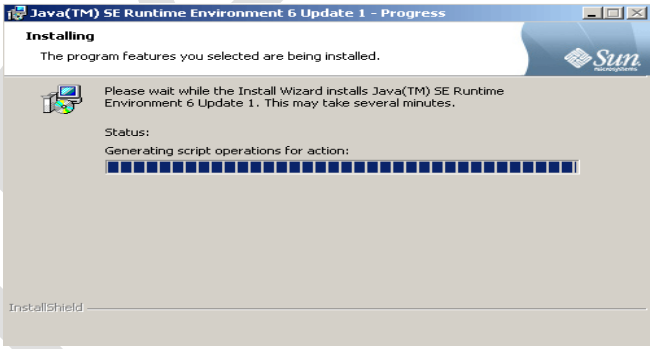
Step 5: Next window asks to install Runtime Environment.



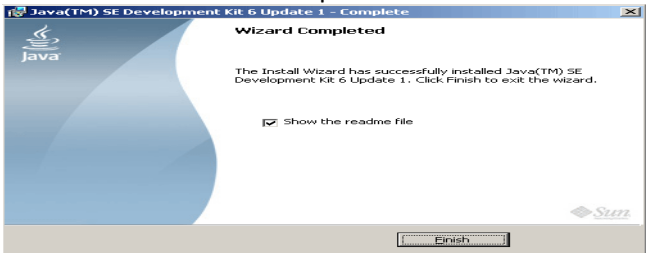
Click the "Change" button to choose the installation directory of Runtime Environment. We prefer not to change it. So click "OK" button.



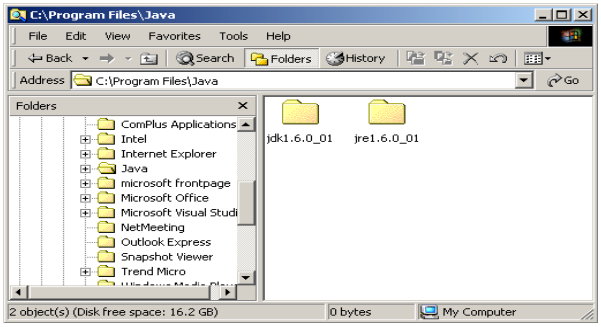
Step 6: Click "OK" button starts the installation.



Step 7: Now "Complete" window appears indicating that installation of jdk 1.6 has completed successfully. Click "Finish" button to exit from the installation process.

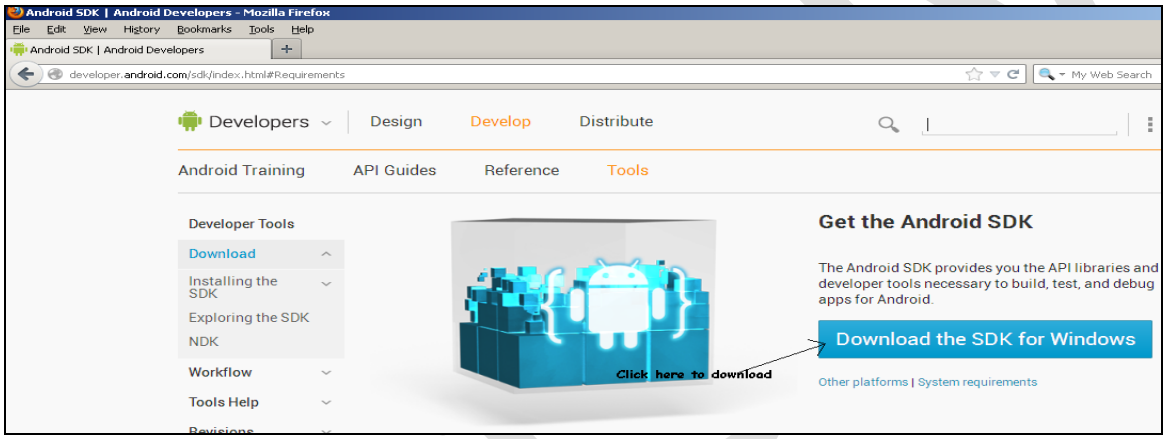


Step 8: The above installation will create two folders "jdk1.6.0_01" and "jre1.6.0_01" in "C:\ Program Files\ java" folder.

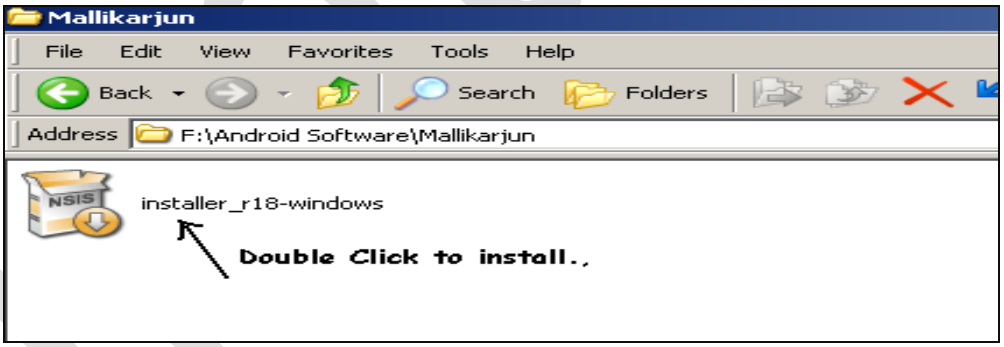


Installing Android SDK Tools & API ?

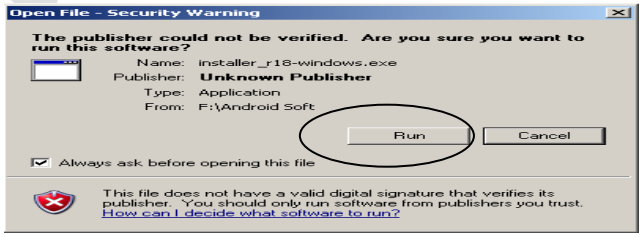
Step-1) Download Android SDK from http://developer.android.com/sdk/index.html, as I shown below.



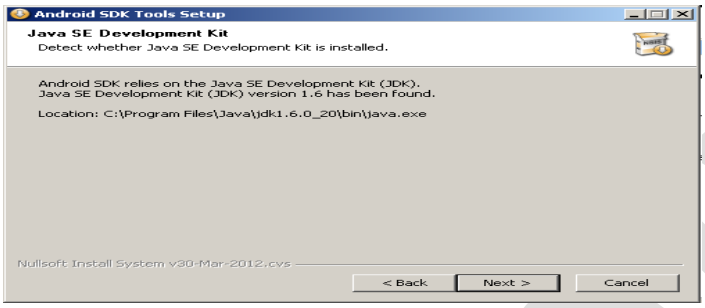
We will get the downloaded exe file , as



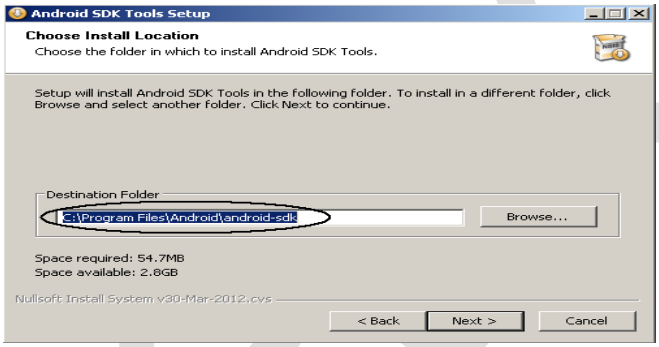
Step-2) Double click on SDK Installer.exe file to install



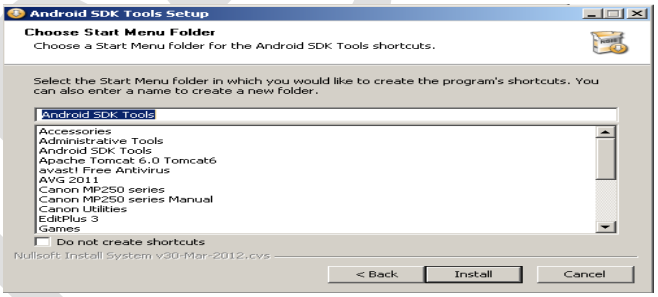
Click on Next button



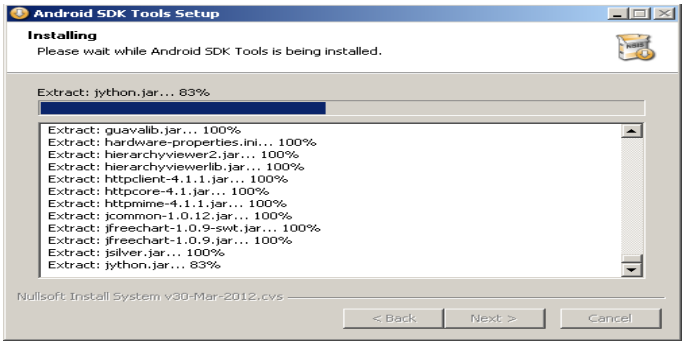
Click on Next button

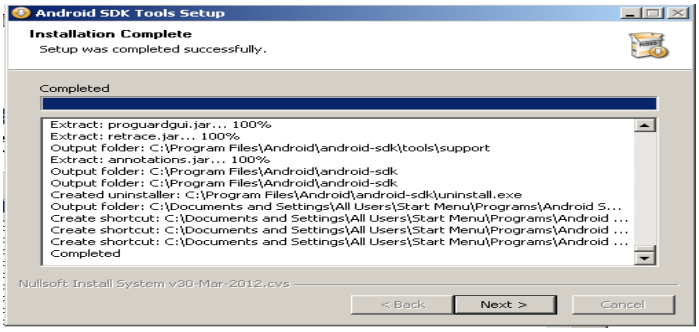


Step-3) Ensure Destination Folder, and click on Next button

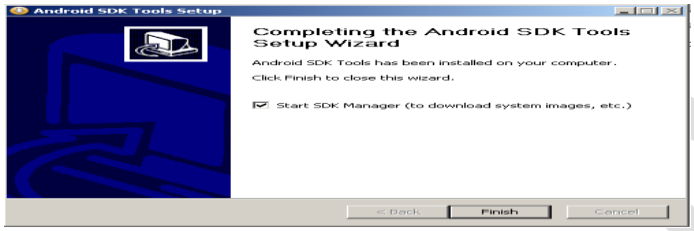


Click on Install Button





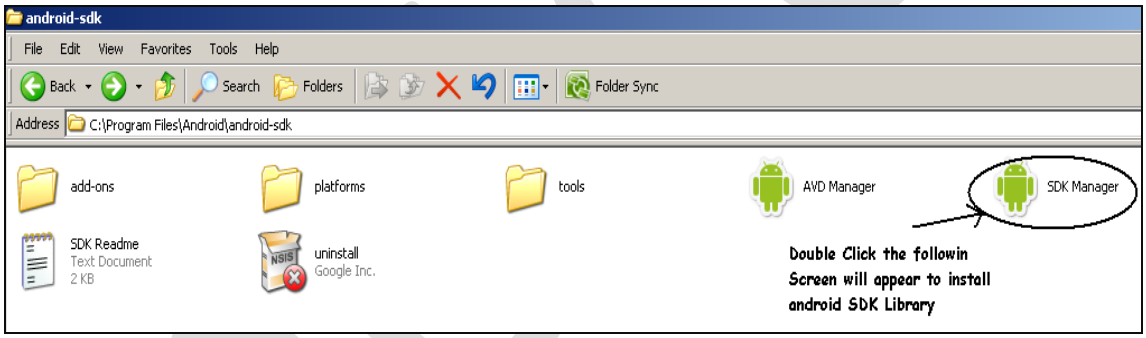
Click on Next button



Click on Finish button

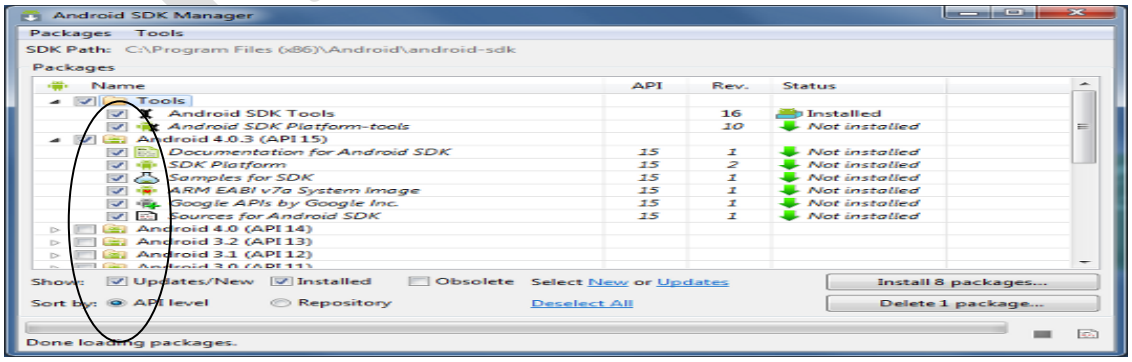
Step-4) Installing Android SDK tools and Android API Levels.

Ensure the following installed Android SDK location in file system, and click on SDK manager from the android-sdk directory.

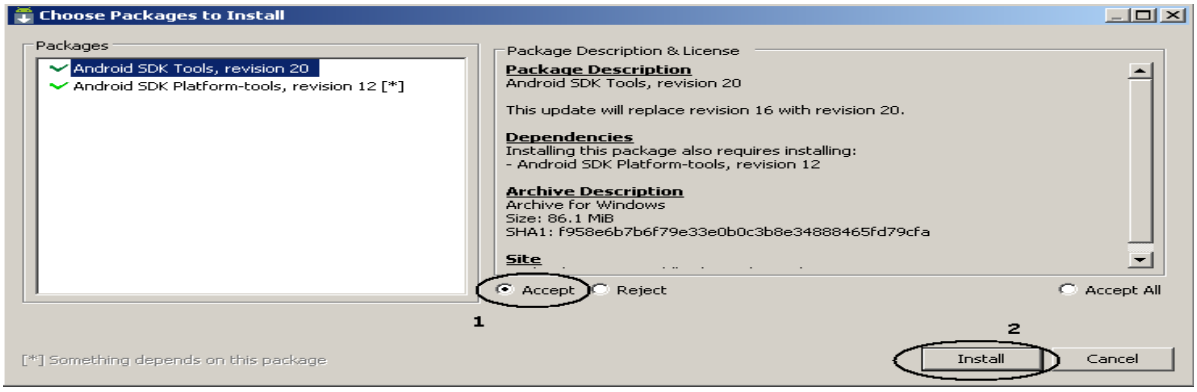


It opens the following window,

Step-5) This step is very important , from the following Window, select tools, API Levels like API15, API 10, ..etc , and also select Extras-> Google USB Driver package for USB Connections with real device.



Click on Install XX Packages button.

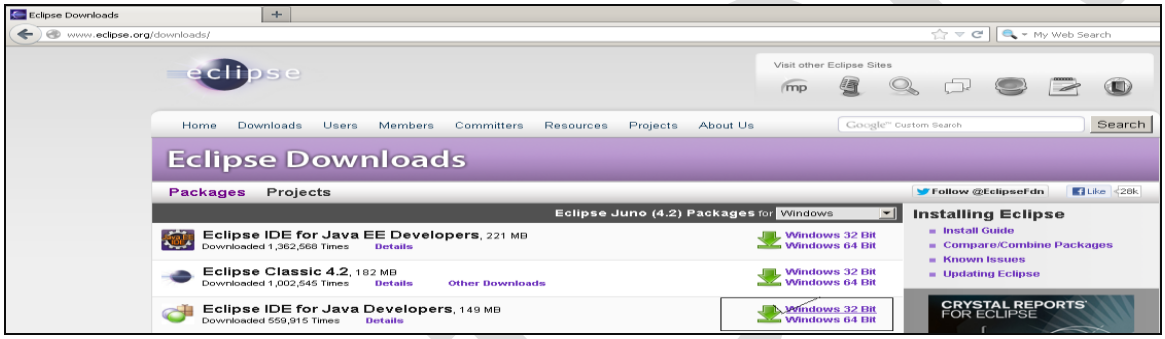


Be patient Installation will start will take more time..

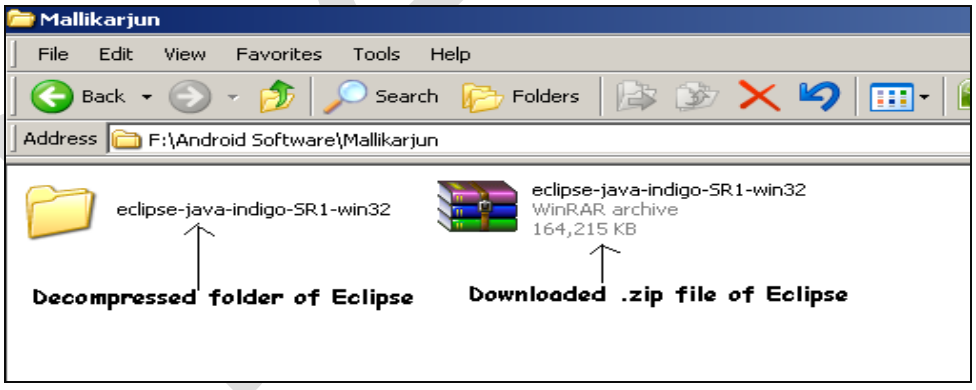
Installing Eclipse & Configuring Android Plugin for Eclipse :

Configuring Android Tools for Eclipse:

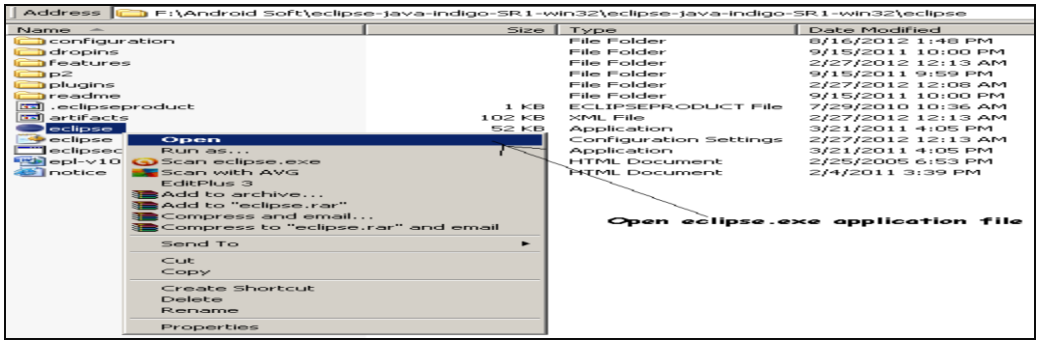
Step-1) Download eclipse software from www.eclipse.org\downloads location.



Step-2) Confirm the downloaded .zip file location, and decompress using either winzip (or) winrar software.

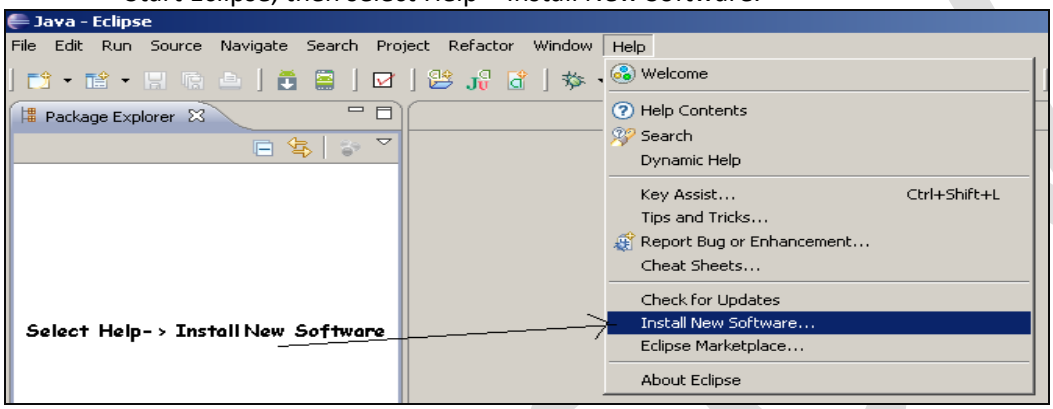


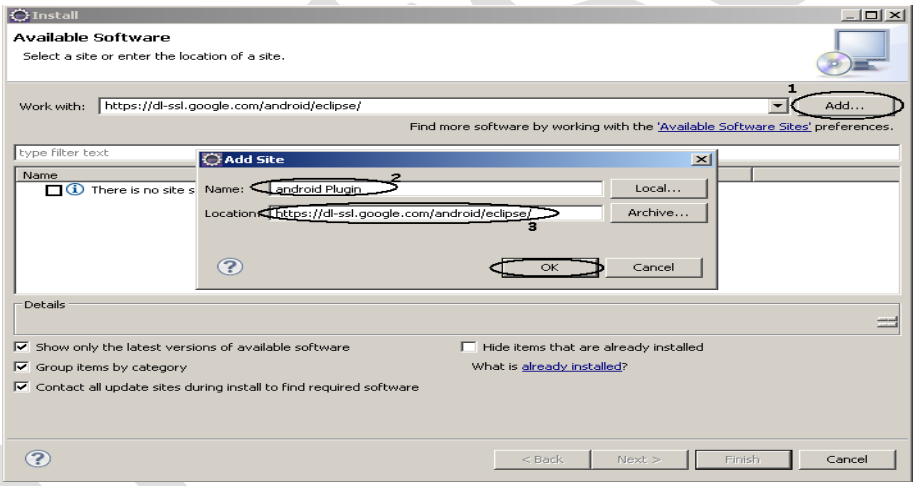
Step-3) Open Eclipse.exe file, from the file system location.



Step-4) *Configruing the ADT Plugin :*

- Start Eclipse, then select Help > Install New Software.



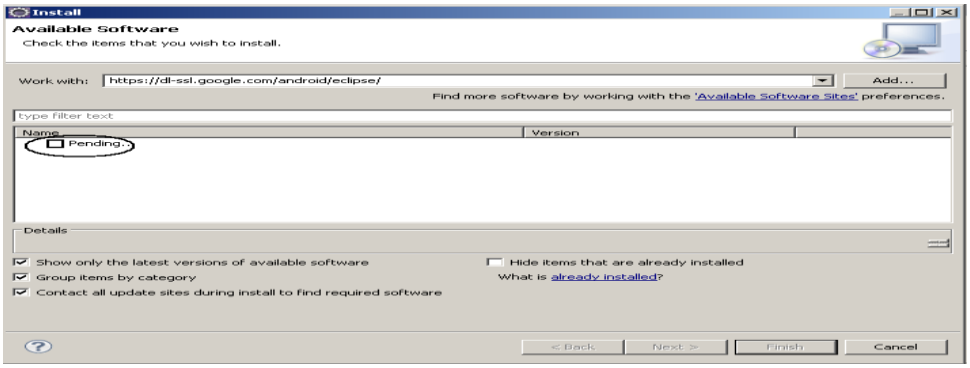


- Click Add, in the top-right corner.
- In the Add Repository dialog that appears, enter "ADT Plugin" for the *Name* and the following URL for the *Location*:

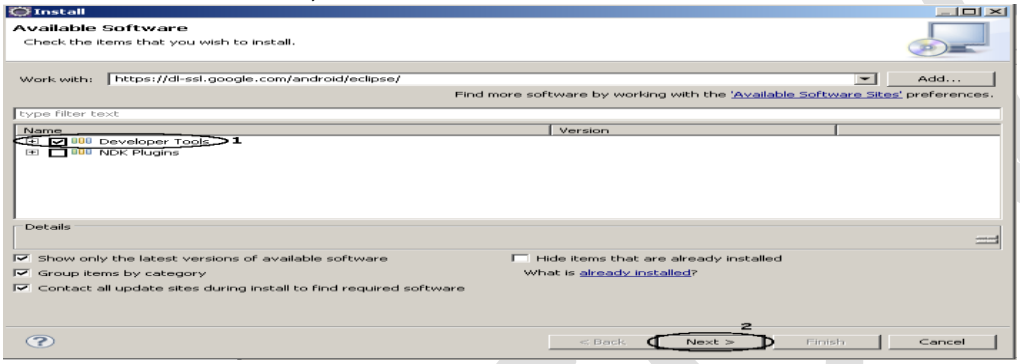
https://dl-ssl.google.com/android/eclipse/

Click OK.

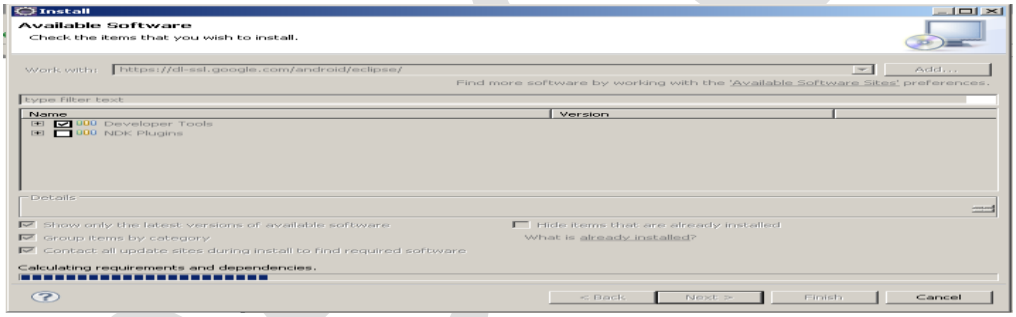
If you have trouble for the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).



- In the Available Software dialog, select the checkbox next to Developer Tools and click Next.
- In the next window, we'll see a list of the tools to be downloaded. Click Next.



- Click on “Next” button

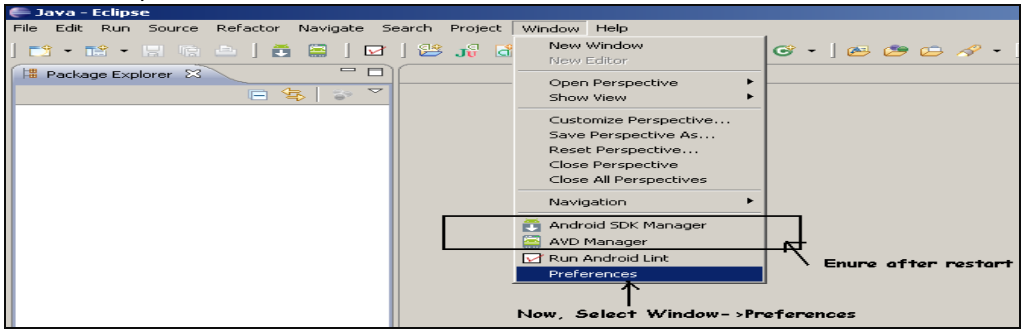


- Read and accept the license agreements, then click Finish.
- If you get a security warning saying that the authenticity or validity of the software can't be established, click OK.
- When the installation completes, restart Eclipse.

Configuring the ADT Plugin for Eclipse :

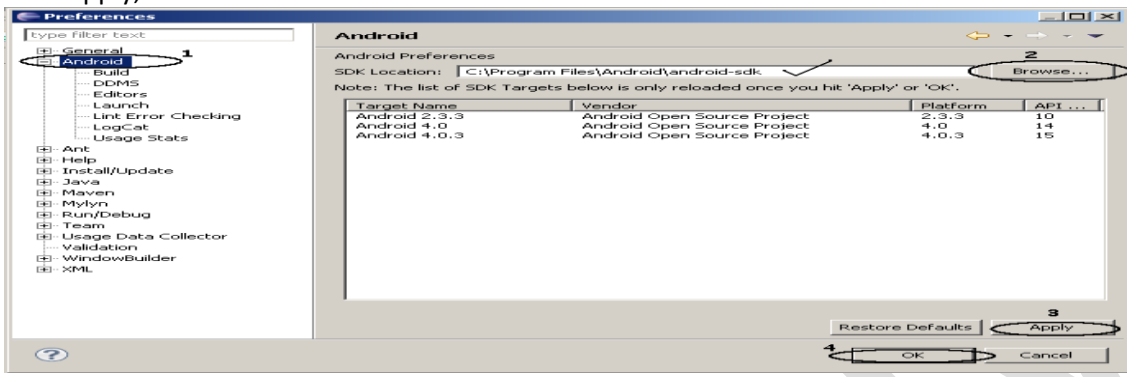
After we've installed ADT and restarted Eclipse, we must specify the location of our Android SDK directory:

1. **Select Window** > Preferences... to open the Preferences panel (on Windows OS X, select Eclipse > Preferences).



2. **Select Android from the left panel.**
   You may see a dialog asking whether you want to send usage statistics to Google. If so, make your choice and click Proceed.

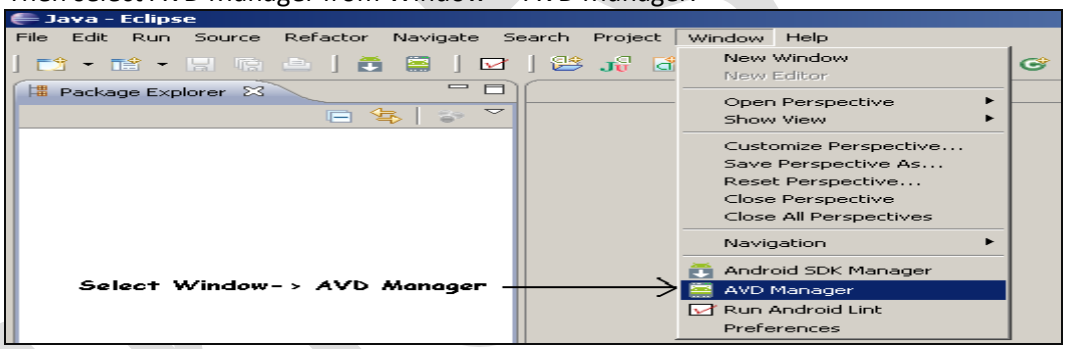
3. For the *SDK Location* in the main panel, click Browse... and locate your downloaded Android SDK directory (such as android-sdk-windows).
4. Click Apply, then OK.

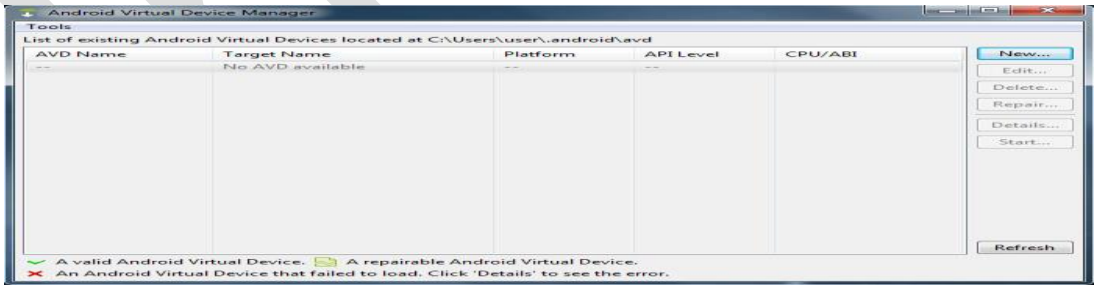


Note : "With this your Android Installation part is completed, we can configure Android plugin for other IDE's like Netbeans, IntelliJ IDE also. "

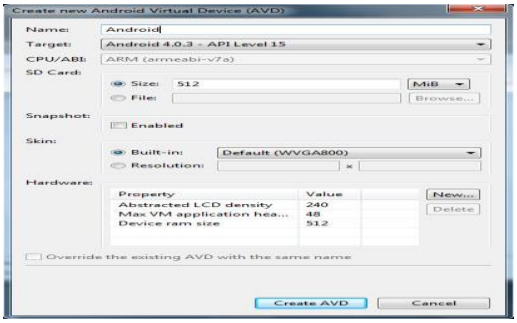
What is AVD ? How to Configure AVD?

- ✓ AVD, stands Android Virtual Device, AVD's are used to start emulator. When we launch the emulator, we should specify the AVD configuration that we want to load..
- ✓ Before Configuring AVD, first confirm SDK tools and API levels are installed or not !
- ✓ Then select AVD Manager from Window -> AVD Manager.

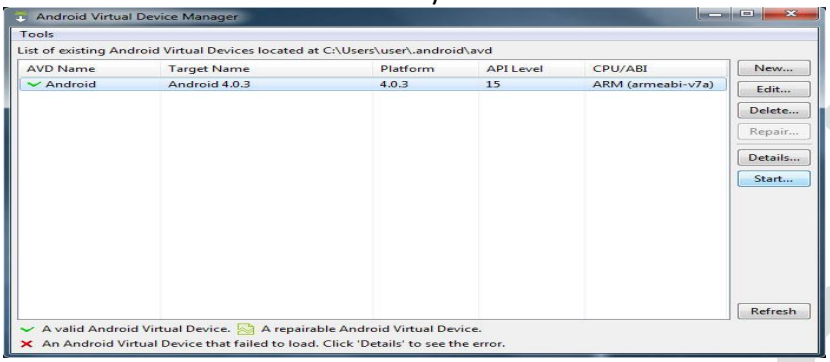




- ✓ Click New... button, name your new AVD, select API target, SD Card size, then click Create AVD button.



- ✓ Click Start... button to start your new AVD. It will takes a few minutes for creating new AVD.



What is Emulator ? How to Start ?

- ✓ The emulator lets you prototype, develop and test Android applications without using a physical device.
- ✓ To use the emulator, first we must create AVD configuration, Remember, If project is used with Android API15 level , then we should use AVD API level also 15.
- ✓ Every Emulator needs one AVD support, because the AVD configuration provides both software and hardware support for Emulator, like it can use the services of the Android platform to invoke other applications, access the network, play audio and video, store and retrieve data, notify the user, and render graphical transitions and themes.
- ✓ The emulator also includes a variety of debug capabilities.
- ✓ The emulator provides dynamic binary translation of device machine code to the OS and processor architecture of your development machine.
- ✓ The Android emulator contains all of the hardware and software features of a real mobile device, except that it cannot place actual phone calls, can't carry.
- ✓ Emulator provides a screen like as real device to display output of our testing application, using this we can easily test our applications.

- ✓ To start an instance of the emulator from the command line, navigate to the tools/ folder of the SDK. Enter emulator command like this:
  - ➢ emulator -avd <avd_name> [<options>]

Emulator Keyboard Shortcuts :

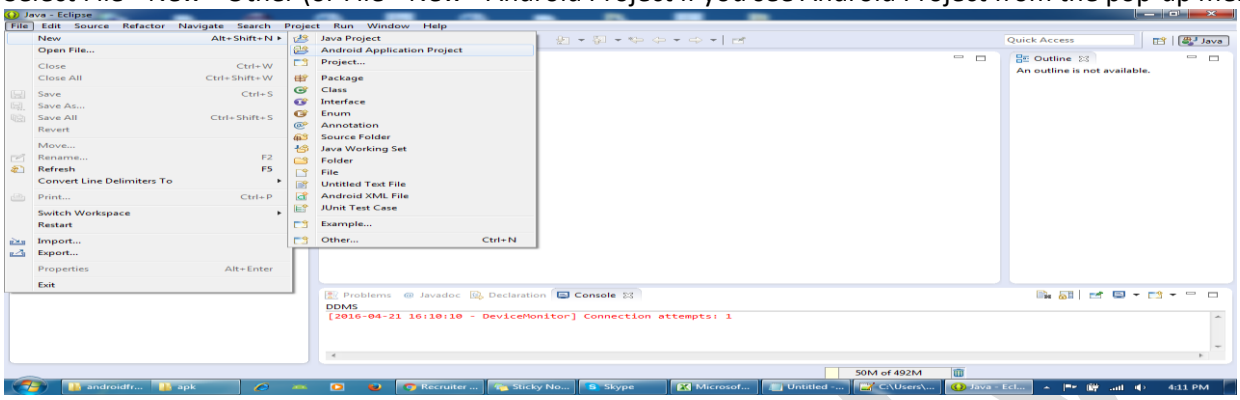
Emulated Device Key	Keyboard Key
Home	HOME
Menu (left softkey)	F2 *or* Page-up button
Star (right softkey)	Shift-F2 *or* Page Down
Back	ESC
Call/dial button	F3
Hangup/end call button	F4
Search	F5
Power button	F7
Audio volume up button	KEYPAD_PLUS, Ctrl-F5
Audio volume down button	KEYPAD_MINUS, Ctrl-F6
Camera button	Ctrl-KEYPAD_5, Ctrl-F3
Switch to previous layout orientation (for example, portrait, landscape)	KEYPAD_7, Ctrl-F11
Switch to next layout orientation (for example, portrait, landscape)	KEYPAD_9, Ctrl-F12
Toggle cell networking on/off	F8
Toggle code profiling	F9 (only with -trace startup option)
Toggle fullscreen mode	Alt-Enter
Toggle trackball mode	F6
Enter trackball mode temporarily (while key is pressed)	Delete
DPad left/up/right/down	KEYPAD_4/8/6/2
DPad center click	KEYPAD_5
Onion alpha increase/decrease	KEYPAD_MULTIPLY(*) / KEYPAD_DIVIDE(/)

4. First Project

4.1. Hello World Program

1. Creating a new Android Project.

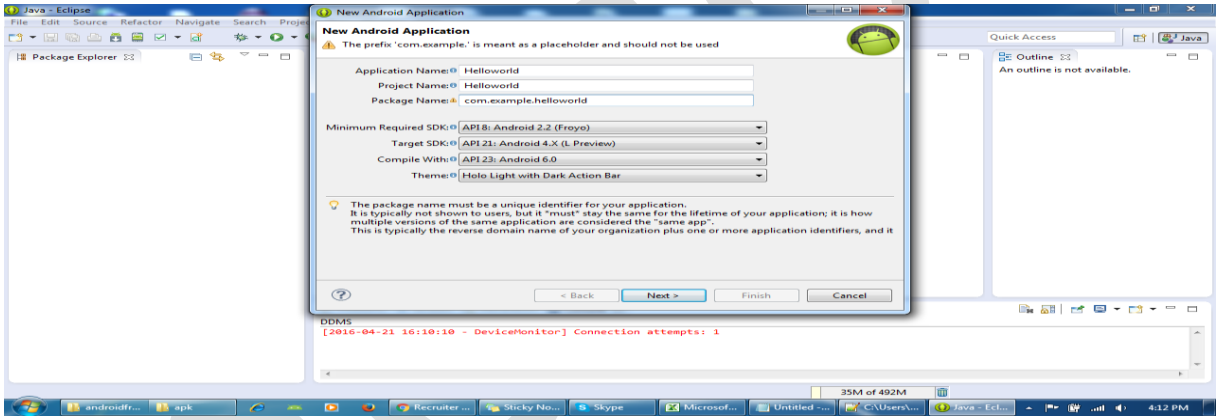
- Select File->New->Other (or File->New->Android Project if you see Android Project from the pop-up menu)



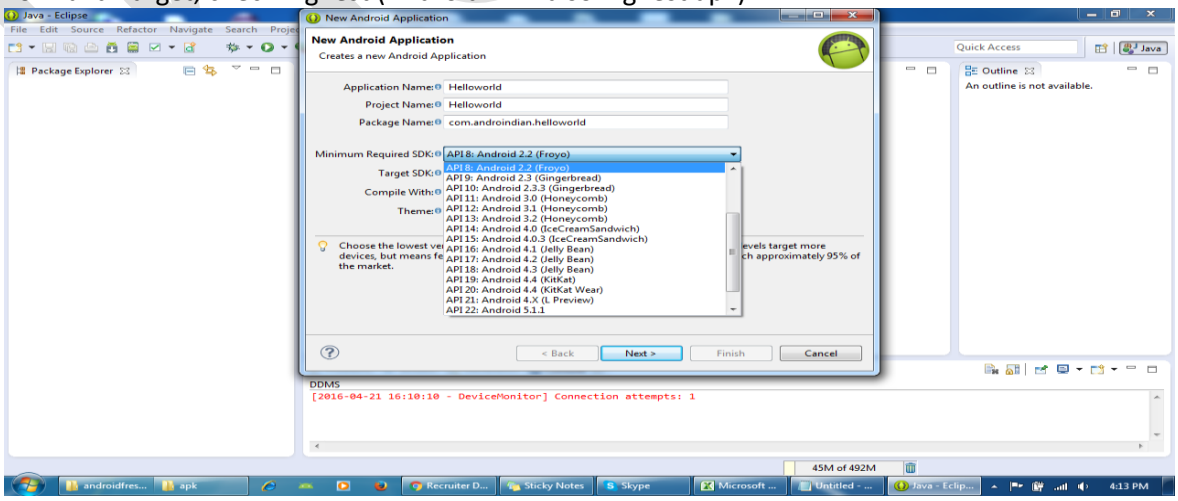
- Select Android->Android Application Project,

2. **Fill in the project details.**

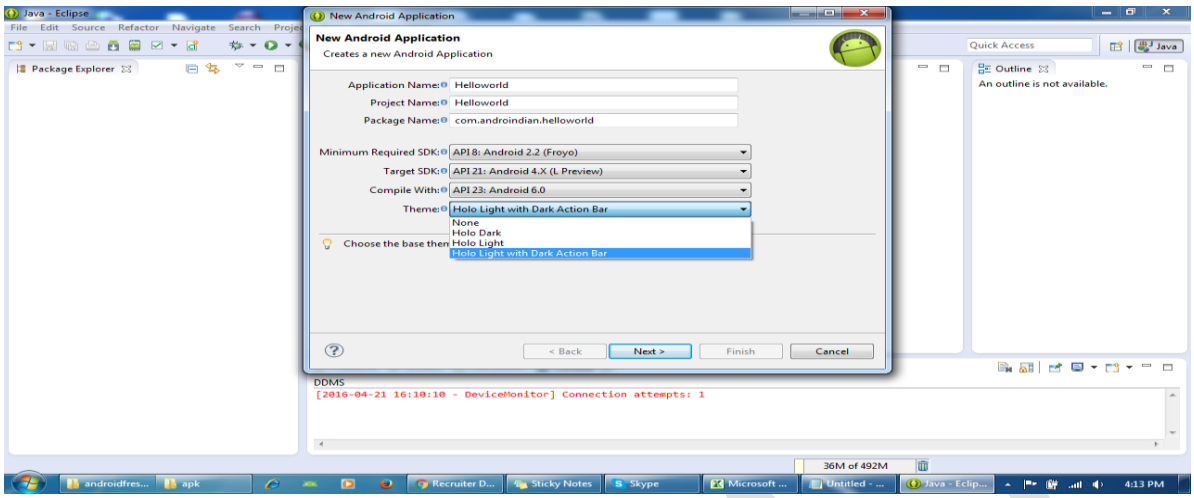
- For the Project name field, enter HelloAndroid (or whatever project name of your choice)



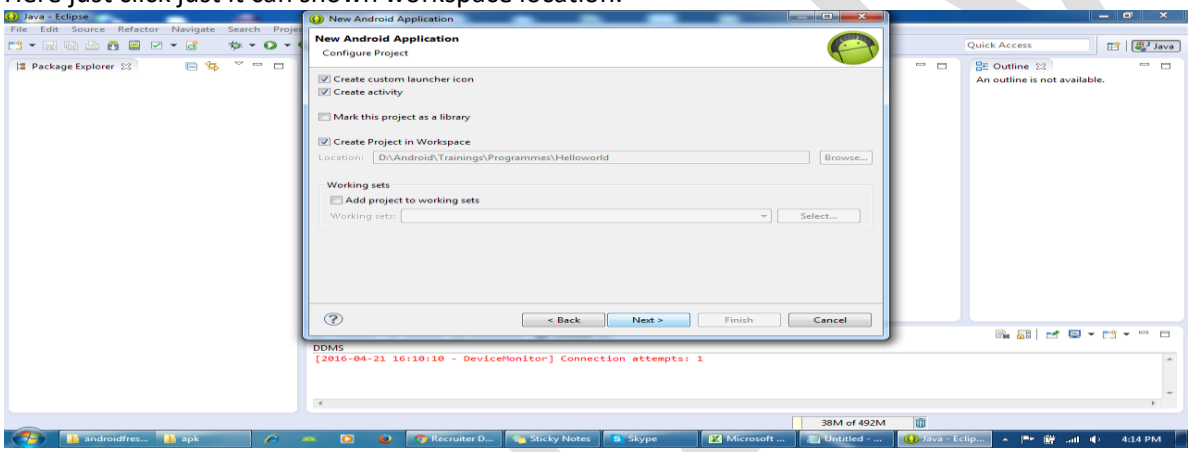
- For Build Target, check highest (Android AVD also highest apk)



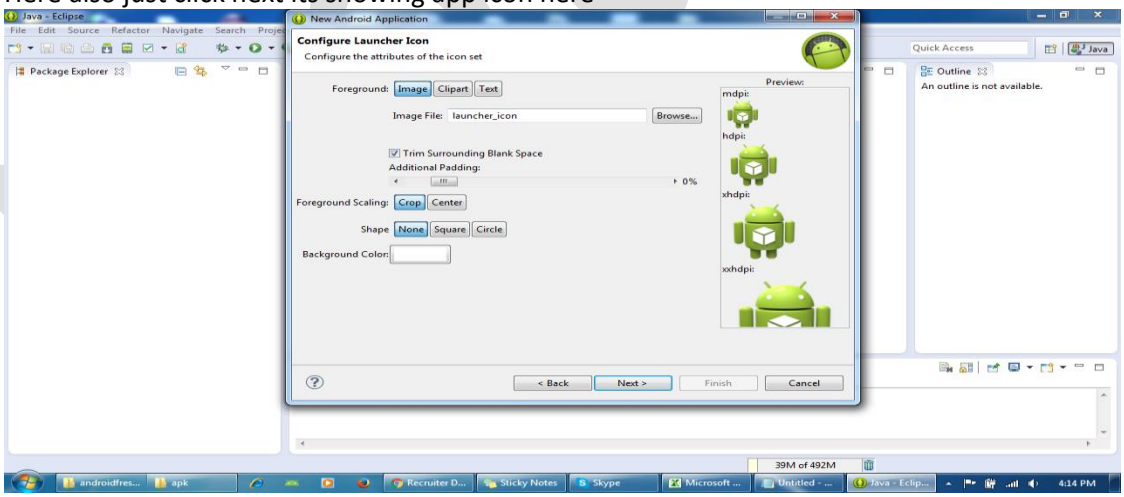
- Select the theme



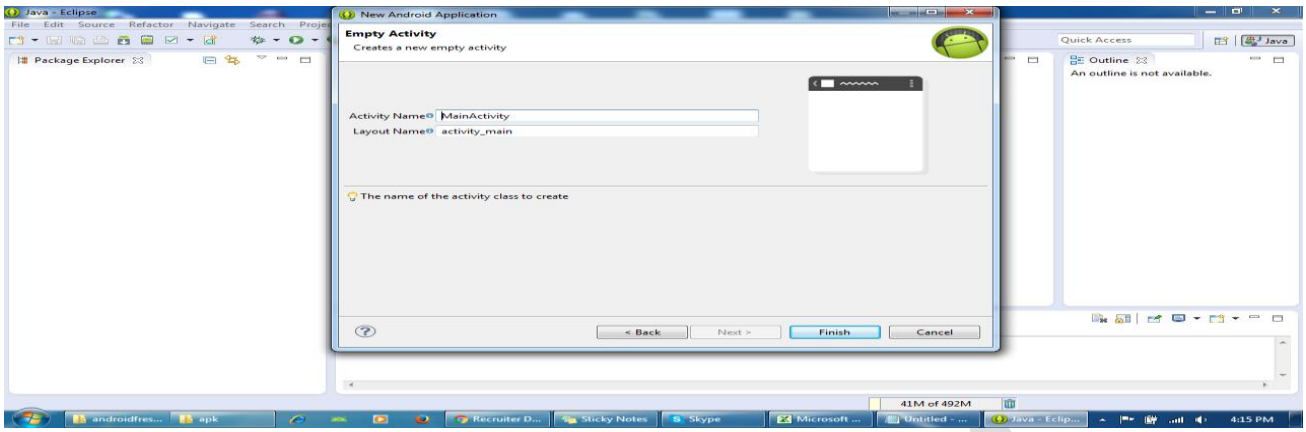
- Here just click just it can shown workspace location.



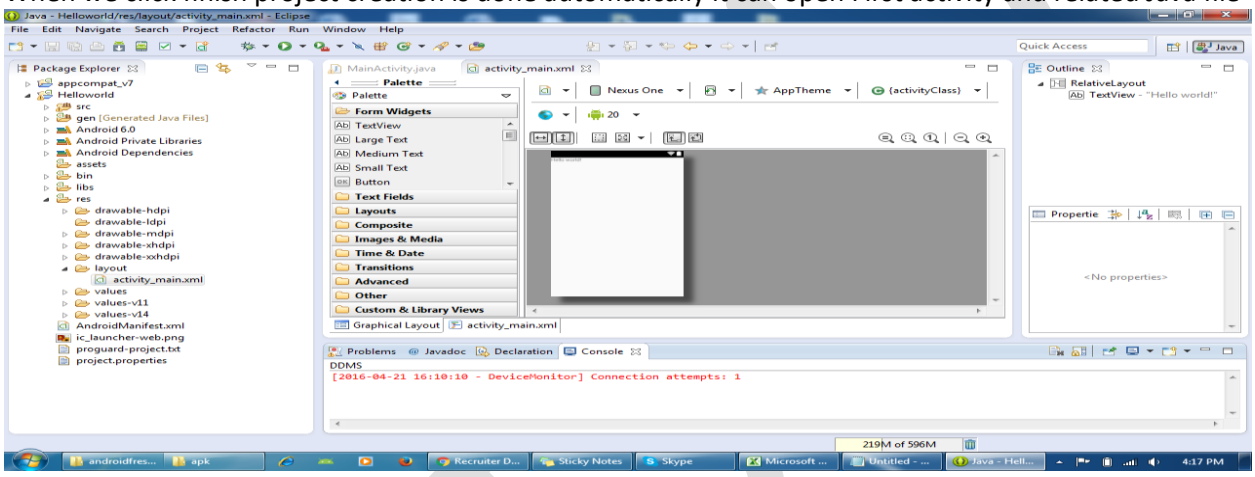
- Here also just click next its showing app icon here



- It is our activity name if you want to change you can change and click finish.

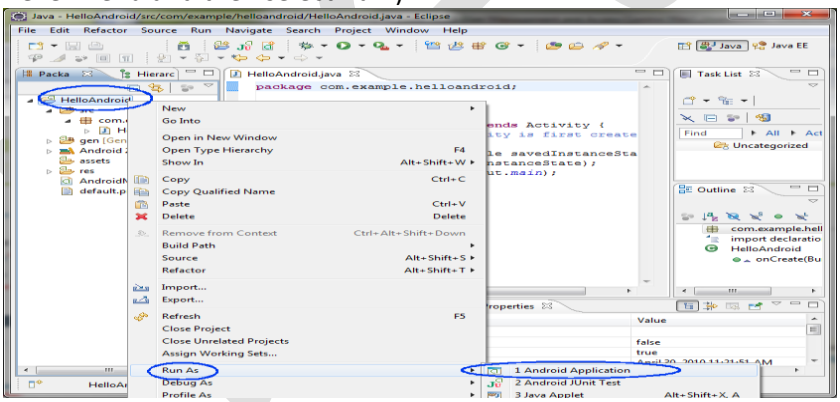


- When we click finish project creation is done automatically it can open First activity and related Java file



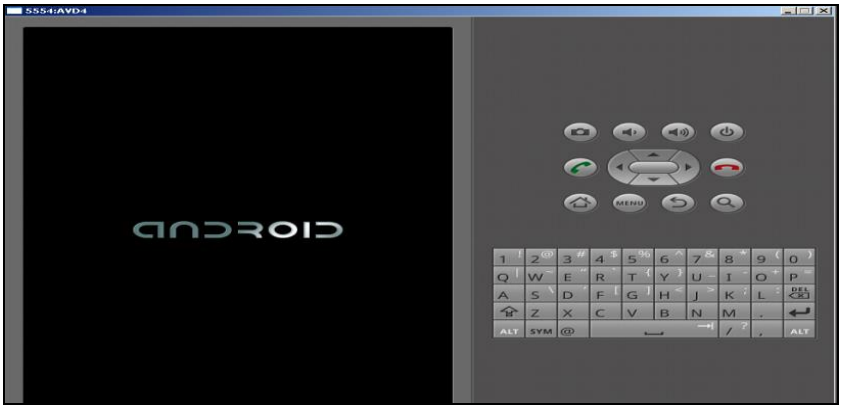
3. **Build and run the application.**

- Right click HelloAndroid project node and select Run As->Android Application. (Or select Run from the top-level menu and then select Run.)



4. **Observe that the Android device emulator gets displayed.**

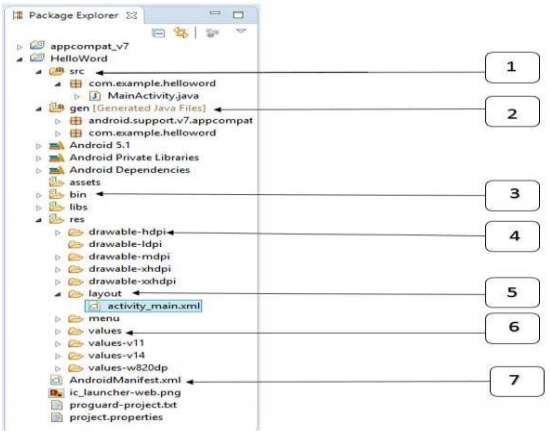
The Eclipse plugin automatically creates a new run configuration for your project and then launches the Android Emulator. Depending on your environment, the Android emulator might take several minutes to boot fully, so please be patient.



5. **Slide down the arrow to the right.**

6. **Observe that the result.**

4.2. Understanding Of Project Structure



Please find the discriptions in below table.

4.3. Project Flow

s.no	Folder, File & Description
1	**Src** This contains the **.java** source files for your project. By default, it includes an *MainActivity.java* source file having an activity class that runs when your app is launched using the app icon.
2	**Gen** This contains the **.R** file, a compiler-generated file that references all the resources found in your project. You should not modify this file.
3	**Bin** This folder contains the Android package files **.apk** built by the ADT during the build process and everything else needed to run an Android application.
4	**res/drawable-hdpi** This is a directory for drawable objects that are designed for high-density screens.
5	**res/layout** This is a directory for files that define your app's user interface
6	**res/values** This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
7	**AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

4.4. Project Output And installation in Real Device (Generating Signed APK)

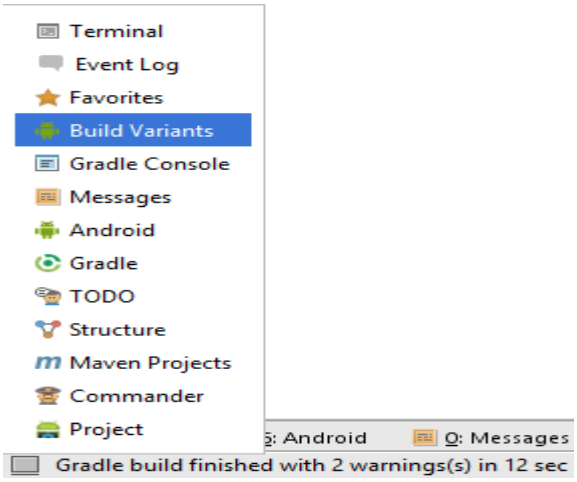
The Release Preparation Process (Android Studio)

Up until this point in the book we have been building application projects in a mode suitable for testing and debugging. Building an application package for release to customers via the Google Play store, on the other hand, requires that some additional steps be taken. The first requirement is that the application be compiled in release mode instead of debug mode. Secondly, the application must be signed with a private key that uniquely identifies you as the application's developer. Finally, the application package must be aligned. This is simply a process by which some data files in the application package are formatted with a certain byte alignment to improve performance.

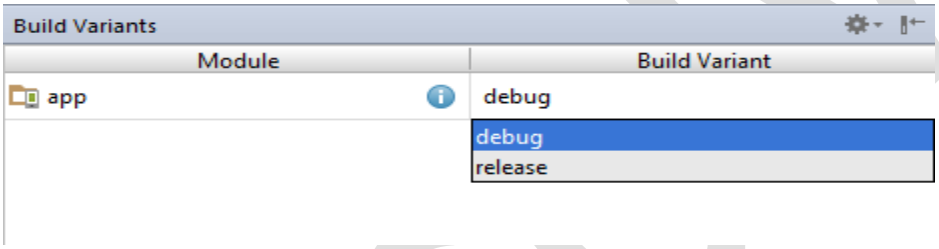
Whilst each of these tasks can be performed outside of the Android Studio environment, the procedures can more easily be performed using the Android Studio build mechanism as outlined in the remainder of this chapter.

Changing the Build Variant

The first step in the process of generating a signed application APK file involves changing the build variant for the project from debug to release. This is achieved using the Build Variants tool window which can be accessed from the tool window quick access menu (located in the bottom left hand corner of the Android Studio main window as shown in Figure



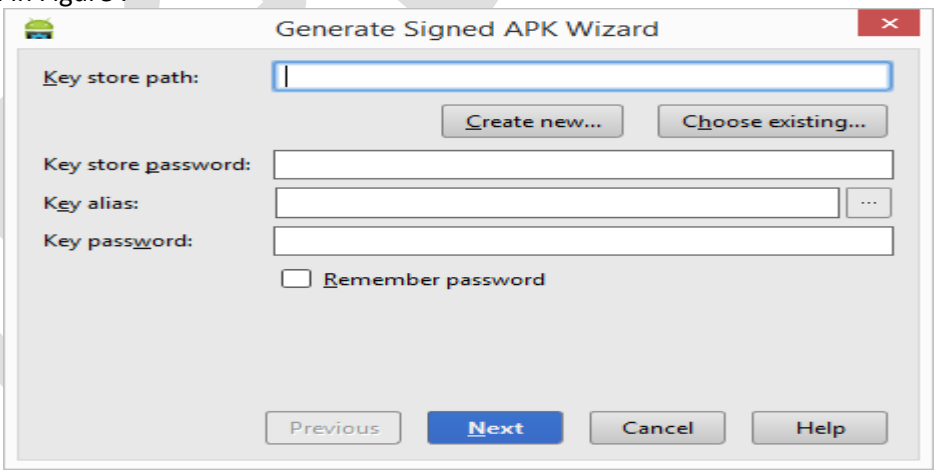
Once the Build Variants tool window is displayed, change the Build Variant settings for all the modules listed from debug to release:



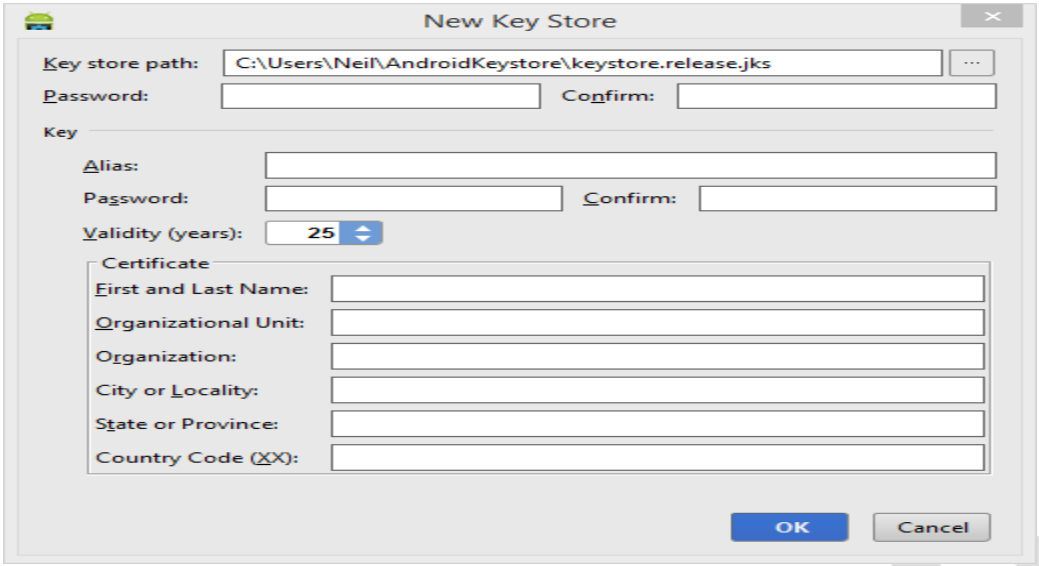
The project is now configured to build in release mode. The next step is to configure signing key information for use when generating the signed application package.

Creating a Keystore File

To create a keystore file, select the Build -> Generate Signed APK... menu option to display the Generate Signed APK Wizard dialog as shown in Figure :



In the event that you have an existing release keystore file, click on the Choose existing... button and navigate to and select the file. In the event that you have yet to create a keystore file, click on the Create new... button to display the New Key Store dialog (Figure 54-4). Click on the button to the right of the Key store path field and navigate to a suitable location on your file system, enter a name for the keystore file (for example release.keystore.jks) and click on the OK button.



The New Key Store dialog is now divided into two sections. The top section relates to the keystore file. In this section, enter a strong password with which to protect the keystore file into both the Password and Confirm fields. The lower section of the dialog relates to the release key that will be stored in the key store file.

Generating a Private Key

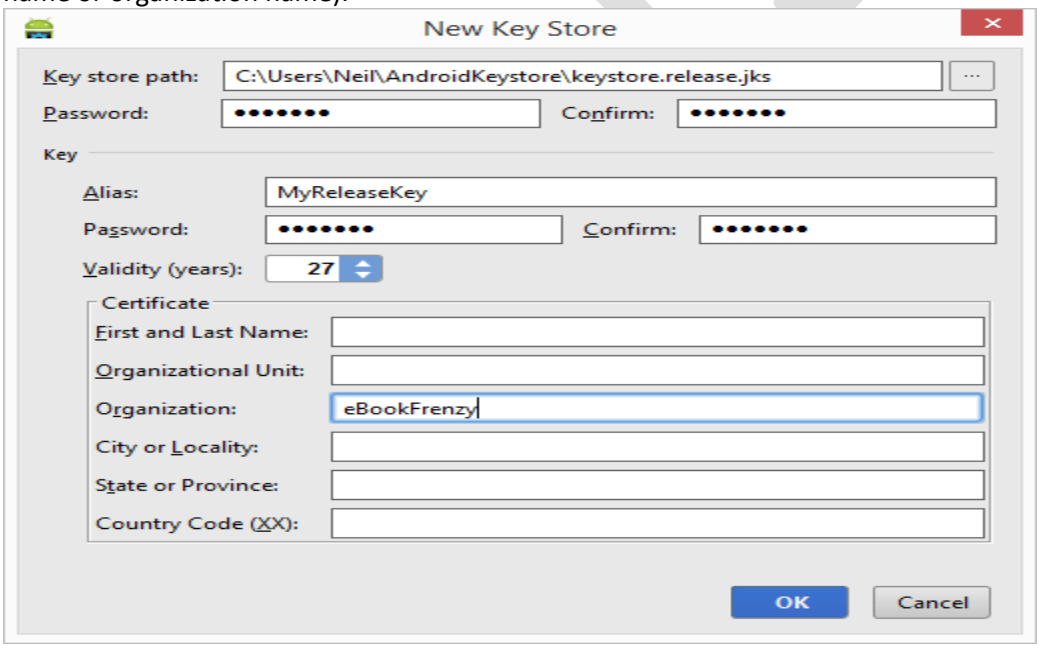
The next step is to generate a new private key which will be used to sign the application package. Within the Key section of the New Key Store dialog, enter the following details:

An alias by which the key will be referenced. This can be any sequence of characters, though only the first 8 are used by the system.

A suitably strong password to protect the key.

The number of years for which the key is to be valid (Google recommends a duration in excess of 27 years).

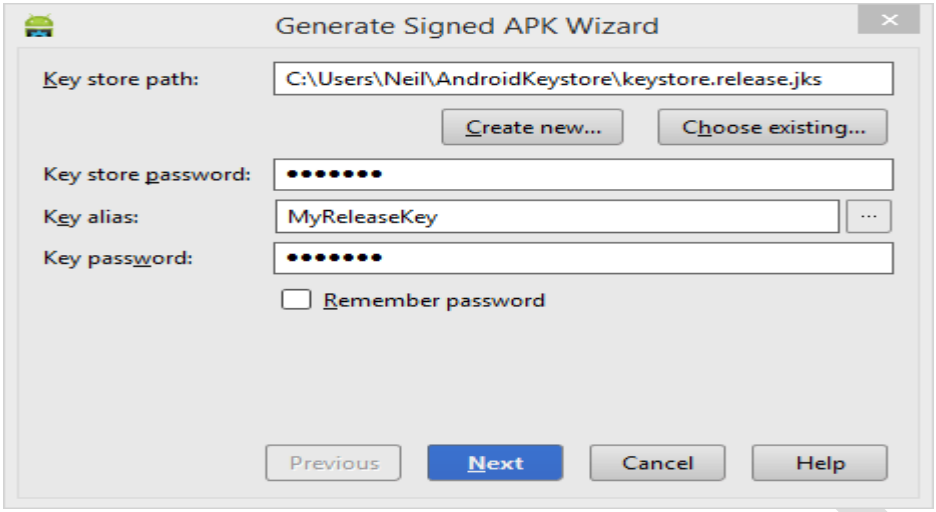
In addition, information must be provided for at least one of the remaining fields (for example your first and last name or organization name).



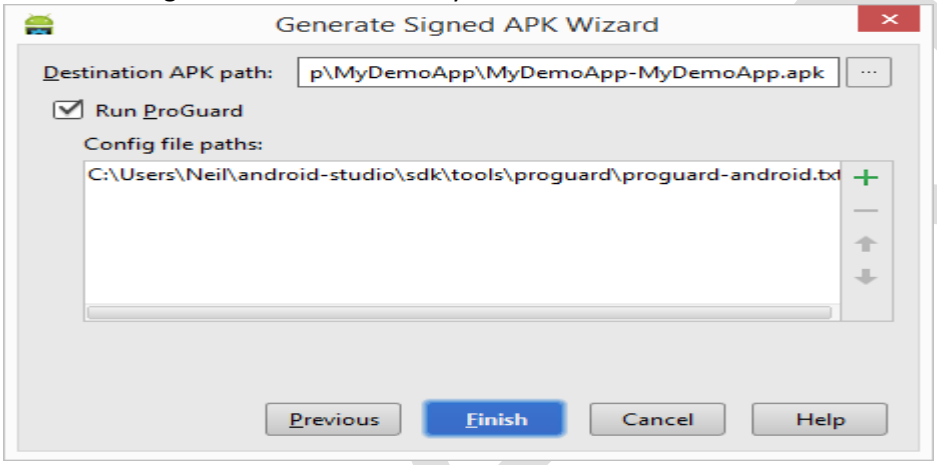
Once the information has been entered, click on the Next button to proceed with the package creation.

Creating the Application APK File

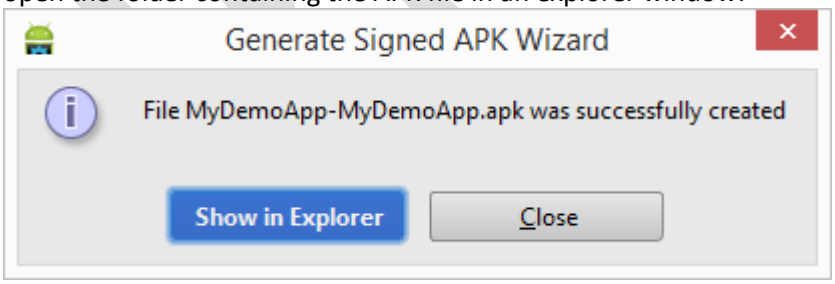
The next task to be performed is to instruct Android Studio to build the application APK package file in release mode and then sign it with the newly created private key. At this point the Generate Signed APK Wizard dialog should still be displayed with the keystore path, passwords and key alias fields populated with information:



Assuming that the settings are correct, click on the Next button to proceed to the APK generation screen (Figure). Within this screen, review the Destination APK path: setting to verify that the location into which the APK file will be generated is acceptable. In the event that another location is preferred, click on the button to the right of the text field and navigate to the desired file system location.



Enable the Run ProGuard checkbox (ProGuard performs a series of optimization and verification tasks that result in smaller and more efficient byte code) before clicking on the Finish button. At this point the Gradle system will compile the application in release mode. Once the build is complete, a dialog will appear providing the option to open the folder containing the APK file in an explorer window:



At this point the application is ready to be submitted to the Google Play store.

The private key generated as part of this process should be used when signing and releasing future applications and, as such, should be kept in a safe place and securely backed up.

The final step in the process of bringing an Android application to market involves submitting it to the Google Play Developer Console. Once submitted, the application will be available for download from the Google Play App Store.

Register for a Google Play Developer Console Account

The first step in the application submission process is to create a Google Play Developer Console account. To do so, navigate to https://play.google.com/apps/publish/signup/ and follow the instructions to complete the registration process. Note that there is a one-time $25 fee to register. Once an application goes on sale, Google will keep 30% of all revenues associated with the application.

Once the account has been created, the next step is to gather together information about the application. In order to bring your application to market, the following information will be required:

Title – The title of the application.

Description – Up to 4000 words describing the application.

Screenshots – Up to 8 screenshots of your application running (a minimum of two is required). Google recommends submitted screenshots of the application running on a 7" or 10" tablet.

Language – The language of the application (the default is US English).

Promotional Text – The text that will be used when your application appears in special promotional features within the Google Play environment.

Application Type – Whether your application is considered to be a game or an application.

Category – The category that best describes your application (for example finance, health and fitness, education, sports etc.).

Locations – The geographical locations into which you wish your application to be made available for purchase.

Contact Information – Methods by which users may contact you for support relating to the application. Options include web, email and phone.

Pricing & Distribution – Information about the price of the application and the geographical locations where it is to be marketed and sold.

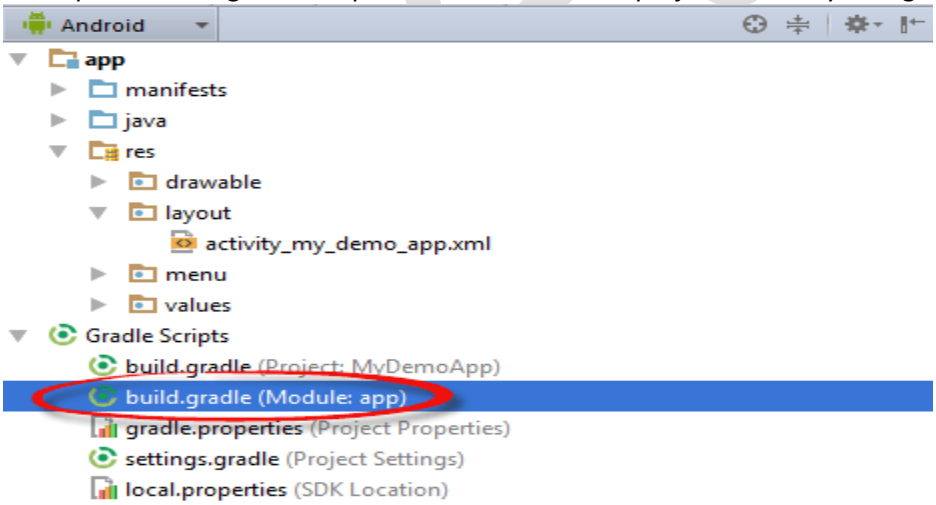
Having collected the above information and prepared the application package file for release, simply follow the steps in the Google Play Developer Console to submit the application for testing and sale.

Uploading New APK Versions to the Google Play Developer Console

The first APK file uploaded for your application will invariably have a version code of 1. If an attempt is made to upload another APK file with the same version code number, the console will reject the file with the following error:

You need to use a different version code for your APK because you already have one with version code 1.

To resolve this problem, the version code embedded into the APK needs to be increased. This is performed in the module level build.gradle file of the project, shown highlighted in Figure 54-9. It is important to note that this is not the top level build.gradle file positioned lower in the project hierarchy listing:



By default, this file will typically read as follows:

```
apply plugin: 'com.android.application'
android {
   compileSdkVersion 21
   buildToolsVersion "21.1.2"
   defaultConfig {
      applicationId "com.ebookfrenzy.mydemoapp"
```

```
        minSdkVersion 21
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
```

To change the version code, simply change the number declared next to versionCode. To also change the version number displayed to users of your application, change the versionName string. For example:

```
    versionCode 2
    versionName "2.0"
```

Having made these changes, rebuild the APK file and perform the upload again.

Summary

Before an application can be submitted to the Google Play store, it must first be built in release mode, signed with a private certificate and the resulting APK package file subjected to a process referred to as alignment. As outlined in this chapter, all of these steps can be performed with relative ease through the use of the Android Studio build system.

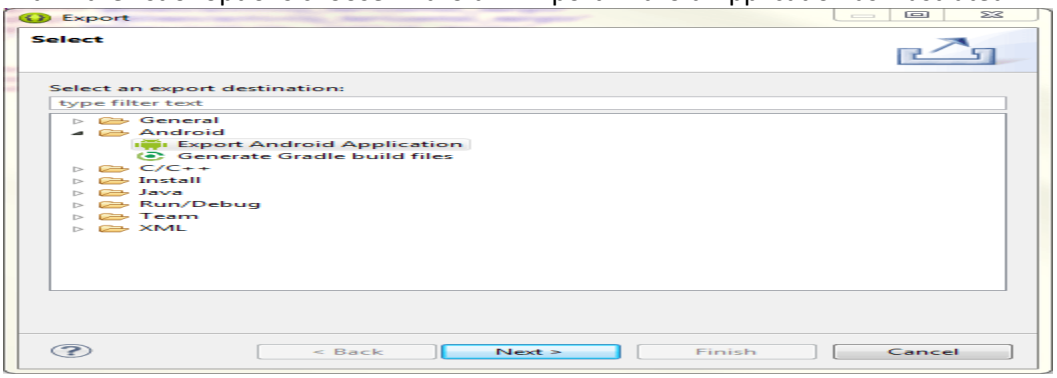
The Release Preparation Process(Eclipse)

Up until this point in the book we have been building application projects in a mode suitable for testing and debugging. Building an application package for release to customers via the Google Play store, on the other hand, requires that some additional steps be taken. The first requirement is that the application be compiled in release mode instead of debug mode. Secondly, the application must be signed with a private key that uniquely identifies you as the application's developer. Finally, the application package must be aligned. This is simply a process by which some data files in the application package are formatted with a certain byte alignment to improve performance.

Whilst each of these tasks can be performed outside of the Eclipse ADT environment, the tasks can more easily be performed using the Export Wizard as outlined in the remainder of this chapter. Note that Eclipse assumes that the JDK tools are in the PATH environment of your system as outlined in Setting up an Android Development Environment.

Accessing the Export Wizard

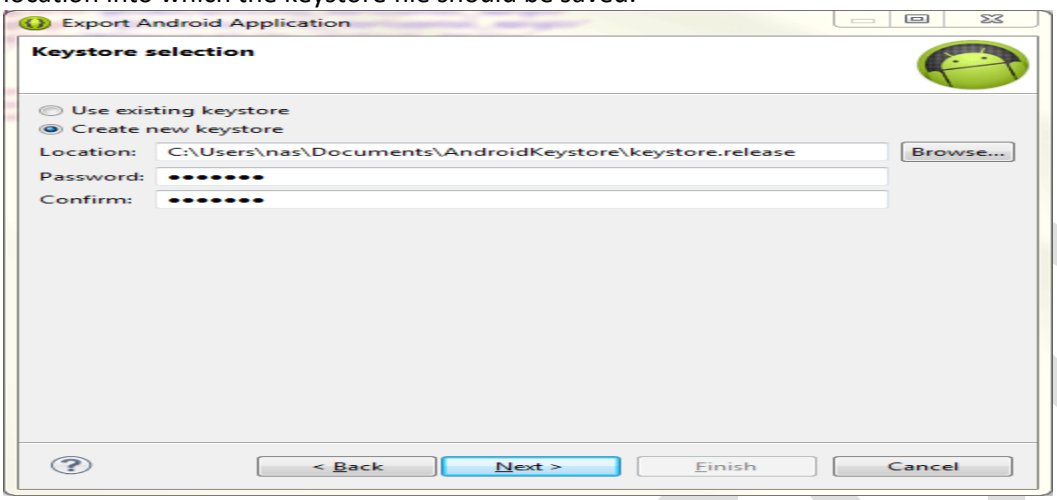
To gain access to the Export Wizard, launch the Eclipse environment and select the project to be released from the Package Explorer panel. Once selected, select the File -> Export menu option to display the Export dialog. From within the list of options choose Android -> Export Android Application as illustrated in Figure



Click Next to proceed to the Project Checks screen where the project will be scanned to ensure that there are no potential problems that might prevent the application from being exported. In the event that no errors are found, click Next once again to proceed to the Keystore selection screen.

Creating a Keystore File

The Keystore selection screen (Figure) provides the option either to use an existing keystore file, or to generate a new file. If you already have a keystore file simply click on the Browse button to locate and load the file. If you have yet to create a file, however, select the Create new keystore menu option and enter a file system path and file name location into which the keystore file should be saved.



Next, enter a strong password with which to protect the keystore file into both the Password and Confirm fields before clicking on Next.

Generating a Private Key

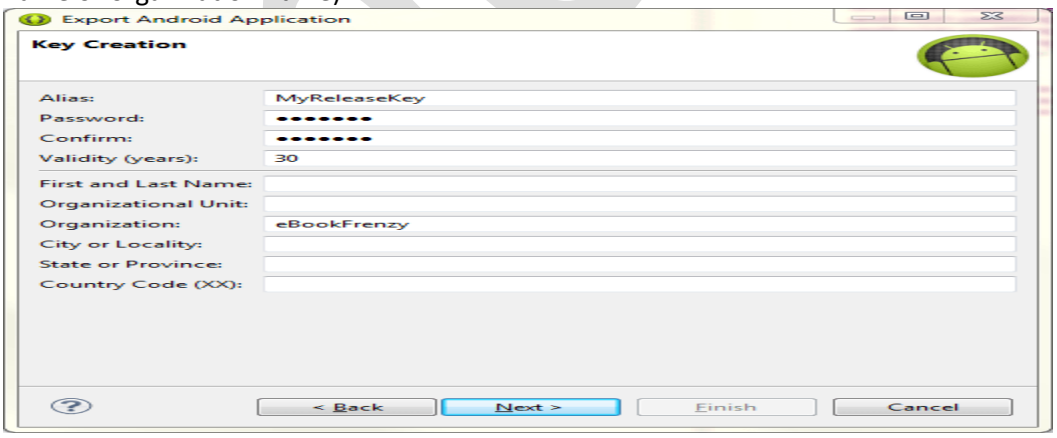
The next step is to generate a new private key which will be used to sign the application package. At this point, the Key Creation screen (Figure 44-3) of the Export Wizard should be displayed. Within this screen, enter the following information:

An alias by which the key will be referenced. This can be any sequence of characters, through only the first 8 are used by the system.

A suitably strong password to protect the key.

The number of years for which the key is to be valid (Google recommends a duration in excess of 27 years).

In addition, information must be provided for at least one of the remaining fields (for example your first and last name or organization name).



Once the information has been entered, click on the Next button to proceed with the package creation.

Creating the Application APK File

The next task to be performed is to instruct Eclipse to build the application APK package file in release mode and then sign it with the newly created private key. At this point the Destination and key/certificate checks screen should be displayed. Using the Browse…button, navigate to a suitable folder into which the application APK file should be

created. Once a path and file name have been specified, click on Finish to generate the APK and sign it with your private key.
During this process, output similar to that outlined below will appear within the Console panel:
[2013-06-13 10:34:39 - ReleaseTest] New keystore C:\Users\nas\Documents\AndroidReleaseAPK\ReleaseTest.apk has been created.

```
[2013-06-13 10:34:39 - ReleaseTest] Certificate fingerprints:
[2013-06-13 10:34:39 - ReleaseTest]   MD5 : FA:65:D9:F4:CA:17:DD:24:C2:16:FB:D4:77:22:AB:46
[2013-06-13 10:34:39 - ReleaseTest]   SHA1: D1:E1:33:43:99:FF:CF:DF:65:71:AA:EE:22:A7:DA:CF:E9:61:45:13
```

On completion of the generation process, the keystore file and APK file will have been generated into the designated locations of the file system. At this point the application is ready to be submitted to the Google Play store.
The private key generated as part of this process should be used when signing and releasing future application and, as such, should be kept in a safe place and securely backed up.
The final step in the process of bringing an Android application to market involves submitting it to the Google Play Developer Console. Once submitted, the application will be available for download from the Google Play App Store.

Register for a Google Play Developer Console Account

The first step in the application submission process is to create a Google Play Developer Console account. To do so, navigate to https://play.google.com/apps/publish/signup/ and follow the instructions to complete the registration process. Note that there is a one-time $25 fee to register. Once an application goes on sale, Google will keep 30% of all revenues associated with the application.
Once the account has been created, the next step is to gather together information about the application. In order to bring your application to market, the following information will be required:
Title – The title of the application.
Description – Up to 4000 words describing the application.
Screenshots – Up to 8 screenshots of your application running (a minimum of two is required). Google recommends submitted screenshots of the application running on a 7” or 10” tablet.
Language – The language of the application (the default is US English).
Promotional Text – The text that will be used when your application appears in special promotional features within the Google Play environment.
Application Type – Whether your application is considered to be a game or an application.
Category – The category that best describes your application (for example finance, health and fitness, education, sports etc).
Locations – The geographical locations into which you wish to your application to be made available for purchase.
Contact Information – Methods by which users may contact you for support relating to the application. Options include web, email and phone.
Having collected the above information and prepared the application package file for release, simply follow the steps in the Google Play Developer Console to submit the application for sale.
Summary
Before an application can be submitted to the Google Play store, it must first be built in release mode, signed with a private certificate and the resulting APK package file subjected to a process referred to as alignment. As outlined in this chapter, all of these steps can be performed with relative ease through the use of the Eclipse Export Wizard.

5. Android Studio Installation

5.1. Installation

Before we get started, you'll need to ensure that you have installed JDK 6 or higher (JDK7 is required for Android 5.0 and higher) on your PC.

If the JDK doesn't show up, or you have an older version and would like to upgrade with the following link http://www.oracle.com/technetwork/java/javase/downloads/index.html

How to Install Android Studio on Windows

1. Download and launch the .exe file to your PC from the Android Studio home page. https://developer.android.com/sdk/index.html
2. Follow the instructions on the setup wizard to install Android Studio.
3. If asked to point to where Java is installed, you need to set an environment variable in order to direct the installer to the proper location. To do that, select Start menu > Computer > System Properties > Advanced System Properties. From there you'll open the "Advanced" tab and click "Environment Variables." Here you'll add a new system variable titled JAVA_HOME that points to your JDK folder.

How to Set Up Android Studio on Mac OS X

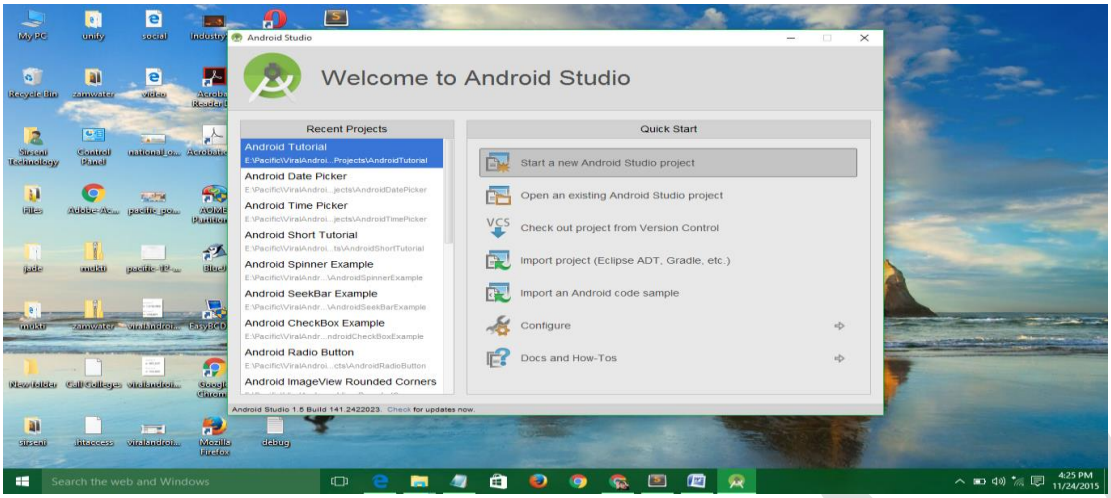
1. Download the installer to your PC and launch the .dmg file from the Android Studio home page. http://developer.android.com/sdk/index.html
2. Drag and drop the .dmg into your Applications folder.
3. Open Android Studio and follow the instructions from the setup wizard.

If you get a warning saying that the file is damaged and should be moved to the trash, go to System Preferences > Security & Privacy and under the "Allow applications downloaded from" section select "Anywhere." From here you can repeat Step 3 and install the program.

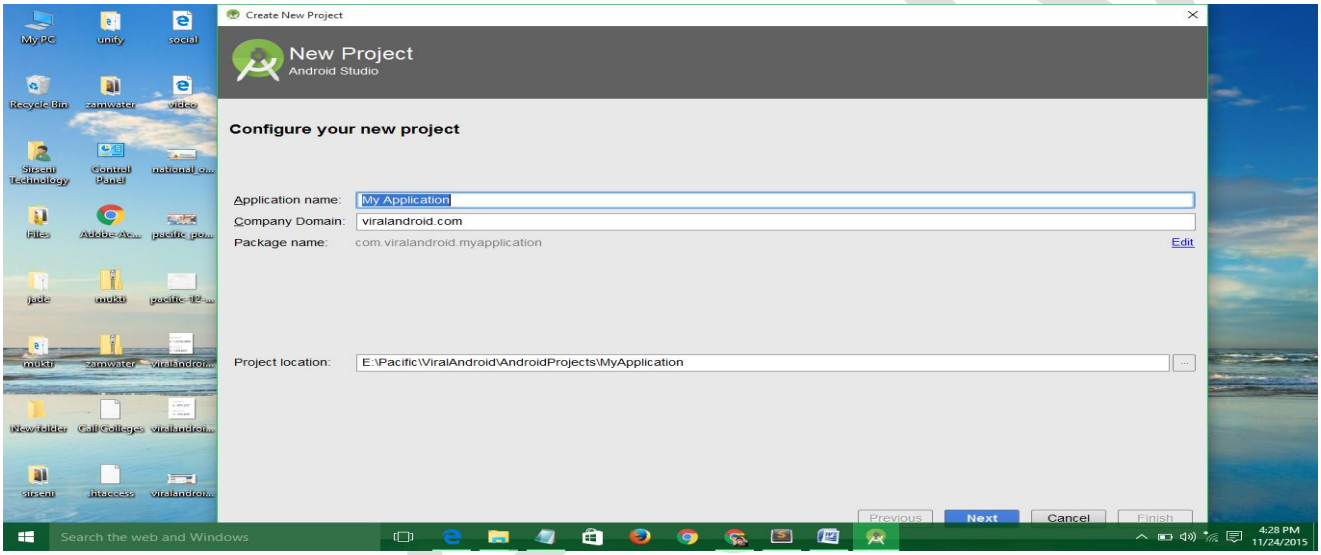
5.2 Hello world Project:

Create Android Application :

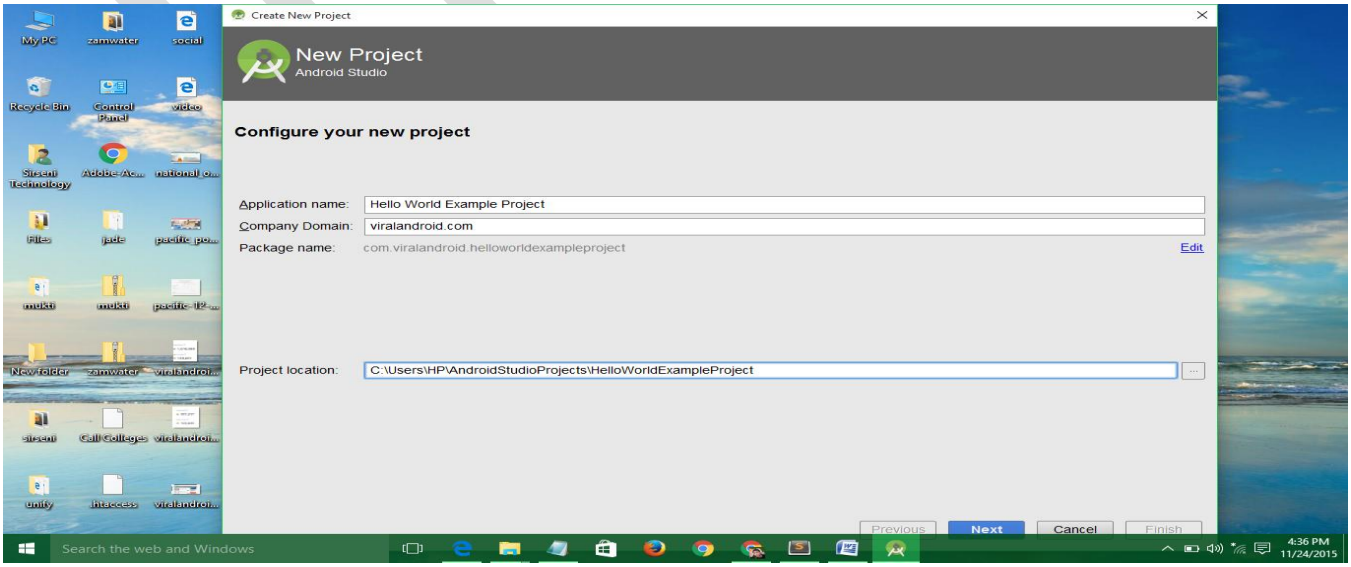
To create a new android project from android studio click Start a new Android Studio Project from Quick Start which will looks like below screenshot.



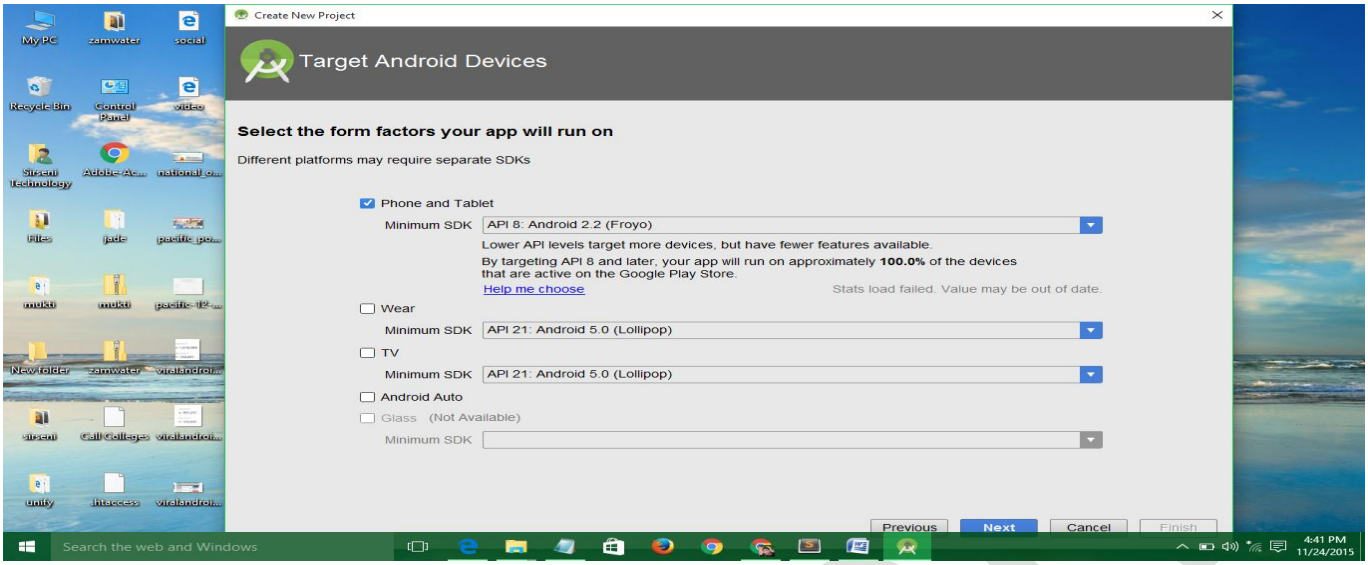
After clicking start a new android studio project, new widow will appear there which will look like below.



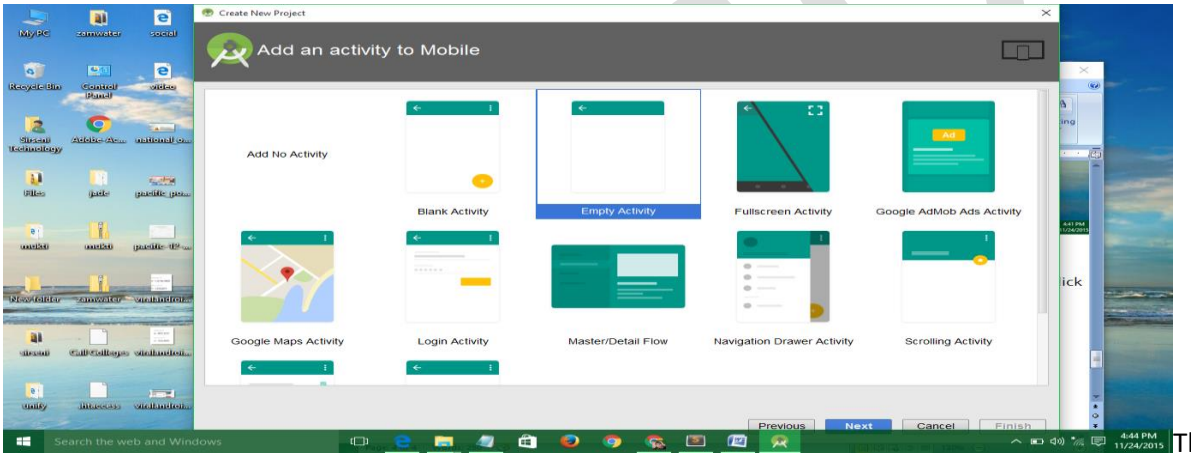
Here, you have to fill up application name, company domain, package name and choose project location. Just write Hello World Example Project in application name, viralandroid.com in company domain, com.viralandroid.helloworldexampleproject in package name and use default directory of project location and clickNext button.



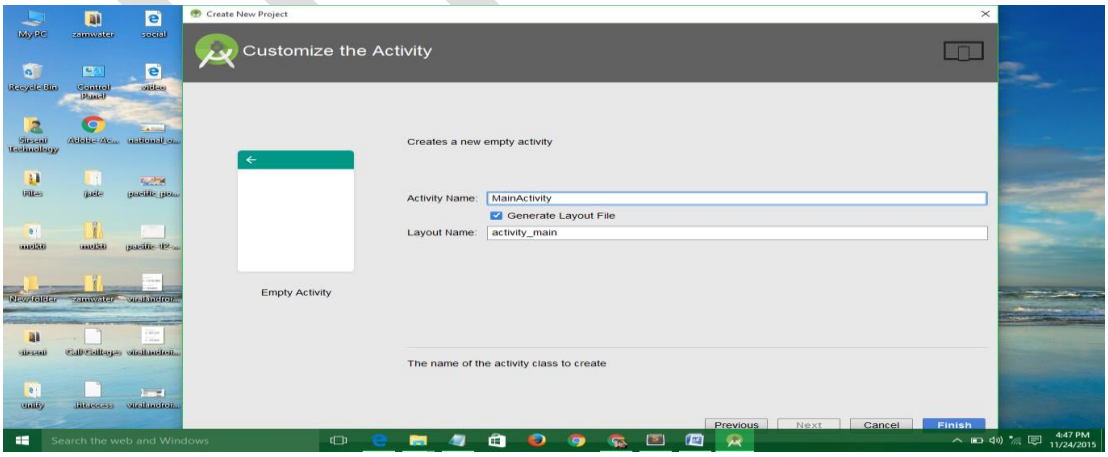
Now, you will see the place to choose different android platforms and minimum SDK (minimum API level). Just check on Phone & Tablet and choose API 8: Android 2.2 (Froyo) in minimum SDK and click Next button.



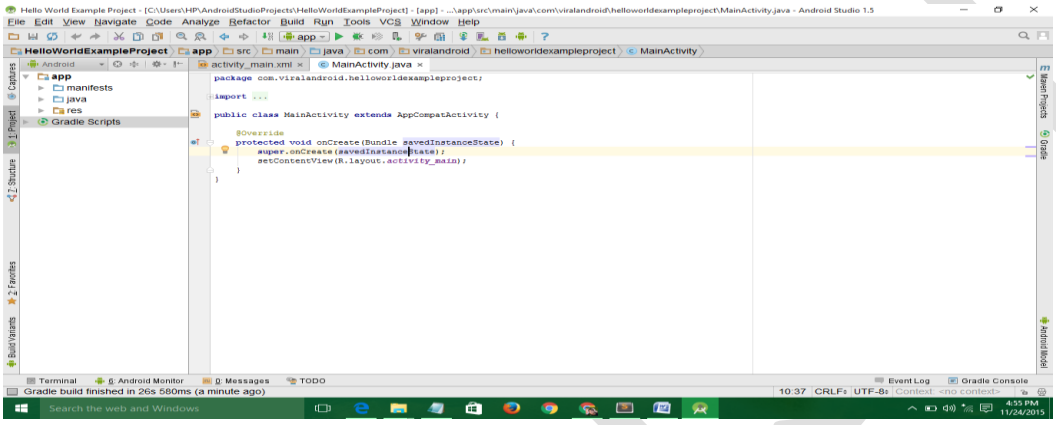
Then you will see add an activity to mobile window, just select Empty Activity and click Next button.



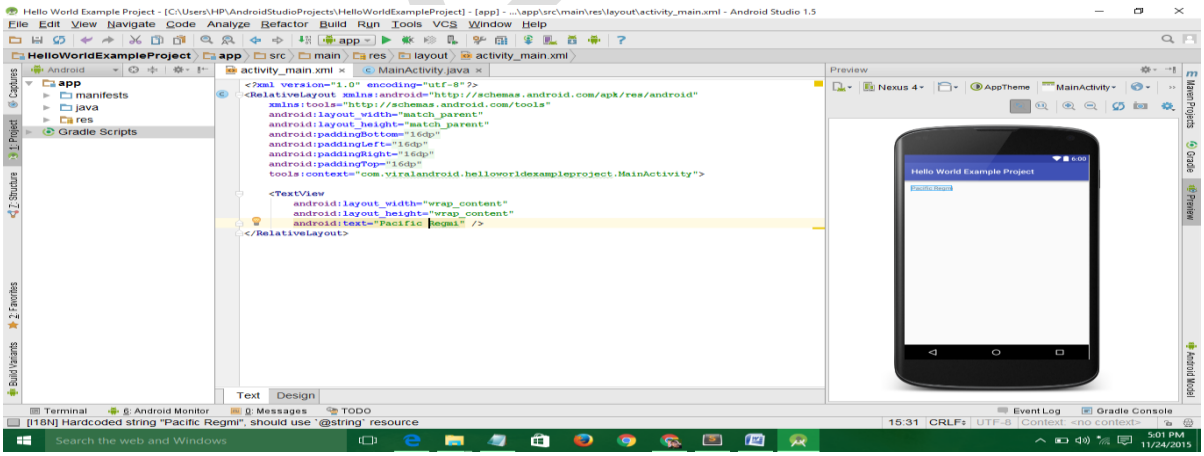
Then, you will see customize the activity window just use default activity name & Layout name and click Finish button.



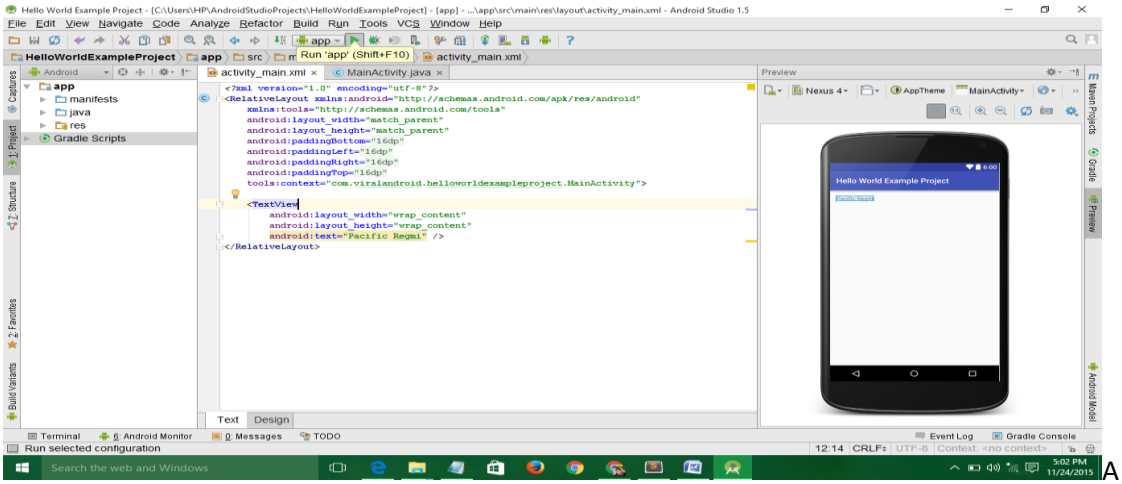
After opening android studio, you will see like the screenshot given below.



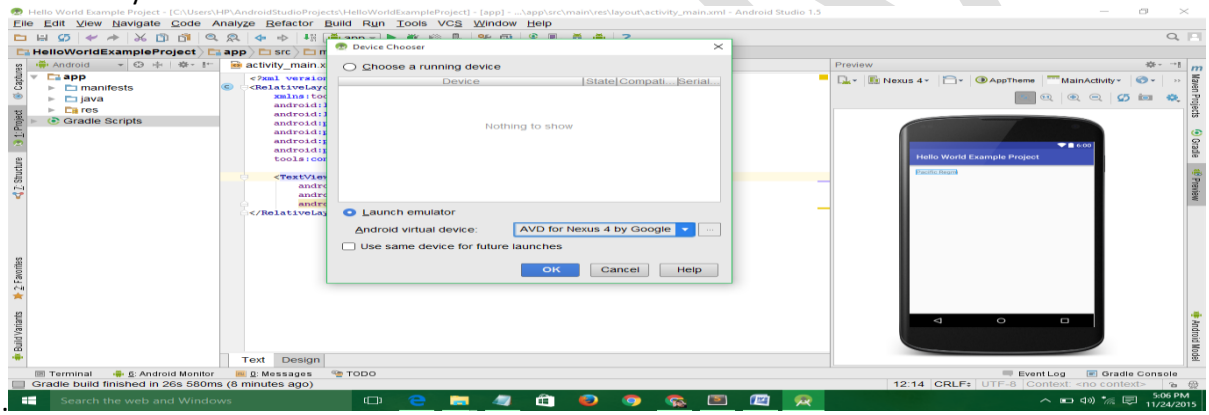
And click the activity_main.xml tab. By default, android uses Hello World! In each new project, replace Hello World! with your name in the TextView like below.



To run your application, click Run icon from toolbar

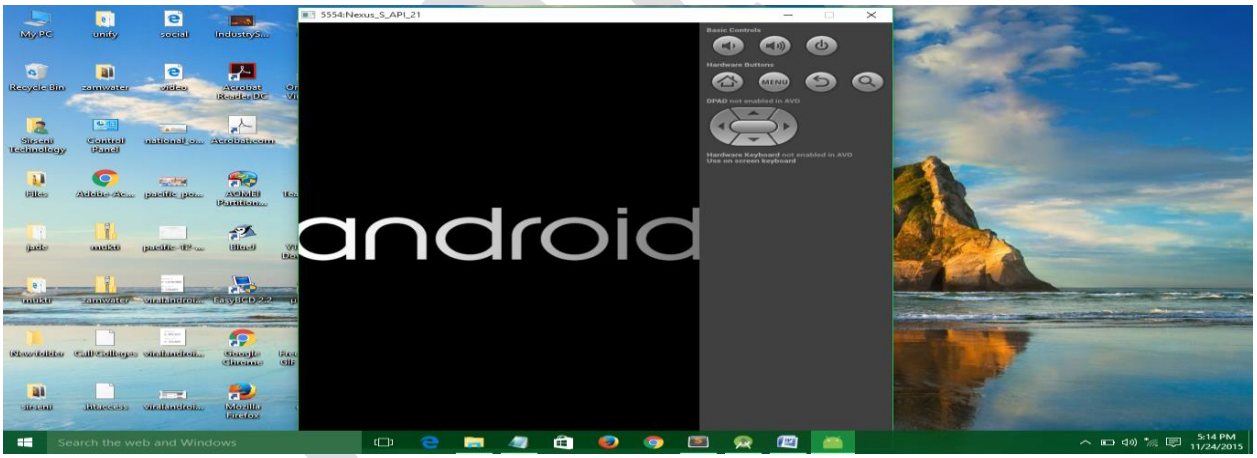


A new window will appear there where you can choose running device or launch new emulator.Choose launch emulator and choose one of the virtual devices you have created before and

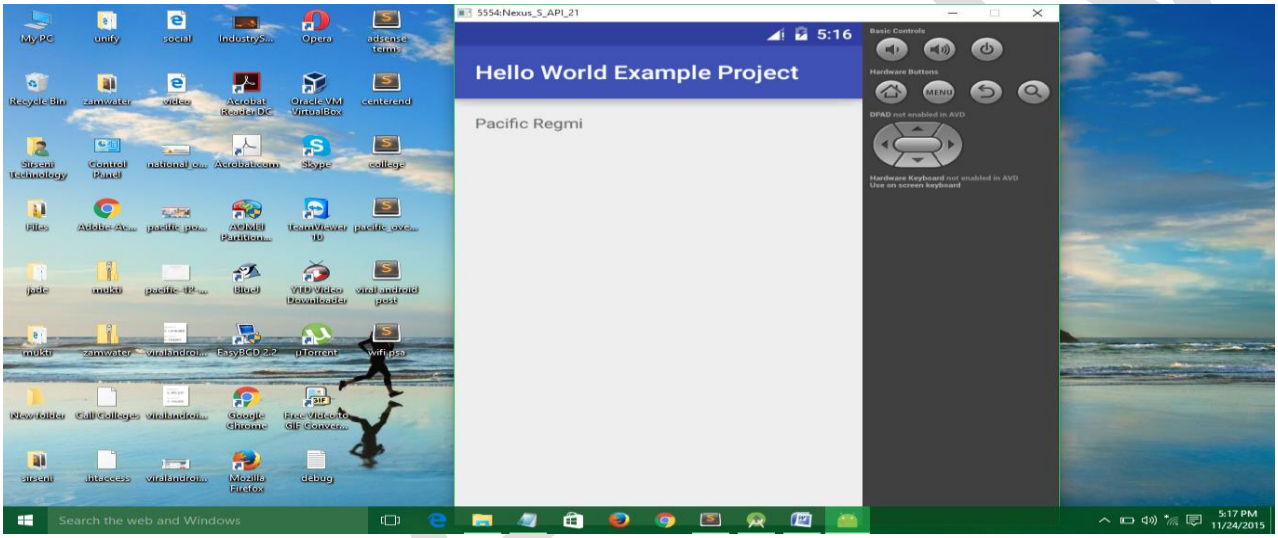


click Ok button.

It takes few minutes to start your emulator for the first time.



Now, unlock your emulator screen to see your first application with your name which will look like below screenshot.



6. Android Application Components

Activity:

An activity represents a single screen with a user interface just like window or frame of Java. Android activity is the subclass of ContextThemeWrapper class

Service

A **service** is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed.

BroadCast Reciver

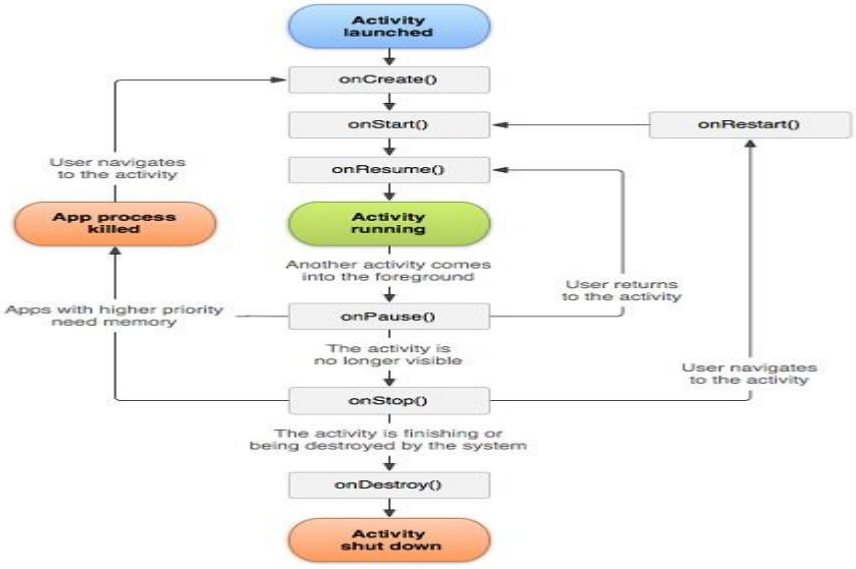
Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network.

6.1. Activity & Life Cycle

If you have worked with C, C++ or Java programming language then you must have seen that your program starts from **main()** function. Very similar way, Android system initiates its program with in an **Activity** starting with a call on*onCreate()* callback method. There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the below Activity life cycle diagram:



The Activity class defines the following call backs i.e. events. You don't need to implement all the callbacks methods. However, it's important that you understand each one and implement those that ensure your app behaves the way users expect.

Call Back	Discription
onCreate()	This is the first callback and called when the activity is first created.
onStart()	This callback is called when the activity becomes visible to the user.
onResume()	This is called when the user starts interacting with the application.
onPause()	The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
onStop()	This callback is called when the activity is no longer visible.
onDestroy()	This callback is called before the activity is destroyed by the system.
onRestart()	This callback is called when the activity restarts after stopping it.

6.2. Activity lifecycle example Program

```
package com.raj;
import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
public class MainActivity extends Activity {
  String msg = "Android : ";
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.d(msg, "The onCreate() event");
  }
  /** Called when the activity is about to become visible. */
  @Override
  protected void onStart() {
    super.onStart();
    Log.d(msg, "The onStart() event");
  }
  /** Called when the activity has become visible. */
  @Override
  protected void onResume() {
    super.onResume();
    Log.d(msg, "The onResume() event");
  }
  /** Called when another activity is taking focus. */
  @Override
  protected void onPause() {
    super.onPause();
    Log.d(msg, "The onPause() event");
  }
  /** Called when the activity is no longer visible. */
  @Override
  protected void onStop() {
    super.onStop();
    Log.d(msg, "The onStop() event");
  }
  /** Called just before the activity is destroyed. */
  @Override
  public void onDestroy() {
    super.onDestroy();
    Log.d(msg, "The onDestroy() event");
  }
}
```

An activity class loads all the UI component using the XML file available in res/layout folder of the project. Following statement loads UI components from res/layout/activity_main.xml file:
setContentView(R.layout.activity_main);

An application can have one or more activities without any restrictions. Every activity you define for your application must be declared in your AndroidManifest.xml file and the main activity for your app must be declared in the manifest with an <intent-filter> that includes the MAIN action and LAUNCHER category as follows:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.helloworld"
  android:versionCode="1"
  android:versionName="1.0" >
  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="22" />

  <application
     android:icon="@drawable/ic_launcher"
     android:label="@string/app_name"
     android:theme="@style/AppTheme" >
     <activity
       android:name=".MainActivity"
       android:label="@string/title_activity_main" >
       <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER"/>
       </intent-filter>
     </activity>
  </application>
</manifest>
```

If either the MAIN action or LAUNCHER category are not declared for one of your activities, then your app icon will not appear in the Home screen's list of apps.

Let's try to run our modified Hello World! application we just modified. I assume you had created your AVD while doing environment setup. To run the app from Eclipse, open one of your project's activity files and click Run Eclipse Run Icon icon from the toolbar. Eclipse installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display Emulator window and you should see following log messages in LogCat window in Eclipse IDE:

```
07-19 15:00:43.405: D/Android :(866): The onCreate() event
07-19 15:00:43.405: D/Android :(866): The onStart() event
07-19 15:00:43.415: D/Android :(866): The onResume() event
```

Android LotCat Window

Let us try to click Red button Android Red Button on the Android emulator and it will generate following events messages in LogCat window in Eclipse IDE:

```
07-19 15:01:10.995: D/Android :(866): The onPause() event
07-19 15:01:12.705: D/Android :(866): The onStop() event
```

Let us again try to click Menu button Android Menu Button on the Android emulator and it will generate following events messages in LogCat window in Eclipse IDE:

```
07-19 15:01:13.995: D/Android :(866): The onStart() event
07-19 15:01:14.705: D/Android :(866): The onResume() event
```

Next, let us again try to click Back button Android Back Button on the Android emulator and it will generate following events messages in LogCat window in Eclipse IDE and this completes the Activity Life Cycle for an Android Application.

```
07-19 15:33:15.687: D/Android :(992): The onPause() event
07-19 15:33:15.525: D/Android :(992): The onStop() event
07-19 15:33:15.525: D/Android :(992): The onDestroy() event
```

7. Android Resources

7.1. Handling Application Resources :

Resources play a key role in Android Architecture. A resource in Android is a file (like a music file)or a value(like the title of a dialog box) that is bound to an executable application. These files and values are bound to the executable code so that we can change them without need of recompiling the application. Examples of resources include strings, colors, and bitmaps.

We can create and store our resource files under the appropriate subdirectory "res/" directory in our project. Android has a resource compiler (AAPT-Android Asset Packaging Tool) that compiles resources according to which subfolder they are in, and the format of the file.

Why We need Resources :

- ✓ The resources are bound to the application so that we can change them without needing to change the source code or recompile the application.
- ✓ The Android Generates an ID for each resource file so we can access them directly and easily in our java code. All the resources IDs are added to the R.Java file.
- ✓ Using resources is very useful in Localization and Internationalization of the application while develop multilingual applications.
- ✓ Resources includes not just the labels but it can include alignment, directions images or any kind of files.

Resources class :

- ✓ Class for accessing an application's resources, defined in android.content.res.* package.
- ✓ The Android resource system refers all non-code assets associated with an application, like images, audio, video ..etc. We can use this class to access our application's resources. We can generally acquire the Resources instance associated with our application with getResources().

Method :

```
AssetManager getAssets();
 boolean getBoolean(int id);
 int getColor(int id);
float getDimension(int id);
int[] getIntArray(int id);
int getInteger(int id);
Movie getMovie(int id);
String getString(int id);
String[] getStringArray(int id);
XMLResourceParser getXML(int id);
InputStream openRawResource(int id);.....etc
```

Configuration Class :

- ✓ This class describes all device configuration information like as input modes, screen size and screen orientation.
  - ✓ We can get this object from Resources, using getConfiguration(),from an activity, we can get it by chaining the request with getResources():

```
Resources res=getResources();
Configuration config = res.getConfiguration();
```

Fields :

1. int ORIENTATION_LANDSCAPE
2. int ORIENTATION_PORTRAIT
3. int ORIENTATION_SQUARE

....etc

Example : Get Current Screen Orientation in ANDROID

The following method does the required functionality.

```
public int getScreenOrientation() {
//orientationis.
if (getResources().getConfiguration().orientation == Configuration.ORIENTATION_PORTRAIT){
  // The following message is only displayed once.
   return 1; // Portrait Mode
}else if (getResources().getConfiguration().orientation == Configuration.ORIENTATION_LANDSCAPE) {
  // The following message is only displayed once.
   return 2;   // Landscape mode
}
return 0;
}
```

Types of Resources

- Strings, colors, arrays, dimensions. Defined in res/values directory. Using them is very useful in Localization and Internationalization.
- Images put in res/drawable directory. We can put all the images or icons we need in our application.
- Animations, defined in res/anim directory. We can define animations.
- XML, defined in res/xml directory for xml files containing our custom data.
- Layout resources, defined in res/layout for declaring the views that construct the user interface of the activities.

Resource Type	Location	Description
Colors	/res/values/any-file	Represents Color identifiers pointing to color codes. Can access using R.color.* The XML Node in the file is /resources/<color>
Strings	/res/values/any-file	Represents String resources. Can access using R.string.*. The XML Node in the file is /resources/<string>
String arrays	/res/values/any-file	Represents a resource that is an array of strings. Can access using R.array.*. The XML Node in the file is /resources/<string-array>
Dimensions	/res/values/any-file	Represents dimensions or sizes of various elements or views in Android. Supports pixels, inches, millimeter, density independent pixels, and scale independent pixels. Can access using R.dimen.*. The XML Node in the file is /resources/<dimen>
Images	/res/drawable/multiple-files	Represents image resources. Supports images include .jpg, .gif, .png etc. Can access usingChennai H O R.drawable.*
Arbitrary XML files	/res/xml/*.xml	Android allows arbitrary XML files are resources. These files will be compiled by AAPT compiler. Can access using R.xml.*
Arbitrary raw resources	/res/raw/*.*	Allows arbitrary non compiled binary or text files under this directory like .txt, audio or video files. Can access R.raw.*

Working with string resources

/res/values/string.xml

1. Single Value String values:

```
<resource>
<string name=”name”> InetSolv </string>
<string name=”shortname”>Solv </string>
</resources>
```

Accessing String values in layout files (for Ex in main.xml)

```
<TextView …..
        android:text=”@string/name”/>
</TextView>
```

2. String Array Values:

```
<resource>
        <string-array name=”test_array”>
                <item>one</item>
                <item>two</item>
                …etc
        </string-array>
</resources>
```

Accessing from java code

```
Resources res=this.getResources();
String strings[]=res.getSringArray(R.array.test_array);
for(….){ …..}
```

Working with Boolean resources

<bool> is the tag used to Boolean values in xml file, values may be true/false res/values/*bools*.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <bool name="screen_small">true</bool>
   <bool name="adjust_view_bounds">true</bool>
</resources>
```

Accessing from layout xml file (for Ex : main.xml)

```
<ImageView…… android:adjustViewBounds="@bool/adjust_view_bounds"/>
```

Accessing from Java Code :

```
Resources res = getResources();
boolean screenIsSmall = res.getBoolean(R.bool.screen_small);
```

Working with integer resources

Used to define Integer constants in xml file we can define in two ways, are

Single Value Integers :

XML file saved at res/values/integers.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <integer name="max_speed">75</dimen>
   <integer name="min_speed">5</dimen>
</resources>
```

Accessing from java code :

```
        Resources res = getResources();
 int maxSpeed = res.getInteger(R.integer.max_speed);
```

i) Integer Array :

XML file saved at res/values/integers.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <integer-array name="bits">
      <item>4</item>
      <item>8</item>
      <item>16</item>
      <item>32</item>
   </integer-array>
</resources>
```

Accessing from java code :

```
        Resources res = getResources();
  int[] bits = res.getIntArray(R.array.bits);
```

Working with color resources

res/values/color.xml

```
<resources>
        <color name=”red”>#f00</color>
        <color name=”green”>#f0f0</color>
        …etc
</resources>
```

Applying color to textview in layout file (for Ex main.xml)

```
<TextView ….
        android:textColor=”@color/red”>
</TextView>
```

Getting color in java code

```
int color=currentactivity.getResources.getColor(R.color.red);
```

Working with Dimension Resources :

Pixels, inches, and points are all examples of dimensions that can play a part in XML layouts or Java Code. The dimension resources can be defined in the following units:

- Pixels: (px).
- Inches: (in).
- Millimeters: (mm).
- Points: (pt).
- Density: (dp) density-independent pixels based on 160 dpi (dot per inch).
- Scale: (sp) Scale-independent pixels (dimensions that allow for user sizing; helpful for use in font sizes, image sizes..etc).

res/values/dimens.xml

```
<resources>
   <dimen name="icon_width">55dp</dimen>
   <dimen name="icon_height">55dp</dimen>
        <dimen name=”medium_size”>100sp</dimen>
        ..etc
```

```
</resources>
```

Accessing from layout xml file(for Ex main.xml file)

```
<TextView ……android:textSize=”@dimen/medium_size”/>
```

Getting dimension in java code :

```
txtdp.setTextSize(this.getResources().getDimension(R.dimen.medium_size));
```

Working with image resources

1. Android generates resource IDs for image files placed in the /res/drawable subdirectory. The supported image types include .gif, .jpg, and .png. Each image file in this directory generates a unique ID from its base file name. If the image file name is sample_image.jpg, for example, then the resource ID generated will be R.drawable.sample_image.
2. We have four types of drawable folders:
   - Low density screens (ldpi): 36x36px, 120dpi
   - Medium density screens (mdpi): 48x48px, 160dpi
   - High density screens (hdpi): 72x72px, 240dpi
   - Extra high density screens (xdpi): 96x96px, 320dpi
3. These folders are used to put our images in to adapt to different screen sizes. For example we may create two files of the same image, one for high density screens (hdpi) and the other with smaller resolution for less density screens (mdpi or ldpi).

Accessing images in layout xml file (for Ex main.xml):

```
<TextView ….. android:background=”@drawable/sample_image”/>
```

Accessing from Java code :

```
button.setBackgroundResource(R.drawable.icon);
```

Working with color drawable resources

We can define XML files that contain definitions for color drawable resources which are color rectangles that can be used as backgrounds.

/res/values/myresources.xml

```
<resources>
        <drawable name="red_box">#ff0000</drawable>
        <drawable name="blue_box">#0000ff</drawable>
        <drawable name="green_box">#00ff00</drawable>
</resources>
```

Java Code to Access :

```
TextView txt=(TextView)findViewById(R.id.txt);
txt.setBackgroundResource(R.drawable.redBox);          (or)
txt.setBackgroundResource(R.drawable.redBox);
```

Example on Color Resources :

Define an Android project, add a new Resource XML file in values folder, and modify the main.xml, run the project, observe the output./res/values/mycolor.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="red">#ff0000</color>
<color name="green">#00ff00</color>
```

```
<color name="blue">#0000ff</color>
</resources>
```

/res/layout/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="fill_parent"
   android:layout_height="fill_parent"
   android:id="@+id/background"    >
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Raj..Sathya "    />
<!-- "white" defined in Android base set of colors -->
<Button
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="WHITE"
   android:textColor="@android:color/white"
   android:id="@+id/whitebutton" />
<!-- direct define textColor -->
<Button
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="RED"
   android:textColor="#ff0000"
   android:id="@+id/redbutton" />
<!-- "green" defined in mycolor.xml -->
<Button
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="GREEN"
   android:textColor="@color/green"
   android:id="@+id/greenbutton" />

<!-- "blue" defined in mycolor.xml -->
<Button
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="BLUE"
   android:textColor="@color/blue"
   android:id="@+id/bluebutton" />
</LinearLayout>
```

8. User Interfaces

8.1. About XML

XML is a software- and hardware-independent tool for storing and transporting data.

XML Simplifies Things

- It simplifies data sharing
- It simplifies data transport
- It simplifies platform changes
- It simplifies data availability

Many computer systems contain data in incompatible formats. Exchanging data between incompatible systems (or upgraded systems) is a time-consuming task for web developers. Large amounts of data must be converted, and incompatible data is often lost.
XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data.
XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.
With XML, data can be available to all kinds of "reading machines" like people, computers, voice machines, news feeds, etc.

Android User Interface Elements :

- **Basic Android Debugging with Logs :**

In software engineering, debugging is a key aspect of Android development.

- **The Log Class**

Android SDK includes a useful logging utility class called android.util.Log. The class allows us to log messages

categorized based severity; each type of logging message has its own message. Here is a listing of the message types, and their respective method calls, ordered from lowest to highest priority:

- The Log.v() method is used to log verbose messages.
- The Log.d() method is used to log debug messages.
- The Log.i() method is used to log informational messages.
- The Log.w() method is used to log warnings.
- The Log.e() method is used to log errors.
- The Log.wtf() method is used to log events that should never happen ("wtf" being an abbreviation for "What a Terrible Failure", of course). We can think of this method as the equivalent of Java's assert method.

One should *always* consider a message's type when assigning log messages to one of the six method calls, as this will allow us to filter our logput output when a

ppropriate. It is also important to understand when it is acceptable to compile log messages into our application:

- ✓ Verbose logs should never be compiled into an application except during development. When development is complete and we are ready to release our application to the world, we should remove all verbose method calls either by commenting them out.
- ✓ Debug logs are compiled in but are ignored at runtime.
- ✓ Error, warning and informational logs are always kept.

- **A Simple Pattern**

A simple way to organize debugging is with the sample pattern implemented below. A global, static string is given to represent the specific class (an Activity in this example, but it could be a service, an adapter, anything), and a boolean variable to represent whether or not log messages should be printed to the logcat.

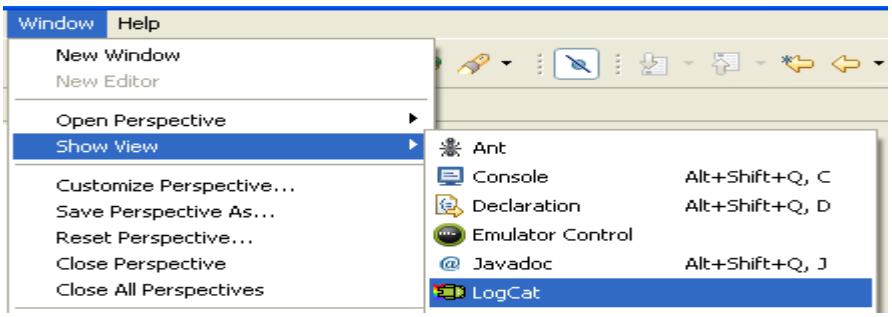
```
public class SampleActivity extends Activity {
 /**
  * A string constant to use in calls to the "log" methods. Its
  * value is often given by the name of the class, as this will
  * allow you to easily determine where log methods are coming
  * from when you analyze your logcat output.
  */
 private static final String TAG = "SampleActivity";
 /**
  * Toggle this boolean constant's value to turn on/off logging
  * within the class.
  */
  @Override
 public void onCreate(Bundle savedInstanceState) {

   super.onCreate(savedInstanceState);
   if (true)

     Log.v(TAG, "+++ ON CREATE +++");
 }
 @Override

 public void onStart() {
  super.onStart();

   if (true)
Log.v(TAG, "++ ON START ++");

 }
  @Override
 public void onResume() {
   super.onResume();
   if (true) Log.v(TAG, "+ ON RESUME +");

 }

}
```

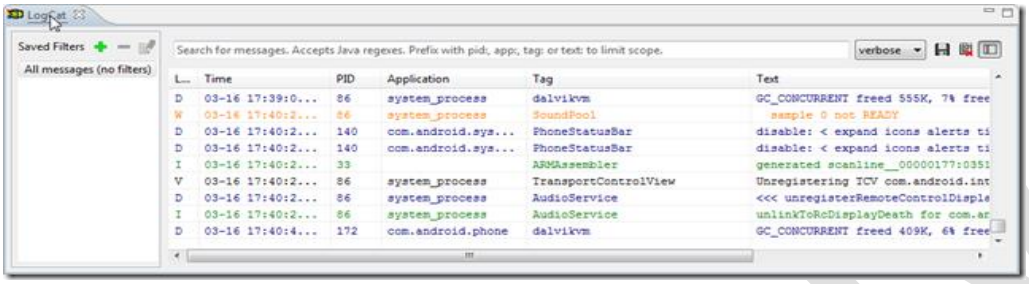
When the sample activity above is launched, the resulting logcat is presented with nicely formatted and human-readable way:

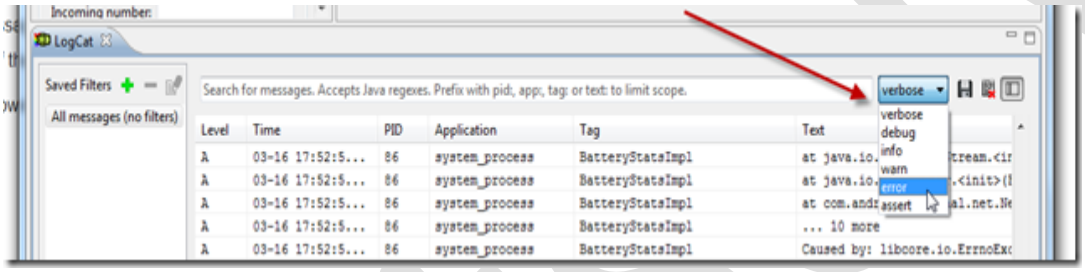
Log Output on LogCat window :

V SampleActivity +++ ON CREATE +++
V SampleActivity ++ ON START++
V SampleActivity + ON RESUME +

Accessing LogCat window :

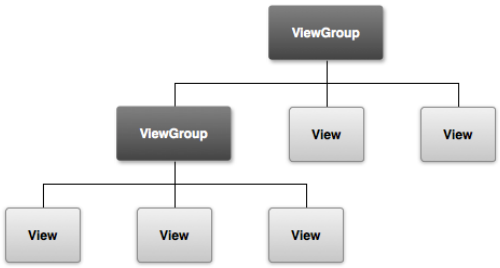






8.2. Views and Layouts

- ✓ The basic unit of the Android UI is the View. A View represents a widget that has an appearance on the screen. We have various common views we can use in Android development.
- ✓ Examples of widgets are buttons, labels, text boxes, etc. A View derives from the base class android.view.View
- ✓ One or more Views can be grouped together into a ViewGroup. A ViewGroup (which is by itself is a special type of View) provides the layout in which you can order the appearance and sequence of views. Examples of Viewgroups are LinearLayout, FrameLayout, etc. A ViewGroup derives from the base class android.view.ViewGroup



8.3. Android GUI Widgets

A Widget is Simply a view object that serves as interaction interface to the user. Android provides bunch of predefined widgets, also we can create our own,
widgets are
Form Widgets

- ✓ **TextView**

A TextView displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing.

```
<TextView
    android:id="@+id/text_id"
    android:layout_width="300dp"
    android:layout_height="200dp"
    android:capitalize="characters"
    android:text="hello_world"
    android:textColor="#ff00ff"
    android:textColorHighlight="#000"
    android:layout_centerVertical="true"
    android:layout_alignParentEnd="true"
    android:textSize="50dp"/>
```

✓ **Button**

A Button is a Push-button which can be pressed, or clicked, by the user to perform an action.

```
<Button
android:id="@+id/angry_btn"
android:text="Button"
android:textColor="#FFFFFF"
android:textSize="30sp"
android:layout_width="270dp"
android:layout_height="60dp"
android:background="#000"
android:shadowColor="#A8A8A8"
android:shadowDx="0"
android:shadowDy="0"
android:shadowRadius="5"/>
```

✓ **ToggleButton**

✓ A ToggleButton displays checked/unchecked states as a button. It is basically an on/off button with a light indicator.

✓

✓ *ToggleButton Attributes*

✓ Following are the important attributes related to ToggleButton control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Attribute	Description
android:disabledAlpha	This is the alpha to apply to the indicator when disabled.

android:textOff	This is the text for the button when it is not checked.
android:textOn	This is the text for the button when it is checked.

- ✓ Inherited from **android.widget.TextView** Class –

Attribute	Description
android:autoText	If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
android:drawableBottom	This is the drawable to be drawn below the text.
android:drawableRight	This is the drawable to be drawn to the right of the text.
android:editable	If set, specifies that this TextView has an input method.
android:text	This is the Text to display.

- ✓ Inherited from **android.view.View** Class:

Attribute	Description
android:background	This is a drawable to use as the background.
android:contentDescription	This defines text that briefly describes content of the view.
android:id	This supplies an identifier name for this view,
android:onClick	This is the name of the method in this View's context to invoke when the view is clicked.
android:visibility	This controls the initial visibility of the view.

activity_main.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"
    tools:context=".MainActivity" >

    <ToggleButton
        android:id="@+id/toggleButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="22dp"
        android:layout_marginTop="24dp"
        android:onClick="onToggleClicked"
        android:text="ToggleButton" />

    <ToggleButton
        android:id="@+id/toggleButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/toggleButton1"
        android:layout_below="@+id/toggleButton1"
        android:textOn="Turn OFF"
        android:textOff="Turn ON"
        android:layout_marginTop="20dp"
        android:text="ToggleButton" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/toggleButton1"
        android:layout_toRightOf="@+id/toggleButton1"
        android:editable="false"
        android:ems="10"/>

    <EditText
        android:id="@+id/editText2"
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/toggleButton2"
        android:layout_toRightOf="@+id/toggleButton2"
        android:editable="false"
        android:ems="10"/>

</RelativeLayout>
```

MainActivity.Java

```
package com.example.togglebutton;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
```

```
import android.view.Menu;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ToggleButton;

public class MainActivity extends Activity {
 private ToggleButton button1;
 private ToggleButton button2;
 private EditText textView1;
 private EditText textView2;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  button1 = (ToggleButton) findViewById(R.id.toggleButton1);
  button2 = (ToggleButton) findViewById(R.id.toggleButton2);
  textView1=(EditText) findViewById(R.id.editText1);
  textView2=(EditText) findViewById(R.id.editText2);
  //Setting initial values
  textView1.setText("Button1 is OFF");
  textView2.setText("Button2 is OFF");
  button2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
      public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    if (isChecked) {
     Log.i("info", "Button2 is on!");
     textView2.setText("Button2 is ON");
    } else {
     Log.i("info", "Button2 is off!");
     textView2.setText("Button2 is OFF");
    }
     }
  });
 }
 public void onToggleClicked(View view) {
  boolean on = ((ToggleButton) view).isChecked();
  if (on) {
   Log.i("info", "Button1 is on!");
   textView1.setText("Button1 is ON");
  } else {
   Log.i("info", "Button1 is off!");
   textView1.setText("Button1 is OFF");
  }
 }
}
```

- ✓ **Checkboxes**
- ✓ **Android CheckBox** is a type of two state button either checked or unchecked.
- ✓ There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.
- ✓ Android CheckBox class is the subclass of CompoundButton class.

- ✓ *Android CheckBox class*
- ✓ The android.widget.CheckBox class provides the facility of creating the CheckBoxes.

- ✓ **Methods of CheckBox class**

- ✓ There are many inherited methods of View, TextView, and Button classes in the CheckBox class. Some of them are as follows:

Method	Description
public boolean isChecked()	Returns true if it is checked otherwise false.
public void setChecked(boolean status)	Changes the state of the CheckBox.

Activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/LinearLayout1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context="${relativePackage}.${activityClass}" >

   <TextView
      android:id="@+id/textView1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello_world"
      android:gravity="center" />

   <CheckBox
      android:id="@+id/checkBox1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="47dp"
      android:text="@string/check" />

</LinearLayout>
```

MainActivity.java

```
package com.example.checkbox;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.Toast;

public class MainActivity extends Activity implements OnCheckedChangeListener {
```

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

		CheckBox ch=(CheckBox) findViewById(R.id.checkBox1);

//		ch.setOnCheckedChangeListener(new OnCheckedChangeListener() {
//
//			@Override
//		public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
//				// TODO Auto-generated method stub
//
//			}
//		});

		ch.setOnCheckedChangeListener(this);
	}

	@Override
	public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
		// TODO Auto-generated method stub

		Toast.makeText(getApplicationContext(), "checkbox", 3000).show();

	}
}
```

- ✓ **Radiogroup**
- ✓ A RadioGroup class is used for set of radio buttons.

- ✓ If we check one radio button that belongs to a radio group, it automatically unchecks any previously checked radio button within the same group.

- ✓ *RadioGroup Attributes*
- ✓ Following are the important attributes related to RadioGroup control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

Attribute	Description
android:checkedButton	This is the id of child radio button that should be checked by default within this radio group.

- ✓ Inherited from **android.view.View** Class –

Attribute	Description
android:background	This is a drawable to use as the background.
android:contentDescription	This defines text that briefly describes content of the view.
android:id	This supplies an identifier name for this view,
android:onClick	This is the name of the method in this View's context to invoke when the view is clicked.
android:visibility	This controls the initial visibility of the view.

Activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/LinearLayout1"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context="${relativePackage}.${activityClass}" >

  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"
    android:gravity="center"/>

  <RadioButton
    android:id="@+id/radioButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/male" />

  <RadioButton
    android:id="@+id/radioButton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/female" />

  <RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/rg"
    android:orientation="horizontal"
    android:background="#ccddee">

    <RadioButton
```

```
    android:id="@+id/radioButton3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/male"
    />

  <RadioButton
    android:id="@+id/radioButton4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/female" />

  <RadioButton
    android:id="@+id/radioButton5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hiii" />
  </RadioGroup>

</LinearLayout>
```

MainActivity.java

```
package com.example.radiobutton;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioGroup;
import android.widget.RadioGroup.OnCheckedChangeListener;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {

        RadioGroup rg;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                rg=(RadioGroup) findViewById(R.id.rg);
                rg.setOnCheckedChangeListener(new OnCheckedChangeListener() {

                        @Override
                        public void onCheckedChanged(RadioGroup group, int checkedId) {
                                // TODO Auto-generated method stub
                                //
                                if(checkedId==R.id.radioButton3){
                                        Toast.makeText(getApplicationContext(),
"Male",Toast.LENGTH_LONG).show();


                                }
                                else if(checkedId==R.id.radioButton4){
                                        Toast.makeText(getApplicationContext(),
```

```
"FeMale",Toast.LENGTH_LONG).show();
                                        }
                                        else
                                        {
                                                Toast.makeText(getApplicationContext(), "hii", 5000).show();
                                        }
                                }
                        });

        }

        }
```

- ✓ **Checked TextView**

CheckedTextView is an extension of normal TextView that has a checkbox along with some text. It can be useful when included in a ListView where it's setChoiceMode has been set to something other than CHOICE_MODE_NONE.

```
        <CheckedTextView
            android:id="@+id/checkedTextView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="48dp"
            android:checkMark="?android:attr/listChoiceIndicatorMultiple"
            android:checked="true"
    android:text="CheckedTextView" />

        final CheckedTextView ctv = (CheckedTextView) findViewById(R.id.checkedTextView1);
          ctv.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                        if (ctv.isChecked())
                                ctv.setChecked(false);
                        else
                                ctv.setChecked(true);

                }
  });
```

- ✓ **ProgressBar**

Progress bars are used to show progress of a task. For example, when you are uploading or downloading something from the internet, it is better to show the progress of download/upload to the user.

```
        <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"
           tools:context=".MainActivity" >

           <ProgressBar
              android:id="@+id/progressBar1"
              style="?android:attr/progressBarStyleHorizontal"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginLeft="23dp"
            android:layout_marginTop="20dp"
            android:indeterminate="false"
            android:max="100"
            android:minHeight="50dp"
            android:minWidth="200dp"
            android:progress="1" />

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/progressBar1"
            android:layout_below="@+id/progressBar1"/>

</RelativeLayout>

        package com.example.progressbar;

        import android.app.Activity;
        import android.os.Bundle;
        import android.os.Handler;
        import android.view.Menu;
        import android.widget.ProgressBar;
        import android.widget.TextView;

        public class MainActivity extends Activity {
         private ProgressBar progressBar;
         private int progressStatus = 0;
         private TextView textView;
         private Handler handler = new Handler();

         @Override
         protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);
          progressBar = (ProgressBar) findViewById(R.id.progressBar1);
          textView = (TextView) findViewById(R.id.textView1);
          // Start long running operation in a background thread
          new Thread(new Runnable() {
            public void run() {
              while (progressStatus < 100) {
                progressStatus += 1;
           // Update the progress bar and display the
                          //current value in the text view
           handler.post(new Runnable() {
           public void run() {
             progressBar.setProgress(progressStatus);
             textView.setText(progressStatus+"/"+progressBar.getMax());
```

```
            }
                });
                try {
                  // Sleep for 200 milliseconds.
                            //Just to display the progress slowly
                  Thread.sleep(200);
                } catch (InterruptedException e) {
                  e.printStackTrace();
                }
              }
            }
            }).start();
          }
}
```

- ✓ **Seekbar**

Android SeekBar is the extension of ProgressBar. SeekBar allows the user to change the value using touch event/draggable thumb/left right arrow keys. The user can increase the value by dragging the thumb right or by pressing the right arrow key. Similarly the user can decrease the value by dragging the thumb left or by pressing the left arrow key.

```
        <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"
          tools:context=".MainActivity" >

          <SeekBar
            android:id="@+id/seekBar1"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginTop="26dp"
            android:max="10"/>

          <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_below="@+id/seekBar1"
            android:layout_marginLeft="29dp"
            android:layout_marginTop="14dp" />

</RelativeLayout>
        package com.example.seekbar;

        import android.os.Bundle;
        import android.app.Activity;
        import android.view.Menu;
        import android.widget.SeekBar;
        import android.widget.SeekBar.OnSeekBarChangeListener;
        import android.widget.TextView;
```

```
public class MainActivity extends Activity {

 private SeekBar seekBar;
 private TextView textView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  seekBar = (SeekBar) findViewById(R.id.seekBar1);
  textView = (TextView) findViewById(R.id.textView1);
  // Initialize the textview with '0'
  textView.setText(seekBar.getProgress() + "/" + seekBar.getMax());
  seekBar.setOnSeekBarChangeListener(
            new OnSeekBarChangeListener() {
    int progress = 0;
      @Override
     public void onProgressChanged(SeekBar seekBar,
                                   int progresValue, boolean fromUser) {
      progress = progresValue;
     }

     @Override
     public void onStartTrackingTouch(SeekBar seekBar) {
      // Do something here,
                //if you want to do anything at the start of
      // touching the seekbar
     }

     @Override
     public void onStopTrackingTouch(SeekBar seekBar) {
      // Display the value in textview
      textView.setText(progress + "/" + seekBar.getMax());
     }
  });
 }

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

}
```

- ✓ **Quickcontact badge**

  Quick contact badge is gives us the way to add any contact information directly through android application to mobile phone. Quick contact badge is basically used in information and advice apps because with this feature application user can naively store given contact number, email without completing copying process. QuickContactBadge will save the contact using one click contact integration.

```
<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.android_examples.com.quickcontactbadge.MainActivity" >

 <TextView
 android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentTop="true"
 android:layout_centerHorizontal="true"
 android:layout_marginTop="38dp"
 android:text="Email Contact" />

 <QuickContactBadge
 android:id="@+id/quickContactBadge1"
 android:layout_width="100sp"
 android:layout_height="100sp"
 android:layout_below="@+id/textView1"
 android:layout_centerHorizontal="true"
 android:layout_marginTop="18dp"
 android:src="@drawable/b" />

 <TextView
 android:id="@+id/textView2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerHorizontal="true"
 android:layout_centerVertical="true"
 android:text="Phone Contact" />

 <QuickContactBadge
 android:id="@+id/quickContactBadge2"
 android:layout_width="100sp"
 android:layout_height="100sp"
 android:layout_below="@+id/textView2"
 android:layout_centerHorizontal="true"
 android:layout_marginTop="20dp"
 android:src="@drawable/c" />

</RelativeLayout>
```

```
package com.android_examples.com.quickcontactbadge;
import android.app.Activity;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.widget.QuickContactBadge;
import android.widget.TextView;
```

```
public class MainActivity extends Activity {

TextView Email,Phone;
QuickContactBadge EmailPic,PhonePic;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

Email = (TextView)findViewById(R.id.textView1);
Phone = (TextView)findViewById(R.id.textView2);
EmailPic = (QuickContactBadge)findViewById(R.id.quickContactBadge1);
PhonePic = (QuickContactBadge)findViewById(R.id.quickContactBadge2);

//Assign the contact badge to Email Pick badge.

EmailPic.assignContactFromEmail("android@examples.com", true);
EmailPic.setMode(ContactsContract.QuickContact.MODE_MEDIUM);

//Assign the contact badge to phone pick badge.

PhonePic.assignContactFromPhone("+911234567890", true);
PhonePic.setMode(ContactsContract.QuickContact.MODE_MEDIUM);

}
}
```

- ✓ **Rating Bar**

Android RatingBar can be used to get the rating from the user. The Rating returns a floating-point number. It may be 2.0, 3.5, 4.0 etc.
Android RatingBar displays the rating in stars.

```
<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"
   tools:context="${relativePackage}.${activityClass}" >

      <RatingBar
      android:id="@+id/ratingBar1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_below="@+id/textView1"
      android:numStars="6"
      android:stepSize="0.25"/>

</RelativeLayout>

package com.example.ratingbar;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.MenuItem;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.RatingBar;
import android.widget.RatingBar.OnRatingBarChangeListener;
import android.widget.Switch;
import android.widget.Toast;
import android.widget.ToggleButton;

public class RatingActivity extends Activity {

	RatingBar rb;

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

		rb=(RatingBar) findViewById(R.id.ratingBar1);


		rb.setOnRatingBarChangeListener(new OnRatingBarChangeListener() {

			@Override
			public void onRatingChanged(RatingBar ratingBar, float rating,
					boolean fromUser) {
				// TODO Auto-generated method stub
				Toast.makeText(getApplicationContext(), "Rating is "+rating,
Toast.LENGTH_LONG).show();

			}
		});
	}

}
```

TextFields

- ✓ **PlainText**

A EditText is an overlay over TextView that configures itself to be editable. It is the predefined subclass of TextView that includes rich editing capabilities.

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button1"
    android:layout_marginTop="69dp"
    android:layout_toRightOf="@+id/textView1"
    android:ems="10" >
  </EditText>
```

- ✓ **Personname**

```
<EditText
    android:id="@+id/editText1"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginTop="50dp"
    android:ems="10"
    android:inputType="textPersonName" >
</EditText>
```

✓ **Password**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginLeft="28dp"
    android:layout_marginTop="50dp"
    android:ems="10"
    android:inputType="textPassword" >
</EditText>
```

✓ **Password numeric**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginTop="64dp"
    android:ems="10"
    android:inputType="numberPassword" >
</EditText>
```

✓ **Email**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginLeft="52dp"
    android:layout_marginTop="74dp"
    android:ems="10"
    android:inputType="textEmailAddress" >
</EditText>
```

✓ **Phone**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginLeft="14dp"
    android:layout_marginTop="67dp"
    android:ems="10"
    android:inputType="phone" >
```

```
</EditText>
```

✓ **Postal Address**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginTop="132dp"
    android:ems="10"
    android:inputType="textPostalAddress" >

</EditText>
```

✓ **Multiline text**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_centerVertical="true"
    android:layout_marginLeft="26dp"
    android:ems="10"
    android:inputType="textMultiLine" >
</EditText>
```

✓ **Time**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginLeft="23dp"
    android:layout_marginTop="53dp"
    android:ems="10"
    android:inputType="time" >
</EditText>
```

✓ **Date**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button1"
    android:layout_marginTop="64dp"
    android:layout_toLeftOf="@+id/button1"
    android:ems="10"
    android:inputType="date" >
</EditText>
```

✓ **Number**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginLeft="14dp"
    android:layout_marginTop="78dp"
    android:ems="10"
    android:inputType="number" >
  </EditText>
```

✓ **Number Signed**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/button1"
    android:layout_marginTop="146dp"
    android:ems="10"
    android:inputType="numberSigned" >
  </EditText>
```

✓ **Number Decimal**

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_centerVertical="true"
    android:layout_marginLeft="54dp"
    android:ems="10"
    android:inputType="numberDecimal" >
  </EditText>
```

✓ **SearchView**

```
        <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" >
          <SearchView
            android:id="@+id/searchView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_alignParentTop="true"
            android:layout_marginTop="30dp" >
          </SearchView>

</RelativeLayout>

        Package raj.searchview_static;

        import android.os.Bundle;
        import android.app.Activity;
        import android.view.View;
        import android.widget.SearchView;
        import android.widget.SearchView.OnQueryTextListener;
        import android.widget.Toast;
```

```
public class MainActivity extends Activity {

        SearchView search;

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

      search=(SearchView) findViewById(R.id.searchView1);
      search.setQueryHint("SearchView");

      //*** setOnQueryTextFocusChangeListener ***
      search.setOnQueryTextFocusChangeListener(new View.OnFocusChangeListener() {

                  @Override
                  public void onFocusChange(View v, boolean hasFocus) {
                        // TODO Auto-generated method stub

                        Toast.makeText(getBaseContext(), String.valueOf(hasFocus),
                              Toast.LENGTH_SHORT).show();
                  }
         });

      //*** setOnQueryTextListener ***
      search.setOnQueryTextListener(new OnQueryTextListener() {

                  @Override
                  public boolean onQueryTextSubmit(String query) {
                        // TODO Auto-generated method stub

                        Toast.makeText(getBaseContext(), query,
                              Toast.LENGTH_SHORT).show();

                        return false;
                  }

                  @Override
                  public boolean onQueryTextChange(String newText) {
                        // TODO Auto-generated method stub

                  //    Toast.makeText(getBaseContext(), newText,
                              Toast.LENGTH_SHORT).show();
                        return false;
                  }
         });
   }

    }
```

- ✓ **Sliding Drawer**

Sliding drawer is is used to add smooth simple sliding drawer navigational menus on android applications. Sliding drawer works same as web designing sliding menus. It controlled by simple handle button automatically generated by SlidingDrawer tag. Sliding drawer always works inside layout and best work on linear layout. Sliding drawer contains handle buttons and after it another layout tag and all of hidden menus implements on there

```
<LinearLayout 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"
tools:context="com.android_examples.com.slidingdrawer.MainActivity"
android:orientation="vertical">
<SlidingDrawer
android:id="@+id/slidingDrawer1"
android:layout_width="wrap_content"
android:layout_height="350dp"
android:content="@+id/content"
android:handle="@+id/handle"
android:orientation="vertical"
android:rotation="180"  >
<Button
android:id="@+id/handle"
android:layout_width="wrap_content"
android:layout_height="50sp"
android:background="@drawable/arrow_icon"
/>
<LinearLayout
android:id="@+id/content"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:rotation="180" >
<!-- PUT HERE ANY WIDGETS OR BUTTONS, IMAGES, TEXTVIEW, EDITTEXT BOX, SEARCH BOX -->
<!-- TO OPEN INTO SLIDING DRAWER. -->
<ImageView
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="@drawable/sample_image" />
</LinearLayout>
</SlidingDrawer>
</LinearLayout>

package com.android_examples.com.slidingdrawer;
import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.app.Activity;
import android.os.Build;
import android.os.Bundle;
import android.view.DragEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnDragListener;
import android.widget.Button;
import android.widget.SlidingDrawer;
import android.widget.SlidingDrawer.OnDrawerCloseListener;
```

```
import android.widget.SlidingDrawer.OnDrawerOpenListener;
import android.widget.Toast;

@SuppressWarnings("deprecation")
public class MainActivity extends Activity {

 SlidingDrawer slidingdrawer;
 Button SlidingButton;
 @TargetApi(Build.VERSION_CODES.HONEYCOMB)
 @SuppressLint("NewApi")
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);

 slidingdrawer = (SlidingDrawer)findViewById(R.id.slidingDrawer1);
 SlidingButton = (Button)findViewById(R.id.handle);

 slidingdrawer.setOnDrawerOpenListener(new OnDrawerOpenListener() {

 @Override
 public void onDrawerOpened() {

 Toast.makeText(MainActivity.this, "Sliding drawer open", Toast.LENGTH_LONG).show();
 }
 });

 slidingdrawer.setOnDrawerCloseListener(new OnDrawerCloseListener() {

 public void onDrawerClosed() {

 Toast.makeText(MainActivity.this, "Sliding drawer close", Toast.LENGTH_LONG).show();

 }

 });
 }
}
```

- ✓ **Webview**

```
<?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" >
  <WebView
   android:id="@+id/webView1"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"/>
</LinearLayout>
package com.androidexample.webview;

import android.app.Activity;
```

```
import android.app.ProgressDialog;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class ShowWebView extends Activity {

   //private Button button;
   private WebView webView;
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.show_web_view);
      //Get webview
      webView = (WebView) findViewById(R.id.webView1);
      startWebView(http://forum.androindian.com);
   }
   private void startWebView(String url) {
      //Create new webview Client to show progress dialog
      //When opening a url or click on link
            webView.setWebViewClient(new WebViewClient() {
         ProgressDialog progressDialog;
         //If you will not use this method url links are opeen in new brower not in webview
         public boolean shouldOverrideUrlLoading(WebView view, String url) {
            view.loadUrl(url);
            return true;
         }

         //Show loader on url load
         public void onLoadResource (WebView view, String url) {
            if (progressDialog == null) {
               // in standard case YourActivity.this
               progressDialog = new ProgressDialog(ShowWebView.this);
               progressDialog.setMessage("Loading...");
               progressDialog.show();
            }
         }
         public void onPageFinished(WebView view, String url) {
            try{
            if (progressDialog.isShowing()) {
               progressDialog.dismiss();
               progressDialog = null;
            }
            }catch(Exception exception){
               exception.printStackTrace();
            }
         }
      });

       // Javascript inabled on webview
      webView.getSettings().setJavaScriptEnabled(true);

      // Other webview options
      /*
```

```
                webView.getSettings().setLoadWithOverviewMode(true);
                webView.getSettings().setUseWideViewPort(true);
                webView.setScrollBarStyle(WebView.SCROLLBARS_OUTSIDE_OVERLAY);
                webView.setScrollbarFadingEnabled(false);
                webView.getSettings().setBuiltInZoomControls(true);
                */
                /*
                 String summary = "<html><body>You scored <b>192</b> points.</body></html>";
                 webview.loadData(summary, "text/html", null);
                 */
                //Load url in webview
                webView.loadUrl(url);

            }

}
```

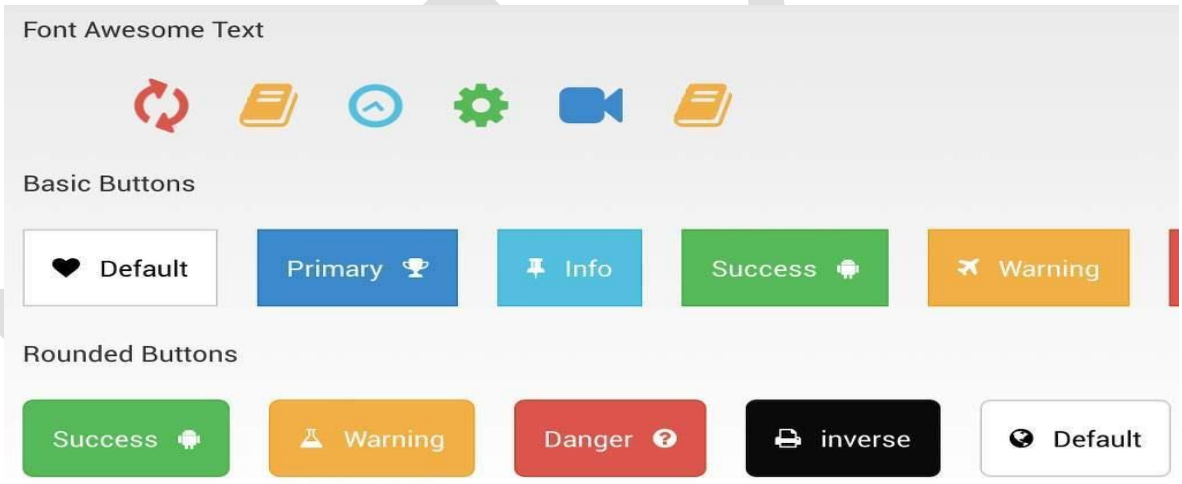
- ✓ **Imageview**

```
        <?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" >
          <ImageView
            android:id="@+id/imageView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/android" />
          <Button
            android:id="@+id/btnChangeImage"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Change Image" />

</LinearLayout>
        package com.android;

        import android.app.Activity;
        import android.os.Bundle;
        import android.widget.Button;
        import android.widget.ImageView;
        import android.view.View;
        import android.view.View.OnClickListener;

        public class MyAndroidAppActivity extends Activity {

                Button button;
                ImageView image;

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

```
                addListenerOnButton();

        }

        public void addListenerOnButton() {

                image = (ImageView) findViewById(R.id.imageView1);

                button = (Button) findViewById(R.id.btnChangeImage);
                button.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View arg0) {
                                image.setImageResource(R.drawable.android3d);
                        }

                });

        }

}
```

- ✓ **ImageButton**

A ImageButton is a AbsoluteLayout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.



- ✓
- ✓ *ImageButton Attributes*
- ✓ Following are the important attributes related to ImageButton control. You can check Android official documentation for complete list of attributes and related methods which you can use to change these attributes are run time.

- ✓ Inherited from **android.widget.ImageView** Class –

Attribute	Description

android:adjustViewBounds	Set this to true if you want the ImageView to adjust its bounds to preserve the aspect ratio of its drawable.
android:baseline	This is the offset of the baseline within this view.
android:baselineAlignBottom	If true, the image view will be baseline aligned with based on its bottom edge.
android:cropToPadding	If true, the image will be cropped to fit within its padding.
android:src	This sets a drawable as the content of this ImageView.

- ✓ Inherited from **android.view.View** Class –

Attribute	Description
android:background	This is a drawable to use as the background.
android:contentDescription	This defines text that briefly describes content of the view.
android:id	This supplies an identifier name for this view,
android:onClick	This is the name of the method in this View's context to invoke when the view is clicked.
android:visibility	This controls the initial visibility of the view.

```
<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:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">
  <TextView android:text="Tutorials Point"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30dp"
    android:layout_alignParentTop="true"
    android:layout_alignRight="@+id/imageButton"
    android:layout_alignEnd="@+id/imageButton" />
```

```
    <ImageButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageButton"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true"
      android:src="@drawable/abc"/>
</RelativeLayout>

package com.example.myapplication;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
import android.widget.Toast;

public class MainActivity extends Activity {
  @Override
  ImageButton imgButton;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    imgButton =(ImageButton)findViewById(R.id.imageButton);
    imgButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Toast.makeText(getApplicationContext(),"You download is
resumed",Toast.LENGTH_LONG).show();
      }
    });
  }
  }
```

- ✓ **Video View**

```
<?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" >
  <TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/hello"  />
  <Button
   android:id="@+id/playvideoplayer"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
```

```
            android:text="- PLAY Video -"  />

            <VideoView
             android:id="@+id/videoview"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"    />

</LinearLayout>

        package com.android.AndroidVideoPlayer;

        import android.app.Activity;
        import android.graphics.PixelFormat;
        import android.media.MediaPlayer;
        import android.net.Uri;
        import android.os.Bundle;
        import android.view.SurfaceHolder;
        import android.view.SurfaceView;
        import android.view.View;
        import android.widget.Button;
        import android.widget.VideoView;

        //Implement SurfaceHolder interface to Play video
        //Implement this interface to receive information about changes to the surface
        public class AndroidVideoPlayer extends Activity implements SurfaceHolder.Callback{

           MediaPlayer mediaPlayer;
           SurfaceView surfaceView;
           SurfaceHolder surfaceHolder;
           boolean pausing = false;;


           /** Called when the activity is first created. */
           @Override
           public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.main);

              Button buttonPlayVideo = (Button)findViewById(R.id.playvideoplayer);

              getWindow().setFormat(PixelFormat.UNKNOWN);

              //Displays a video file.
              VideoView mVideoView = (VideoView)findViewById(R.id.videoview);


              String uriPath = "android.resource://com.android.AndroidVideoPlayer/"+R.raw.k;
              Uri uri = Uri.parse(uriPath);
              mVideoView.setVideoURI(uri);
              mVideoView.requestFocus();
              mVideoView.start();
```

```
        buttonPlayVideo.setOnClickListener(new Button.OnClickListener(){

            @Override
            public void onClick(View v) {

                // VideoView refference see main.xml
                VideoView mVideoView = (VideoView)findViewById(R.id.videoview);

                String uriPath = "android.resource://com.android.AndroidVideoPlayer/"+R.raw.k;

                Uri uri = Uri.parse(uriPath);
                mVideoView.setVideoURI(uri);
                mVideoView.requestFocus();
                mVideoView.start();

            }});
        }
        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,
                int height) {
            // TODO Auto-generated method stub

        }
        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            // TODO Auto-generated method stub
        }
         @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            // TODO Auto-generated method stub

        }
}
```

✓ **Date picker**

```
 <?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/btnChangeDate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change Date" />

    <TextView
        android:id="@+id/lblDate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Current Date (M-D-YYYY): "
```

```
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/tvDate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <DatePicker
        android:id="@+id/dpResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

```
package androindian.android;

import java.util.Calendar;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;

public class MyAndroidAppActivity extends Activity {

        private TextView tvDisplayDate;
        private DatePicker dpResult;
        private Button btnChangeDate;

        private int year;
        private int month;
        private int day;

        static final int DATE_DIALOG_ID = 999;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                setCurrentDateOnView();
                addListenerOnButton();

        }

        // display current date
        public void setCurrentDateOnView() {
```

```
        tvDisplayDate = (TextView) findViewById(R.id.tvDate);
        dpResult = (DatePicker) findViewById(R.id.dpResult);

        final Calendar c = Calendar.getInstance();
        year = c.get(Calendar.YEAR);
        month = c.get(Calendar.MONTH);
        day = c.get(Calendar.DAY_OF_MONTH);

        // set current date into textview
        tvDisplayDate.setText(new StringBuilder()
                // Month is 0 based, just add 1
                .append(month + 1).append("-").append(day).append("-")
                .append(year).append(" "));

        // set current date into datepicker
        dpResult.init(year, month, day, null);

    }

    public void addListenerOnButton() {

        btnChangeDate = (Button) findViewById(R.id.btnChangeDate);

        btnChangeDate.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {

                        showDialog(DATE_DIALOG_ID);

                }

        });

    }

    @Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
        case DATE_DIALOG_ID:
          // set date picker as current date
          return new DatePickerDialog(this, datePickerListener,
       year, month,day);
        }
        return null;
    }

    private DatePickerDialog.OnDateSetListener datePickerListener
     = new DatePickerDialog.OnDateSetListener() {

        // when dialog box is closed, below method will be called.
        public void onDateSet(DatePicker view, int selectedYear,
```

```
                                        int selectedMonth, int selectedDay) {
                                year = selectedYear;
                                month = selectedMonth;
                                day = selectedDay;

                                // set selected date into textview
                                tvDisplayDate.setText(new StringBuilder().append(month + 1)
                                  .append("-").append(day).append("-").append(year)
                                  .append(" "));

                                // set selected date into datepicker also
                                dpResult.init(year, month, day, null);

                            }
                        };

}
```

✓ **Time picker**

```
<?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/btnChangeTime"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Change Time" />

  <TextView
    android:id="@+id/lblTime"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Current Time (H:M): "
    android:textAppearance="?android:attr/textAppearanceLarge" />

  <TextView
    android:id="@+id/tvTime"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceLarge" />

  <TimePicker
    android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>

package androindian.android;
```

```
import java.util.Calendar;
import android.app.Activity;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.TimePicker;

public class MyAndroidAppActivity extends Activity {

	private TextView tvDisplayTime;
	private TimePicker timePicker1;
	private Button btnChangeTime;

	private int hour;
	private int minute;

	static final int TIME_DIALOG_ID = 999;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		setCurrentTimeOnView();
		addListenerOnButton();

	}

	// display current time
	public void setCurrentTimeOnView() {

		tvDisplayTime = (TextView) findViewById(R.id.tvTime);
		timePicker1 = (TimePicker) findViewById(R.id.timePicker1);

		final Calendar c = Calendar.getInstance();
		hour = c.get(Calendar.HOUR_OF_DAY);
		minute = c.get(Calendar.MINUTE);

		// set current time into textview
		tvDisplayTime.setText(
	    new StringBuilder().append(pad(hour))
				.append(":").append(pad(minute)));

		// set current time into timepicker
		timePicker1.setCurrentHour(hour);
		timePicker1.setCurrentMinute(minute);

	}
```

```
public void addListenerOnButton() {

	btnChangeTime = (Button) findViewById(R.id.btnChangeTime);

	btnChangeTime.setOnClickListener(new OnClickListener() {

		@Override
		public void onClick(View v) {

			showDialog(TIME_DIALOG_ID);

		}

	});

}

@Override
protected Dialog onCreateDialog(int id) {
	switch (id) {
	case TIME_DIALOG_ID:
		// set time picker as current time
		return new TimePickerDialog(this,
		timePickerListener, hour, minute,false);

	}
	return null;
}

private TimePickerDialog.OnTimeSetListener timePickerListener =
new TimePickerDialog.OnTimeSetListener() {
	public void onTimeSet(TimePicker view, int selectedHour,
				int selectedMinute) {
		hour = selectedHour;
		minute = selectedMinute;

		// set current time into textview
		tvDisplayTime.setText(new StringBuilder().append(pad(hour))
					.append(":").append(pad(minute)));

		// set current time into timepicker
		timePicker1.setCurrentHour(hour);
		timePicker1.setCurrentMinute(minute);

	}
};

private static String pad(int c) {
	if (c >= 10)
	  return String.valueOf(c);
	else
	  return "0" + String.valueOf(c);
```

```
                }
}
```

- ✓ **CalenderView**

```
<CalendarView
    android:id="@+id/calendView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="60dp" />

public void onSelectedDayChange(CalendarView view, int year,
    int month, int dayOfMonth) {
   // TODO Auto-generated method stub

    Toast.makeText(
     getBaseContext(),
     "Selected Date is\n\n" + dayOfMonth + " / " + month
      + " / " + year, Toast.LENGTH_LONG).show();
 }
```

- ✓ **Chronometer**

```
<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" >

  <TextView
    android:id="@+id/textView1"
    android:textColor="#FF0000"
    android:textSize="20dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"
    android:text="Static Chronometer" />

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="10dp"
    android:text="Start" />

  <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button1"
    android:layout_centerHorizontal="true"
```

```
            android:layout_marginTop="10dp"
            android:text="Stop" />

        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/button2"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="10dp"
            android:text="Restart" />

        <Button
            android:id="@+id/button4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/button3"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="10dp"
            android:text="Set Format" />

        <Button
            android:id="@+id/button5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/button4"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="10dp"
            android:text="Clear Format" />

        <Chronometer
            android:id="@+id/chronometer1"
            android:textColor="#4169E1"
            android:textSize="20dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/button5"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="50dp"
            android:text="Chronometer" />

</RelativeLayout>
```

```
package balaji.chronometer_static;

import android.os.Bundle;
import android.os.SystemClock;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Chronometer;
```

```
public class MainActivity extends Activity {

        Chronometer focus;
        Button start, stop, restart, set, clear;

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

      start = (Button) findViewById(R.id.button1);
      stop = (Button) findViewById(R.id.button2);
      restart = (Button) findViewById(R.id.button3);
      set = (Button) findViewById(R.id.button4);
      clear = (Button) findViewById(R.id.button5);

      focus = (Chronometer) findViewById(R.id.chronometer1);

      start.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub

                        focus.start();
                }
        });


      stop.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub

                        focus.stop();
                }
        });


      restart.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub

                        focus.setBase(SystemClock.elapsedRealtime());
                }
        });


      set.setOnClickListener(new View.OnClickListener() {
```

```
                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub

                        focus.setFormat("Formated Time - %s");
                }
        });


    clear.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub

                        focus.setFormat(null);
                }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_main, menu);
      return true;
    }
}

Strings.xml
    <resources>

      <string name="app_name">Chronometer_Static</string>
      <string name="hello_world">Hello world!</string>
      <string name="menu_settings">Settings</string>
      <string name="title_activity_main">MainActivity</string>

</resources>
```

- ✓ **Analog Clock**

```
    <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" >

      <TextView
        android:id="@+id/textView1"
        android:textSize="20dp"
        android:textColor="#FF0000"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="60dp"
        android:text="Static Analog Clock" />
```

```
    <AnalogClock
      android:id="@+id/analogClock1"
      android:dial="@drawable/clock"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true" />

</RelativeLayout>
```

```
package balaji.analogclock_static;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.AnalogClock;
import android.widget.Toast;

public class MainActivity extends Activity {

        AnalogClock clk;

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

      clk = (AnalogClock) findViewById(R.id.analogClock1);

      clk.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub

                        Toast.makeText(getBaseContext(), "This is Static AnalogClock",
                                Toast.LENGTH_SHORT).show();
                }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_main, menu);
      return true;
    }
}
```

```
<resources>

    <string name="app_name">AnalogClock_Static</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
```

```
    <string name="title_activity_main">MainActivity</string>

</resources>
```

✓ **DigitalClock**

```
<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" >

  <TextView
    android:id="@+id/textView1"
    android:textSize="20dp"
    android:textColor="#FF0000"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="60dp"
    android:text="Static DigitalClock" />

  <DigitalClock
    android:id="@+id/digitalClock1"
    android:textSize="20dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"  />

</RelativeLayout>
```

```
package balaji.digitalclock_static;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.DigitalClock;
import android.widget.Toast;

public class MainActivity extends Activity {

        DigitalClock clk;

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

    clk = (DigitalClock) findViewById(R.id.digitalClock1);

    clk.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
```

```
                                        // TODO Auto-generated method stub

                                        Toast.makeText(getBaseContext(), clk.getText().toString(),
                                                Toast.LENGTH_SHORT).show();
                                }
                        });
                }

                @Override
                public boolean onCreateOptionsMenu(Menu menu) {
                   getMenuInflater().inflate(R.menu.activity_main, menu);
                   return true;
                }
}

        <resources>

           <string name="app_name">DigitalClock_Static</string>
           <string name="hello_world">Hello world!</string>
           <string name="menu_settings">Settings</string>
           <string name="title_activity_main">MainActivity</string>

</resources>
```

- ✓ **ImageSwitcher**

```
        <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" >

           <Gallery
              android:id="@+id/gallery1"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_centerHorizontal="true" />

           <ImageSwitcher
              android:id="@+id/imageSwitcher1"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:layout_below="@+id/gallery1"
              android:layout_centerHorizontal="true"
              android:layout_marginTop="5dp" >
           </ImageSwitcher>

</RelativeLayout>
        package balaji.imageswitcher_static;

        import android.app.Activity;
        import android.content.Context;
        import android.os.Bundle;
        import android.view.View;
        import android.view.ViewGroup;
```

```
import android.view.ViewGroup.LayoutParams;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ViewSwitcher.ViewFactory;

public class MainActivity extends Activity implements ViewFactory {

	int imgs[] =
		{
			R.drawable.android,
			R.drawable.cup,
			R.drawable.donut,
			R.drawable.eclair,
			R.drawable.froyo,
			R.drawable.ginger,
			R.drawable.honey,
			R.drawable.ics,
			R.drawable.jellybean
		};

	ImageSwitcher imgSwitcher;

	/** Called when the activity is first created. */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		imgSwitcher = (ImageSwitcher) findViewById(R.id.imageSwitcher1);
		imgSwitcher.setFactory(this);
		imgSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
				android.R.anim.fade_in));
		imgSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
				android.R.anim.fade_out));

		Gallery gallery = (Gallery) findViewById(R.id.gallery1);
		gallery.setAdapter(new ImageAdapter(this));

		gallery.setOnItemClickListener(new OnItemClickListener() {

			public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
					long arg3) {
				imgSwitcher.setImageResource(imgs[arg2]);
			}
		});
	}

	public class ImageAdapter extends BaseAdapter {
```

```
                private Context ctx;

                public ImageAdapter(Context c) {
                        ctx = c;
                }

                public int getCount() {

                        return imgs.length;
                }

                public Object getItem(int arg0) {

                        return arg0;
                }

                public long getItemId(int arg0) {

                        return arg0;
                }

                public View getView(int arg0, View arg1, ViewGroup arg2) {

                        ImageView iView = new ImageView(ctx);
                        iView.setImageResource(imgs[arg0]);
                        iView.setScaleType(ImageView.ScaleType.FIT_XY);
                        iView.setLayoutParams(new Gallery.LayoutParams(200, 150));
                        return iView;
                }
        }

        public View makeView() {
                ImageView iView = new ImageView(this);
                iView.setScaleType(ImageView.ScaleType.FIT_CENTER);
                iView.setLayoutParams(new ImageSwitcher.LayoutParams
                        (LayoutParams.MATCH_PARENT,LayoutParams.MATCH_PARENT));
                iView.setBackgroundColor(0xFF000000);
                return iView;
        }
}
```

```
    <resources>

        <string name="app_name">ImageSwitcher_Static</string>
        <string name="hello_world">Hello world!</string>
        <string name="menu_settings">Settings</string>
        <string name="title_activity_main">MainActivity</string>

</resources>
```

- ✓ **StackView**

```
<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" >

    <StackView
        android:id="@+id/stackView1"
        android:animateLayoutChanges="true"
        android:layout_width="fill_parent"
            android:layout_height="fill_parent" >

    </StackView>

</RelativeLayout>
```

Step 3 : Open res -> layout -> item.xml and add following code :

```
<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" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="35dp"
        android:text="TextView" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="61dp"
        android:src="@drawable/ic_launcher" />

</RelativeLayout>
```

Step 4 : Open src -> package -> MainActivity.java and add following code :

```
package balaji.stackview_static;

import java.util.ArrayList;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.StackView;

public class MainActivity extends Activity {

    @Override
```

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

      StackView stk = (StackView)this.findViewById(R.id.stackView1);

      ArrayList<StackItem> items = new ArrayList<StackItem>();
      items.add(new StackItem("text1", this.getResources().getDrawable(R.drawable.ic_launcher)));
      items.add(new StackItem("text2", this.getResources().getDrawable(R.drawable.ic_launcher)));
      items.add(new StackItem("text3", this.getResources().getDrawable(R.drawable.ic_launcher)));
      items.add(new StackItem("text4", this.getResources().getDrawable(R.drawable.ic_launcher)));
      items.add(new StackItem("text5", this.getResources().getDrawable(R.drawable.ic_launcher)));

      StackAdapter adapt = new StackAdapter(this, R.layout.item, items);

      stk.setAdapter(adapt);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_main, menu);
      return true;
    }
}
```

Step 5 : Open src -> package -> StackItem.java and add following code :

```
package balaji.stackview_static;

import android.graphics.drawable.Drawable;

public class StackItem {

        public String text;
        public Drawable img;

        public StackItem(String text,Drawable photo)
        {
                this.img = photo;
                this.text = text;
        }
}
```

Step 6 : Open src -> package -> StackAdapter.java and add following code :

```
package balaji.stackview_static;

import java.util.ArrayList;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
```

```
import android.widget.ImageView;
import android.widget.TextView;

public class StackAdapter extends ArrayAdapter<StackItem> {

	private ArrayList<StackItem> items;
	private Context ctx;

	public StackAdapter(Context context, int textViewResourceId,
			ArrayList<StackItem> objects) {
		super(context, textViewResourceId, objects);

		this.items = objects;
		this.ctx = context;
	}

	@Override
	public View getView(int position, View convertView, ViewGroup parent) {
		View v = convertView;
		if (v == null) {
			LayoutInflater vi = (LayoutInflater)
				ctx.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
			v = vi.inflate(R.layout.item, null);
		}

		StackItem m = items.get(position);

		if (m != null) {
			TextView text = (TextView) v.findViewById(R.id.textView1);
			ImageView img = (ImageView)v.findViewById(R.id.imageView1);

			if (text != null) {
				text.setText(m.text);
				img.setImageDrawable(m.img);
			}
		}
		return v;
	}
}
```

Step 7 : Open AndroidManifest.xml and add following code :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="balaji.stackview_static"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="11"
      android:targetSdkVersion="15" />

   <application
      android:icon="@drawable/ic_launcher"
```

```
            android:label="@string/app_name"
            android:theme="@style/AppTheme" >
            <activity
              android:name=".MainActivity"
              android:label="@string/title_activity_main" >
              <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

        </manifest>

        Step 8 : Open res ->values ->strings.xml and add following code :

        <resources>

          <string name="app_name">StackView_Static</string>
          <string name="hello_world">Hello world!</string>
          <string name="menu_settings">Settings</string>
          <string name="title_activity_main">MainActivity</string>

</resources>
```

- ✓ **TextSwitcher**

```
        <?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:background="#1E90FF" >

          <Button android:id="@+id/Add"
            android:textColor="#FFFFFF"
            android:layout_centerHorizontal="true"
            android:layout_alignParentTop="true"
            android:layout_marginTop="60dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Addition" />

          <Button android:id="@+id/Subtract"
            android:textColor="#FFFFFF"
            android:layout_centerHorizontal="true"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="60dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Subtraction" />

          <Button android:id="@+id/Multiple"
            android:textColor="#FFFFFF"
```

```
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="0dp"
        android:rotation="270"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Multiplication" />

    <Button android:id="@+id/Divide"
        android:textColor="#FFFFFF"
        android:layout_centerVertical="true"
        android:layout_alignParentRight="true"
        android:layout_marginRight="10dp"
        android:rotation="90"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Division" />

    <TextSwitcher android:id="@+id/switcher"
        android:textColor="#FFFFFF"
        android:layout_centerInParent="true"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Step 3 : Open src -> package -> MainActivity.java and add following code :

```
package balaji.textswitcher_creation;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.TextSwitcher;
import android.widget.TextView;
import android.widget.ViewSwitcher;

public class MainActivity extends Activity implements ViewSwitcher.ViewFactory,
        View.OnClickListener {

    TextSwitcher switcher;
    float counter = 0;
    Button Add, Subtract, Multiple, Divide;

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

```
switcher = (TextSwitcher) findViewById(R.id.switcher);
switcher.setFactory(MainActivity.this);

Animation in = AnimationUtils.loadAnimation(this, android.R.anim.fade_in);
Animation out = AnimationUtils.loadAnimation(this, android.R.anim.fade_out);

switcher.setInAnimation(in);
switcher.setOutAnimation(out);

Add = (Button) findViewById(R.id.Add);
Subtract = (Button) findViewById(R.id.Subtract);
Multiple = (Button) findViewById(R.id.Multiple);
Divide = (Button) findViewById(R.id.Divide);

Add.setOnClickListener(new View.OnClickListener() {

        public void onClick(View v) {
                // TODO Auto-generated method stub

                counter++;
                switcher.setText(String.valueOf(counter));
        }
    });

Subtract.setOnClickListener(new View.OnClickListener() {

        public void onClick(View v) {
                // TODO Auto-generated method stub

                counter--;
                switcher.setText(String.valueOf(counter));
        }
    });

Multiple.setOnClickListener(new View.OnClickListener() {

        public void onClick(View v) {
                // TODO Auto-generated method stub

                counter = counter*2;
                switcher.setText(String.valueOf(counter));
        }
    });

Divide.setOnClickListener(new View.OnClickListener() {

        public void onClick(View v) {
                // TODO Auto-generated method stub

                counter = counter/2;
                switcher.setText(String.valueOf(counter));
        }
    });
```

```
    }

    public View makeView() {
      TextView t = new TextView(this);
      t.setGravity(Gravity.TOP | Gravity.CENTER_HORIZONTAL);
      t.setTextSize(36);
      return t;
    }

    public void onClick(View v) {
          // TODO Auto-generated method stub
    }
}
```

Step 4 : Open AndroidManifest.xml and add following code :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="balaji.textswitcher_creation"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="3"
        android:targetSdkVersion="15" />

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

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

</manifest>
```

Step 5 : Open res ->values ->strings.xml and add following code :

```
<resources>

    <string name="app_name">TextSwitcher_Creation</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>

</resources>
```

- ✓ **ViewAnimator**

```
<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" >

  <ViewAnimator
    android:id="@+id/viewAnimator1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="100dp"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="100dp"
        android:textSize="20dp"
        android:text="TextView" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
          android:layout_gravity="center_horizontal"
          android:layout_marginTop="100dp"
          android:text="Button" />

  </ViewAnimator>

</RelativeLayout>
```

Step 3 : Open src -> package -> MainActivity.java and add following code :

```
package balaji.viewanimator_static;

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.widget.ViewAnimator;

public class MainActivity extends Activity {
```

```
ViewAnimator mContainer;

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

   initContainer();
}

private void initContainer() {
        mContainer = (ViewAnimator)findViewById(R.id.viewAnimator1);

        Animation inAnim = new AlphaAnimation(0, 1);
        inAnim.setDuration(1000);
        Animation outAnim = new AlphaAnimation(1, 0);
        outAnim.setDuration(1000);

        mContainer.setInAnimation(inAnim);
        mContainer.setOutAnimation(outAnim);
}

@Override
public boolean onTouchEvent(MotionEvent event) {
        if(event.getAction() == MotionEvent.ACTION_UP) {
                mContainer.showNext();
        }
        return true;
}
}
```

- ✓ **ViewFlipper**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/LinearLayout01"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <RelativeLayout
      android:id="@+id/RelativeLayout02"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content">
      <ViewFlipper
         android:id="@+id/ViewFlipper01"
         android:layout_width="fill_parent"
            android:layout_height="400dp" >
         <RelativeLayout
            android:layout_width="fill_parent"
                  android:layout_height="fill_parent"
            android:background="#4B0082" >
            <TextView
```

```
            android:id="@+id/TextView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#FFFFFF"
            android:textSize="20dp"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="100dp"
            android:text="Flipper Content 1" >
        </TextView>
    </RelativeLayout>
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#7CFC00"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/TextView02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#000000"
            android:textSize="20dp"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="100dp"
            android:text="Flipper Content 2" >
        </TextView>
    </RelativeLayout>
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#1E90FF"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/TextView03"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#FFFFFF"
            android:textSize="20dp"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="100dp"
            android:text="Flipper Content 3" >
        </TextView>

    </RelativeLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFFF00"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/TextView04"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#000000"
                android:textSize="20dp"
                android:layout_centerHorizontal="true"
                android:layout_marginTop="100dp"
                android:text="Flipper Content 4" >
            </TextView>
             </RelativeLayout>
    </ViewFlipper>
  </RelativeLayout>
  <RelativeLayout
    android:id="@+id/RelativeLayout03"
    android:layout_below="@+id/RelativeLayout02"
    android:background="#000000"
    android:layout_width="fill_parent"
    android:layout_height="match_parent">
    <Button
      android:id="@+id/Previous"
      android:layout_alignParentBottom="true"
      android:layout_alignParentLeft="true"
      android:layout_marginBottom="5dp"
      android:layout_marginLeft="20dp"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Previous" >
    </Button>
    <Button
      android:id="@+id/Next"
      android:layout_alignParentBottom="true"
      android:layout_alignParentRight="true"
      android:layout_marginBottom="5dp"
      android:layout_marginRight="20dp"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Next" >
    </Button>

  </RelativeLayout>
</LinearLayout>
```

Step 3 : Open src -> package -> MainActivity.java and add following code :

```
package balaji.viewflipper_creation;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.ViewFlipper;

public class MainActivity extends Activity {

        ViewFlipper viewFlipper;
```

```
        Button Next, Previous;

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

      viewFlipper = (ViewFlipper) findViewById(R.id.ViewFlipper01);

      Next = (Button) findViewById(R.id.Next);
      Previous = (Button) findViewById(R.id.Previous);

      Next.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
                  // TODO Auto-generated method stub

                  viewFlipper.showNext();
            }
        });

      Previous.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
                  // TODO Auto-generated method stub

                  viewFlipper.showPrevious();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_main, menu);
      return true;
    }
}
```

Step 4 : Open AndroidManifest.xml and add following code :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="balaji.viewflipper_creation"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
      android:minSdkVersion="3"
      android:targetSdkVersion="15" />

    <application
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
```

```
            <activity
                android:name=".MainActivity"
                android:label="@string/title_activity_main" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />

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

    </manifest>

    Step 5 : Open res ->values ->strings.xml and add following code :

    <resources>

        <string name="app_name">ViewFlipper_Creation</string>
        <string name="hello_world">Hello world!</string>
        <string name="menu_settings">Settings</string>
        <string name="title_activity_main">MainActivity</string>

</resources>
```

✓ **ViewSwitcher**

```
    <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" >

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:autoLink="web"
            android:text="http://android-er.blogspot.com/"
            android:textStyle="bold" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <Button
                android:id="@+id/prev"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
```

```
            android:layout_weight="1"
            android:text="previous" />

        <Button
            android:id="@+id/next"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="next" />
    </LinearLayout>

    <ViewSwitcher
        android:id="@+id/viewswitcher"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_launcher" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:orientation="vertical" >

            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="- Button 2 -" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="LinearLayout 2" />
        </LinearLayout>
    </ViewSwitcher>

</LinearLayout>

package com.example.androidviewswitcher;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ViewSwitcher;

public class MainActivity extends Activity {
```

```
Button buttonPrev, buttonNext;
ViewSwitcher viewSwitcher;

Animation slide_in_left, slide_out_right;

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

 buttonPrev = (Button) findViewById(R.id.prev);
 buttonNext = (Button) findViewById(R.id.next);
 viewSwitcher = (ViewSwitcher) findViewById(R.id.viewswitcher);

 slide_in_left = AnimationUtils.loadAnimation(this,
  android.R.anim.slide_in_left);
 slide_out_right = AnimationUtils.loadAnimation(this,
  android.R.anim.slide_out_right);

 viewSwitcher.setInAnimation(slide_in_left);
 viewSwitcher.setOutAnimation(slide_out_right);

 buttonPrev.setOnClickListener(new OnClickListener() {

  @Override
  public void onClick(View arg0) {
   viewSwitcher.showPrevious();
  }
 });

 buttonNext.setOnClickListener(new OnClickListener() {

  @Override
  public void onClick(View arg0) {
   viewSwitcher.showNext();
  }
 });
 ;
}

}
```

- ✓ **RequestFocus**

```
<EditText
    android:id="@+id/editText1"
    android:hint="EditText 1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="60dp"
    android:ems="10"
    android:inputType="textPersonName" >
```

```
            <requestFocus />
</EditText>
```

- ✓ **ViewStub**

```
<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" >

  <TextView
    android:id="@+id/textView1"
    android:textSize="20dp"
    android:textColor="#ff0000"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:text="Static ViewStub" />

  <ViewStub
    android:id="@+id/viewStub1"
    android:layout="@layout/subtree"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="100dp" />


</RelativeLayout>

Step 3 : Open res -> layout -> subtree.xml and add following code :

<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" >

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:text="Button" />

  <ToggleButton
    android:id="@+id/toggleButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_below="@+id/button1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:text="ToggleButton" />

  <CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/toggleButton1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:text="CheckBox" />


</RelativeLayout>
```

Step 4 : Open src -> package -> MainActivity.java and add following code :

```
package balaji.viewstub_static;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.ViewStub;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

        ViewStub stub;
        Button b;

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

    stub = (ViewStub) findViewById(R.id.viewStub1);
    View inflated = stub.inflate();

    b = (Button) findViewById(R.id.button1);

    b.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub

                        Toast.makeText(getBaseContext(), "View Stub Content Created !!!",
                                Toast.LENGTH_LONG).show();
                }
```

```
            });
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.activity_main, menu);
            return true;
        }
    }
```

Step 5 : Open AndroidManifest.xml and add following code :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="balaji.viewstub_static"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="3"
        android:targetSdkVersion="15" />

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

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

</manifest>
```

Step 6 : Open res ->values ->strings.xml and add following code :

```
<resources>

    <string name="app_name">ViewStub_Static</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>

</resources>
```

- ✓ **GestureOverlayView**

activity_main.xml (or) main.xml and add following code :

```
<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" >

   <TextView
      android:id="@+id/textView1"
      android:textSize="18dp"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="30dp"
      android:text="Static TextureView" />

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/textView1"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="30dp"
      android:text="Create" />


</RelativeLayout>
```

Step 3 : Open res -> layout -> textureview.xml and add following code :

```
<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" >

   <TextureView
      android:id="@+id/textureView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true" />

</RelativeLayout>
```

Step 4 : Open src -> package -> MainActivity.java and add following code :

```
package raj.textureview_static;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
```

```
public class MainActivity extends Activity {

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

      b = (Button) findViewById(R.id.button1);

      b.setOnClickListener(new View.OnClickListener() {

                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub

                        startActivity(new Intent(MainActivity.this, Texture.class));
                }
        });
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_main, menu);
      return true;
   }
}
```

Step 5 : Open src -> package -> Texture.java and add following code :

```
package raj.textureview_static;

import java.io.IOException;
import android.os.Bundle;
import android.app.Activity;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.view.Gravity;
import android.view.Menu;
import android.view.TextureView;
import android.widget.FrameLayout;
import android.widget.Toast;

public class Texture extends Activity implements TextureView.SurfaceTextureListener {

        TextureView texture;
        Camera c;

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

```
    texture = (TextureView) findViewById(R.id.textureView1);

    Toast.makeText(getBaseContext(), "TextureView Created", Toast.LENGTH_SHORT).show();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_main, menu);
    return true;
}

@Override
public void onSurfaceTextureAvailable(SurfaceTexture surface, int width,
            int height) {
        // TODO Auto-generated method stub

        c = Camera.open();
        Camera.Size size = c.getParameters().getPreviewSize();

        texture.setLayoutParams(new FrameLayout.LayoutParams(
                    size.width, size.height, Gravity.CENTER));

        try {
            c.setPreviewTexture(surface);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        c.startPreview();

        texture.setAlpha(0.5f);
        texture.setRotation(45.0f);
}

@Override
public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {
        // TODO Auto-generated method stub

        c.stopPreview();
        c.release();

        return true;
}

@Override
public void onSurfaceTextureSizeChanged(SurfaceTexture surface, int width,
            int height) {
        // TODO Auto-generated method stub

}
```

```
    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture surface) {
        // TODO Auto-generated method stub

    }
}
```

Step 6 : Open AndroidManifest.xml and add following code :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="raj.textureview_static"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="15" />

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

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

        <activity android:name=".Texture"> </activity>

</application>
```

✓ **TextureView**

```
<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=".MainActivity" >

    <TextureView
        android:id="@+id/tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
```

```
</RelativeLayout>
```

MainActivity.java

```
import android.annotation.SuppressLint;
import android.app.Activity;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.os.Bundle;
import android.view.Gravity;
import android.view.Menu;
import android.view.TextureView;
import android.view.TextureView.SurfaceTextureListener;
import android.widget.FrameLayout;
import java.io.IOException;

@SuppressLint("NewApi")
public class MainActivity extends Activity implements SurfaceTextureListener {
 private TextureView mTexture;
 private Camera mCamera;

  @SuppressLint("NewApi")
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  mTexture = new TextureView(this);
  mTexture.setSurfaceTextureListener(this);
  setContentView(mTexture);
 }

  @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;
 }

  @SuppressLint("NewApi")
 @Override
 public void onSurfaceTextureAvailable(SurfaceTexture arg0, int arg1,
   int arg2) {
  mCamera = Camera.open();
  Camera.Size previewSize = mCamera.getParameters().getPreviewSize();
  mTexture.setLayoutParams(new FrameLayout.LayoutParams(
    previewSize.width, previewSize.height, Gravity.CENTER));
  try {
   mCamera.setPreviewTexture(arg0);
  } catch (IOException t) {
  }
  mCamera.startPreview();
  mTexture.setAlpha(1.0f);
```

```
 mTexture.setRotation(90.0f);
}

 @Override
public boolean onSurfaceTextureDestroyed(SurfaceTexture arg0) {
 mCamera.stopPreview();
 mCamera.release();
 return true;
}

 @Override
public void onSurfaceTextureSizeChanged(SurfaceTexture arg0, int arg1,
  int arg2) {
 // TODO Auto-generated method stub
}

 @Override
public void onSurfaceTextureUpdated(SurfaceTexture arg0) {
 // TODO Auto-generated method stub
}
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.androidtextureview"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="18" />
   <uses-permission android:name="android.permission.CAMERA" />
   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name="com.example.androidtextureview.MainActivity"
         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>
```

- ✓ **NumberPicker**

```
<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" >


  <TextView
    android:id="@+id/textView1"
    android:textSize="20dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="46dp"
    android:text="Static NumberPicker" />


  <NumberPicker
    android:id="@+id/numberPicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"  />


  <TextView
    android:id="@+id/textView2"
    android:textSize="18dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/numberPicker1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="25dp"
    android:text="Old Value : " />


  <TextView
    android:id="@+id/textView3"
    android:textSize="18dp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView2"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="25dp"
    android:text="New Value : " />


</RelativeLayout>
```

```
package raj.numberpicker_static;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.NumberPicker;
```

```
import android.widget.NumberPicker.OnValueChangeListener;
import android.widget.TextView;

public class MainActivity extends Activity {

        NumberPicker np;
        TextView tv1, tv2;

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

    np = (NumberPicker) findViewById(R.id.numberPicker1);
    tv1 = (TextView) findViewById(R.id.textView2);
    tv2 = (TextView) findViewById(R.id.textView3);

    np.setMinValue(0);
    np.setMaxValue(10);
    np.setWrapSelectorWheel(false);

    np.setOnValueChangedListener(new OnValueChangeListener() {

            @Override
            public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
                // TODO Auto-generated method stub

                String Old = "Old Value : ";
                String New = "New Value : ";

                tv1.setText(Old.concat(String.valueOf(oldVal)));
                tv2.setText(New.concat(String.valueOf(newVal)));
            }
        });
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_main, menu);
    return true;
  }
}
```

- ✓ **ZoomBottom/ZoomControl**

```
<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" >


  <TextView
    android:id="@+id/textView1"
    android:textSize="18dp"
```

```
            android:textColor="#4169E1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="40dp"
            android:text="Static ZoomControls" />

        <ZoomControls
            android:id="@+id/zoomControls1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="40dp" />

        <ImageView
            android:id="@+id/imageView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:src="@drawable/ic_launcher" />

</RelativeLayout>
```

```
    Package raj.zoomcontrols_static;

    import android.os.Bundle;
    import android.app.Activity;
    import android.view.Menu;
    import android.view.View;
    import android.view.View.OnClickListener;
    import android.widget.ImageView;
    import android.widget.ZoomControls;

    public class MainActivity extends Activity {

            ZoomControls zoom;
            ImageView img;

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

            zoom = (ZoomControls) findViewById(R.id.zoomControls1);
            img = (ImageView) findViewById(R.id.imageView1);

            zoom.setOnZoomInClickListener(new OnClickListener() {

                    @Override
                    public void onClick(View v) {
                            // TODO Auto-generated method stub
```

```
                float x = img.getScaleX();
                float y = img.getScaleY();

                img.setScaleX((float) (x+1));
                img.setScaleY((float) (y+1));
            }
        });

        zoom.setOnZoomOutClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub


                float x = img.getScaleX();
                float y = img.getScaleY();

                img.setScaleX((float) (x-1));
                img.setScaleY((float) (y-1));
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

- ✓ **Dialerfilter**

activity_main.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"
    tools:context=".MainActivity" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="47dp"
        android:ems="10"
        android:inputType="phone" >
        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/buttonCall"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:text="DIAL" />


</RelativeLayout>
```

MainActivity.java

```
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity {

 private Button button;
private EditText etPhoneno;

 public void onCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 button = (Button) findViewById(R.id.buttonCall);
 etPhoneno = (EditText) findViewById(R.id.editText1);
 // add button listener
 button.setOnClickListener(new OnClickListener() {

   @Override
  public void onClick(View arg0) {
   String phnum = etPhoneno.getText().toString();
   Intent callIntent = new Intent(Intent.ACTION_CALL);
   callIntent.setData(Uri.parse("tel:" + phnum));
   startActivity(callIntent);
  }
 });
}

}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.makephonecall_android"
   android:versionCode="1"
```

```
        android:versionName="1.0" >

        <uses-sdk
          android:minSdkVersion="8"
          android:targetSdkVersion="18" />

        <uses-permission android:name="android.permission.CALL_PHONE" />

        <application
          android:allowBackup="true"
          android:icon="@drawable/ic_launcher"
          android:label="@string/app_name"
          android:theme="@style/AppTheme" >
          <activity
            android:name="com.example.makephonecall_android.MainActivity"
            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>
```

- ✓ **TwoLineListItem**
- ✓ **ActionMenuView**
- ✓ **TextClock**
- ✓ **Toolbar**

8.4. Linear Layout

Android LinearLayout is a view group that aligns all children in either *vertically* or *horizontally*.

LinearLayout Attributes

Following are the important attributes specific to LinearLayout –

Attribute	Description
android:id	This is the ID which uniquely identifies the layout.
android:baselineAligned	This must be a boolean value, either "true" or "false" and prevents the layout from aligning its children's baselines.
android:baselineAlignedChildIndex	When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align
android:divider	This is drawable to use as a vertical divider between buttons. You use a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
android:gravity	This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
android:orientation	This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal.
android:weightSum	Sum up of child weight

```
package com.example.demo;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
    @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
  }
}
```

Following will be the content of **res/layout/activity_main.xml file** –

```
<?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/btnStartService"
    android:layout_width="270dp"
```

```
    android:layout_height="wrap_content"
    android:text="start_service"/>

  <Button android:id="@+id/btnPauseService"
    android:layout_width="270dp"
    android:layout_height="wrap_content"
    android:text="pause_service"/>

  <Button android:id="@+id/btnStopService"
    android:layout_width="270dp"
    android:layout_height="wrap_content"
    android:text="stop_service"/>

</LinearLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

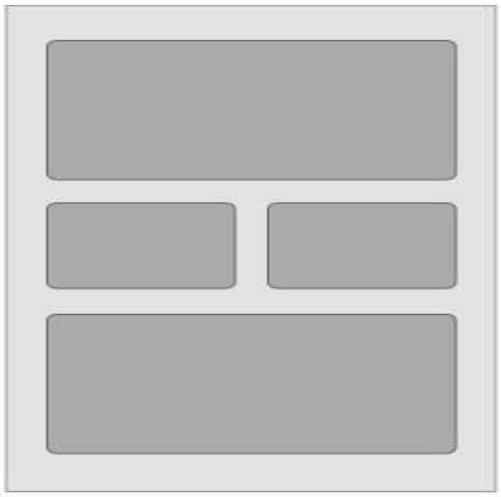
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">HelloWorld</string>
  <string name="action_settings">Settings</string>
</resources>
```

Let's try to run our modified **Hello World!** application we just modified. I assume you had created your **AVD** while doing environment setup. To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. Android studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window −

Now let's change the orientation of Layout as**android:orientation="horizontal"** and try to run the same application, it will give following screen −

8.5. Relative Layouts

Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.



RelativeLayout Attributes

Following are the important attributes specific to RelativeLayout –

Attribute	Description
android:id	This is the ID which uniquely identifies the layout
android:gravity	This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc
android:ignoreGravity	This indicates what view should not be affected by gravity.

Using RelativeLayout, you can align two elements by right border, or make one below another, centered in the screen, centered left, and so on. 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** and few of the important attributes are given below –

Attribute	Description
android:layout_above Positions	the bottom edge of this view above the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name"

android:layout_alignBottom	Makes the bottom edge of this view match the bottom edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_alignLeft	Makes the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_alignParentBottom	If true, makes the bottom edge of this view match the bottom edge of the parent. Must be a boolean value, either "true" or "false".
android:layout_alignParentEnd	If true, makes the end edge of this view match the end edge of the parent. Must be a boolean value, either "true" or "false".
android:layout_alignParentLeft	If true, makes the left edge of this view match the left edge of the parent. Must be a boolean value, either "true" or "false".
android:layout_alignParentRight	If true, makes the right edge of this view match the right edge of the parent. Must be a boolean value, either "true" or "false".
android:layout_alignParentStart	If true, makes the start edge of this view match the start edge of the parent. Must be a boolean value, either "true" or "false".
android:layout_alignParentTop	If true, makes the top edge of this view match the top edge of the parent. Must be a boolean value, either "true" or "false".
android:layout_alignRight	Makes the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_alignStart	Makes the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_alignTop	Makes the top edge of this view match the top edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_below Positions	the top edge of this view below the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_centerHorizontal	If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".
android:layout_centerInParent	If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".
android:layout_centerVertical	If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".
android:layout_toEndOf	Positions the start edge of this view to the end of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_toLeftOf	Positions the right edge of this view to the left of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_toRightOf	Positions the left edge of this view to the right of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_toStartOf	Positions the end edge of this view to the start of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

```
package com.example.demo;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import android.os.Bundle;
import android.app.Activity;
import android.text.format.DateFormat;
import android.view.Menu;
import android.widget.TextView;
public class MainActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu;
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
  }
}
```

Following will be the content of **res/layout/activity_main.xml** file:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:paddingLeft="16dp"
  android:paddingRight="16dp" >
  <EditText
    android:id="@+id/name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/reminder" />
  <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/name">
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button" />
```

```
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New Button"
    android:id="@+id/button2" />

  </LinearLayout>

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

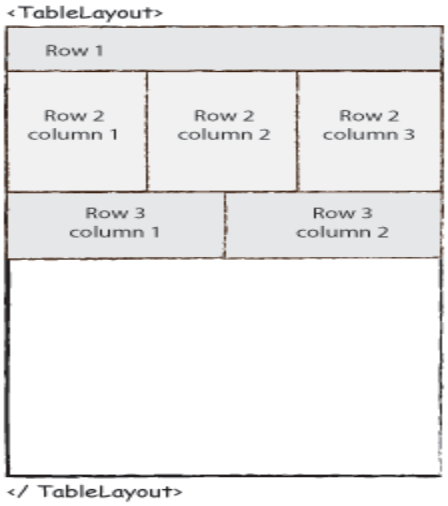
```
<?xml version="1.0" encoding="utf-8"?>

<resources>

  <string name="action_settings">Settings</string>

  <string name="reminder">Enter your name</string>

</resources>
```

Let's try to run our modified **Hello World!** application we just modified. I assume you had created your **AVD** while doing environment setup. To run the app from Android Studio, open one of your project's activity files and click Run icon from the toolbar. Android Studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window −

8.6. Table Layout

Android TableLayout going to be arranged groups of views into rows and columns. You will use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object.

TableLayout containers do not display border lines for their rows, columns, or cells.



Following are the important attributes specific to TableLayout –

Attribute	Description
android:id	This is the ID which uniquely identifies the layout.
android:collapseColumns	This specifies the zero-based index of the columns to collapse. The column indices must be separated by a comma: 1, 2, 5.
android:collapseColumns	The zero-based index of the columns to shrink. The column indices must be separated by a comma: 1, 2, 5.
android:stretchColumns	The zero-based index of the columns to stretch. The column indices must be separated by a comma: 1, 2, 5.

Activity_Main.xml

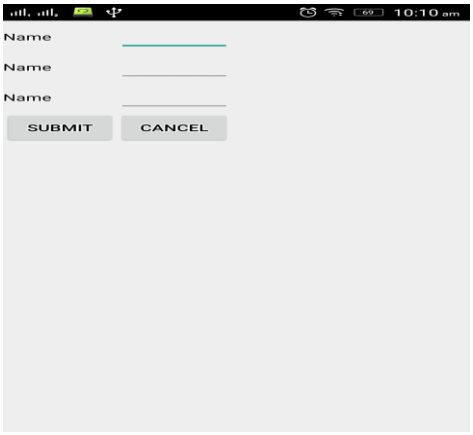
```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/TableLayout1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="${relativePackage}.${activityClass}" >
<TableRow
      android:id="@+id/tr1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" >
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Name" />
   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

   </TableRow>
   <TableRow
      android:id="@+id/tr2"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" >
   <TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name" />
  <EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
  </TableRow>
  <TableRow
    android:id="@+id/tr3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name" />
  <EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
  </TableRow>
  <TableRow
    android:id="@+id/tr4"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Submit" />
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cancel" />
  </TableRow>
        </TableLayout>
```

MainActivity.java

```
package com.example.tablelayout;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
public class MainActivity extends Activity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
        }
}
```

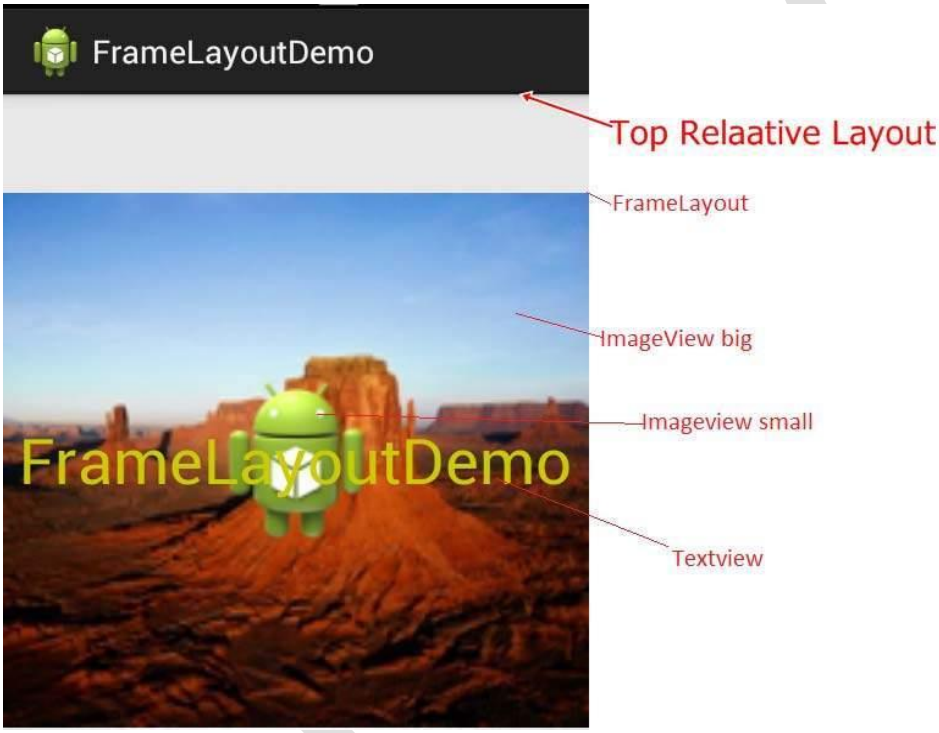
Output:



8.7. Frame Layout

Frame Layout is designed to block out an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other.

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.



FrameLayout Attributes

Following are the important attributes specific to FrameLayout –

Attribute	Description
android:id	This is the ID which uniquely identifies the layout.
android:foreground	This defines the drawable to draw over the content and possible values may be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".

android:foregroundGravity	Defines the gravity to apply to the foreground drawable. The gravity defaults to fill. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
android:measureAllChildren	Determines whether to measure all children or just those in the VISIBLE or INVISIBLE state when measuring. Defaults to false.

```
package com.example.demo;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
  }
}
```

Following will be the content of **res/layout/activity_main.xml** file −

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

  <ImageView
    android:src="@drawable/ic_launcher"
    android:scaleType="fitCenter"
    android:layout_height="250px"
    android:layout_width="250px"/>

  <TextView
    android:text="Frame Demo"
    android:textSize="30px"
    android:textStyle="bold"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:gravity="center"/>
</FrameLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

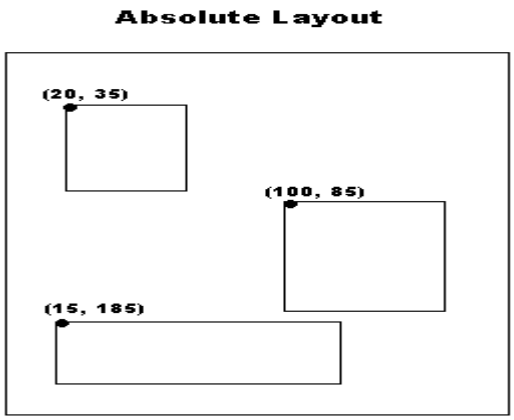
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
  <string name="app_name">demo</string>
  <string name="action_settings">Settings</string>
</resources>
```

Let's try to run our modified **Hello World!** application we just modified. I assume you had created your **AVD** while doing environment setup. To run the app from Android Studio, open one of your project's activity files and click Run icon from the toolbar. Android Studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window –

8.8. Absolute Layout

An Absolute Layout lets you specify exact locations (x/y coordinates) of its children. Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning.



AbsoluteLayout Attributes

Following are the important attributes specific to AbsoluteLayout –

Attribute	Description
android:id	This is the ID which uniquely identifies the layout.
android:layout_x	This specifies the x-coordinate of the view.

android:layout_y	This specifies the y-coordinate of the view.

```
package com.example.demo;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return true;

  }

}
```

Following will be the content of **res/layout/activity_main.xml** file −

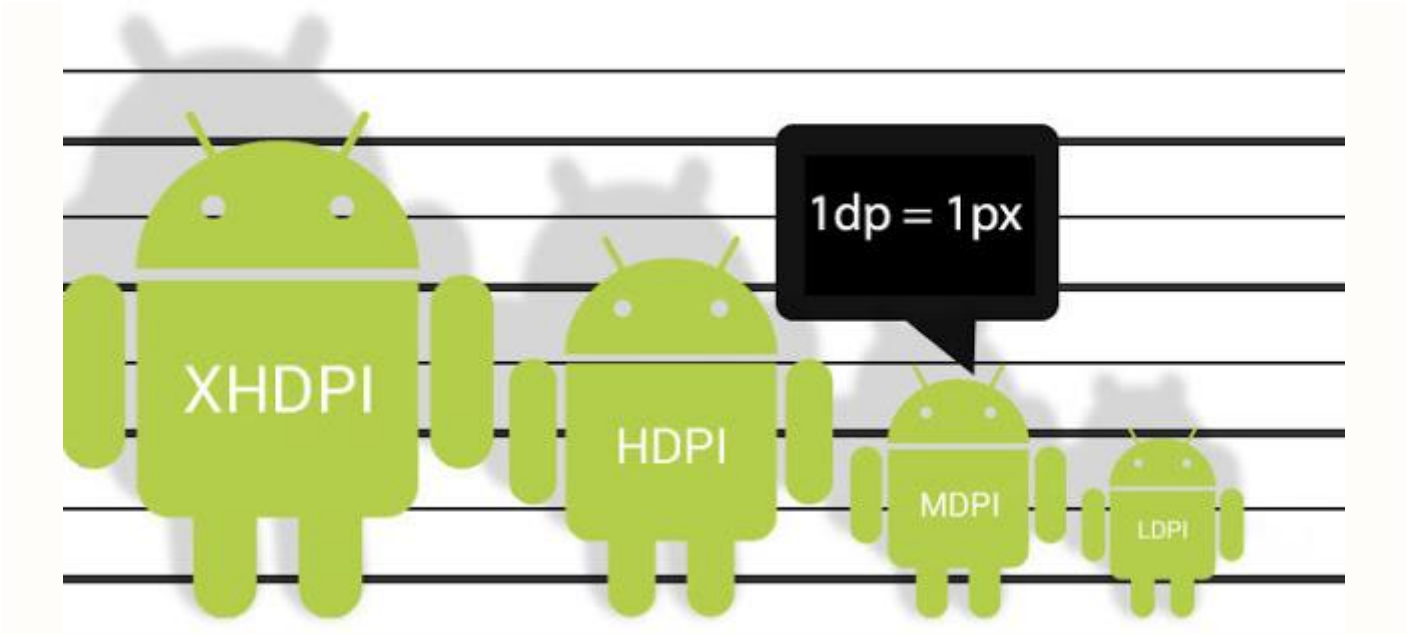
```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
  android:layout_height="fill_parent">

  <Button
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="OK"
    android:layout_x="50px"
    android:layout_y="361px" />

  <Button     android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="Cancel"
    android:layout_x="225px"
    android:layout_y="361px" />

</AbsoluteLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">demo</string>
  <string name="action_settings">Settings</string>

</resources>
```

Let's try to run our modified **Hello World!** application we just modified. I assume you had created your **AVD** while doing environment setup. To run the app from Android Studio, open one of your project's activity files and click Run icon from the toolbar. Android Studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window −

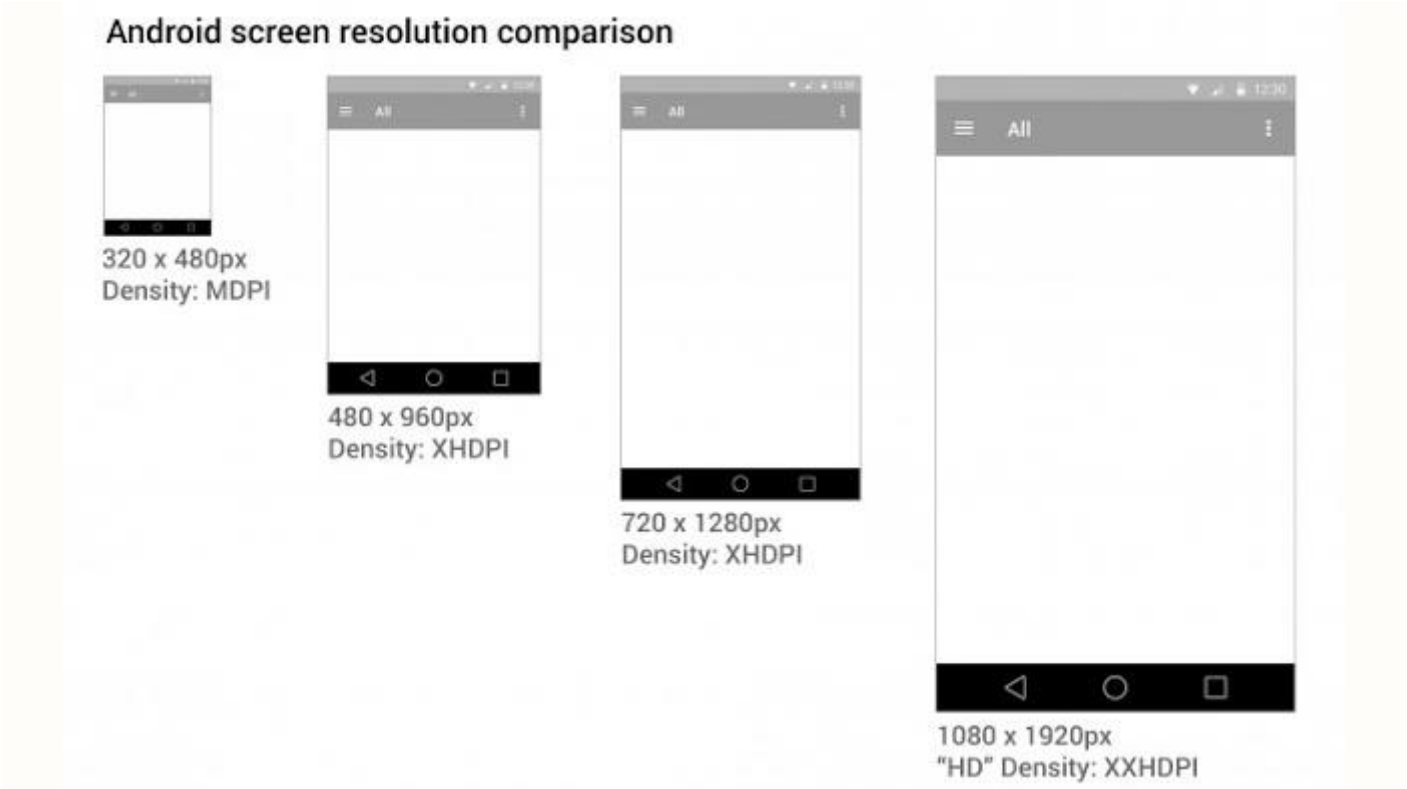
8.9. Design for Tablets and High Resolution Devices



July 2014 EDIT: Android's "L" design guidelines now supply templates in Adobe Illustrator, making it even harder for designers user to Photoshop or Fireworks. They don't provide an answer for how you should crop and optimise your assets for the developer (I assume they'd rather you didn't and just use their native stuff).

See www.google.com/design/

July 2014 EDIT: On consideration of comments from NexogenDev, I have removed the more explicit reference to 72dpi in the graphic above to "1dp = 1px" as this is a more literal starting point and probably better. My point was that Android supplied templates and assets tend to open by default in Photoshop as 72dpi, but dpi in general is misleading in this context.



I found it hard to know where to start designing for Android in Photoshop.

Google tries to help by providing extensive Android user Interface Design Guidelines. There are many devices running Android that have different screen sizes and shapes. There are also different screen densities, which means the quality of the definition of the screen or how many pixels fit in the same sized area on the screen – e.g. medium screen density has 160 pixels in every inch on the device's screen, whereas an extra high density screen has twice as many, 320 pixels, crammed in an inch. To cover the different densities you need to supply graphics and icons at various densities.

Creating a design that will work for all these devices can be daunting. You also need to create different density icons and graphics.

1. Minimum handsetscreen size

I started with a minimum mobile handset size screen first. Starting with the smallest and most limited screen size first makes sure your content and UI will fit, removing any features that aren't completely necessary to the core purpose of the app. Then you can work out how your app will behave on larger and tablet sized screens. You need to decided if your app is going to stretch to fit larger screens, or if you're going to have a completely different layout and design for tablet sized screens.

2. Photoshop image size

I choose to start with a small mobile handset and a small 7" tablet screen as a base for design and then think of how that could scale elagantly. You may need to create **portrait** and **landscape** versions if you plan on allowing users to change orientation.

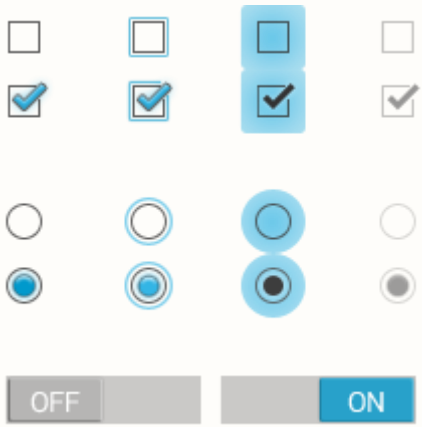
Here are some Photoshop templates

3. Download Android design templates

If you are new to designing for Android, and are confused by dpi, I STRONGLY RECOMMEND you use native Android styling and assets, rather than doing a completely custom design at first.

See www.google.com/design/

It is **important to keep the icons at the same proportion** to maintain the correct "hit" area for the user's finger.



4. Scale down

It's easier to work at the higher pixel density, XXHDPI. It is always advisable to create icon art as vectors in "Smart Objects, or as "shapes" in Photoshop so they can be scaled up or down without loosing quality.

The benefit of working at XXHDPI is that you can be sure you are not loosing any quality.

XXHDPI – 300% – x3.o
XHDPI – 200% – x2.o
HDPI – 150% – x1.5
MDPI – 100% – x1.0
LDPI – 75% (don't bother with this!) – 0.75

When using vectors, you may want to adjust your line art to tighten up any half pixels or it can look blurry at the edges (if you're a perfectionist like me).

5. Workflow

- Create you design PSD at XXHDPI
- Crop out your graphic assets and icons as separate PSDs
- Scale them down to the various densities: XHDPI, HDPI, MDPI

My method is to design one PSD, crop out the icons assets as PSDs, then scale them to the different densities. I often start with a PSD at XXHDPI or MDPI (providing my artwork is vector and can be scaled up or down), then I crop out the icons and assets out, then resize them.

The reason this is easier is that when you need to make design changes, you don't have to change multiple PSDs.

What the frakk is a DP?

Here's what I discovered after a few hours of scratching my head and moaning…

1. "dp" is a magic, made-up measurement that only exists in the world of The Matrix & Tron.
2. Android icons are 32dp x 32dp @ the medium density MDPI
3. Android supply sample icons (Edit: they don't any more the gits) and if you open an icon from the MDPI folder in Photoshop, the icon is 32px x 32px with "image size".
4. SO, in Photoshop:1dp = 1px @ MDPI – 100%
1dp = 1.5px @ HDPI – 150%
1dp = 2px @ XHDPI – 200%
1dp = 3px @ XXHDPI – 300%
1dp = 4px @ XXXHDPI – 400%
5. It doesn't matter what DPI you have set in PHOTOSHOP, it's the canvas size (the number of pixels available) that counts.

If you have a better way of getting started designing in Android.

8.10. Handling UI and Event Listeners

Actvity_Main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/LinearLayout1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context="${relativePackage}.${activityClass}" >
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:text="@string/hello_world"
      android:textSize="20dp"
      android:textStyle="bold" />
   <EditText
      android:id="@+id/editText1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:textColor="#0000EE"
      android:textSize="20dp" >
      <requestFocus />
   </EditText>
   <EditText
      android:id="@+id/editText2"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10"
      android:textColor="#0000EE"
      android:textSize="20dp" />
   <Button
      android:id="@+id/button1"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Clickme"
    android:textColor="#0000EE"
    android:textSize="20dp" />
</LinearLayout>
```

MainActivity.Java

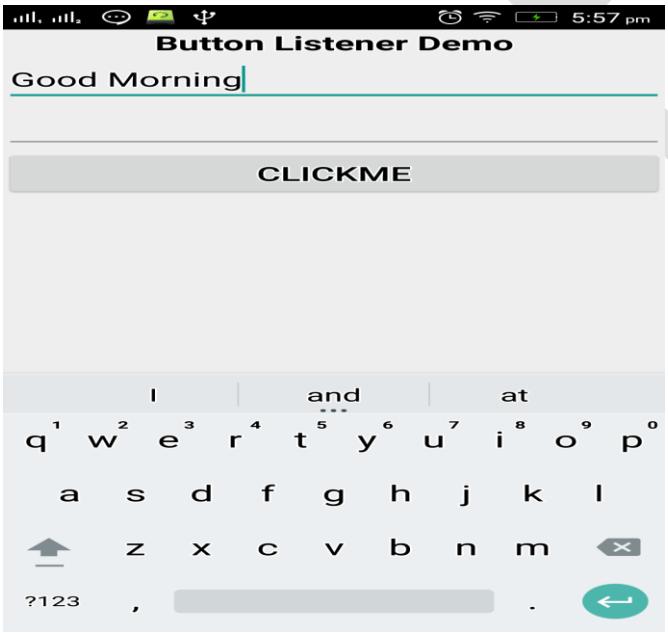
```
package com.example.clicklisteners;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity {
        Button b;
        EditText et1,et2;

        @Override
        protected void onCreate(Bundle savedInstanceState) {            super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                et1=(EditText) findViewById(R.id.editText1);
                et2=(EditText) findViewById(R.id.editText2);
                b=(Button) findViewById(R.id.button1);
                b.setOnClickListener(new OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                // TODO Auto-generated method stub
                                et2.setText("Hello_____"+et1.getText());
                        }
                });
        }

}
```

String.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">ClickListeners</string>
   <string name="hello_world">Button Listener Demo</string>
</resources>
```

Output:


5:56 pm
Button Listener Demo
CLICKME
wat
i
Hooo


5:57 pm
Button Listener Demo
Good Morning
CLICKME
I
and
at

ActivityMain.java

Activity_main.xml

```
package com.example.buttonlistener;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends Activity {
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
	}
	public void ViewMessage(View v){
		Toast.makeText(getApplicationContext(), "Hii", 5000).show();
```

```
        }
}
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/LinearLayout1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context="${relativePackage}.${activityClass}" >

   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:text="@string/hello_world"
      android:textSize="20dp"
      android:textStyle="bold" />

   <Button
      android:id="@+id/button1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Clickme"
      android:textColor="#0000EE"
      android:textSize="20dp"
      android:onClick="ViewMessage" />

</LinearLayout>
```

Strings.xml

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

   <string name="app_name">ButtonListener</string>
   <string name="hello_world">Button Listener Demo</string>

</resources>
```

Output:

8.11. Dynamic Layout Design

Android activity contains various user interface components e.g Button, Radio button, List, Text field etc. The user interface component can be arranged/attached with the activity in two different ways.

1. Declaring UI element in layoutfile

Basically, Layout file is xml formatted file present in ***res/layout***. In this file, tags are used to define any UI component and properties are used to configure the components. It is similar to HTML tags. This style is preferred because UI related codes and business logic codes are separated. It is comparatively easier to manage.

2. Creating view element at runtime

UI Companent can be created programmatically. UI component class setter methods help to configure the component. This style is not recommended unless it's really required. In this style, business logic gets mixed with the component UI code. It doesn't look neat and sometimes it's hard to manage.

MainActivity.Java

```
package com.example.dynamicxml;

import android.app.Activity;
import android.content.res.ColorStateList;
import android.graphics.Color;
import android.os.Bundle;
import android.text.InputType;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RatingBar;
import android.widget.TextView;
import android.widget.ToggleButton;
```

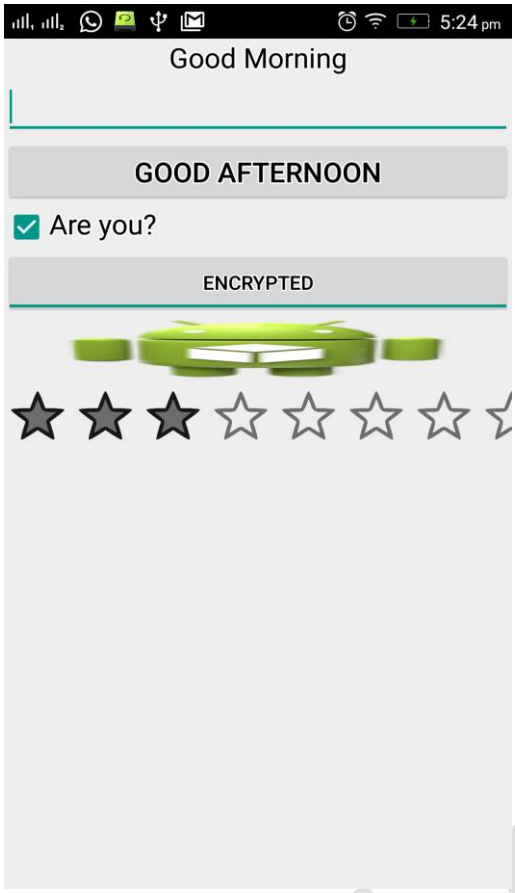
```
public class MainActivity extends Activity {
	LinearLayout layout;
	TextView tv;
	EditText et;
	Button bt;
	CheckBox c;
	ToggleButton tb;
	ImageView iv;
	RatingBar rb;


	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		//setContentView(R.layout.activity_main);
		layout=new LinearLayout(this);
		layout.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));
		layout.setOrientation(LinearLayout.VERTICAL);

		tv=new TextView(this);
		tv.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT));
		tv.setText("Good Morning");
		tv.setTextColor(Color.BLUE);
		tv.setGravity(Gravity.CENTER);
		tv.setTextSize(20);
		layout.addView(tv);

		et=new EditText(this);
		et.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT));
		et.setInputType(InputType.TYPE_CLASS_TEXT|InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS);
		layout.addView(et);

		bt=new Button(this);
		bt.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT));
		bt.setText("Good Afternoon");
		bt.setTextColor(Color.BLUE);
		bt.setGravity(Gravity.CENTER);
		bt.setTextSize(20);
		layout.addView(bt);

		c=new CheckBox(this);
		c.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT));
		c.setText("Are you?");
		c.setTextColor(Color.BLUE);
		c.setGravity(Gravity.LEFT);
		c.setTextSize(20);
		c.setChecked(true);
```

```
                layout.addView(c);

                tb=new ToggleButton(this);
                tb.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT));
                tb.setTextOn("Encrypted");
                tb.setTextOff("Decrypted");
                tb.setChecked(true);
                layout.addView(tb);

                iv=new ImageView(this);
                iv.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT));
                iv.setBackgroundResource(R.drawable.ic_launcher);
                layout.addView(iv);

                rb=new RatingBar(this);
                rb.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT));
                rb.setNumStars(5);
                rb.setRating(2);
                rb.setStepSize(1);
                layout.addView(rb);

                setContentView(layout);


        }
}
```

Activity_main.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"
   tools:context="${relativePackage}.${activityClass}" >

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

</RelativeLayout>
```

Output



8.12. Styles for UI components

A **style** resource defines the format and look for a UI. A style can be applied to an individual View (from within a layout file) or to an entire Activity or application (from within the manifest file).

1.1. Defining Styles

A style is defined in an XML resource that is separate from the XML that specifies the layout. This XML file resides under **res/values/** directory of your project and will have **<resources>** as the root node which is mandatory for the style file. The name of the XML file is arbitrary, but it must use the .xml extension.

You can define multiple styles per file using **<style>** tag but each style will have its name that uniquely identifies the style. Android style attributes are set using **<item>** tag as shown below −

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="CustomFontStyle">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:capitalize">characters</item>
    <item name="android:typeface">monospace</item>
    <item name="android:textSize">12pt</item>
    <item name="android:textColor">#00FF00</item>/>
  </style>
```

```
</resources>
```

The value for the <item> can be a keyword string, a hex color, a reference to another resource type, or other value depending on the style property.

1.2. Using Styles

Once your style is defined, you can use it in your XML Layout file using **style**attribute as follows –

```
<?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" >

  <TextView
    android:id="@+id/text_id"
    style="@style/CustomFontStyle"
    android:text="@string/hello_world" />

</LinearLayout>
```

To understand the concept related to Android Style, you can check Style Demo Example.

1.3. Style Inheritance

Android supports style Inheritance in very much similar way as cascading style sheet in web design. You can use this to inherit properties from an existing style and then define only the properties that you want to change or add.

To implement a custom theme create or edit MyAndroidApp/res/values/themes.xml and add the following –

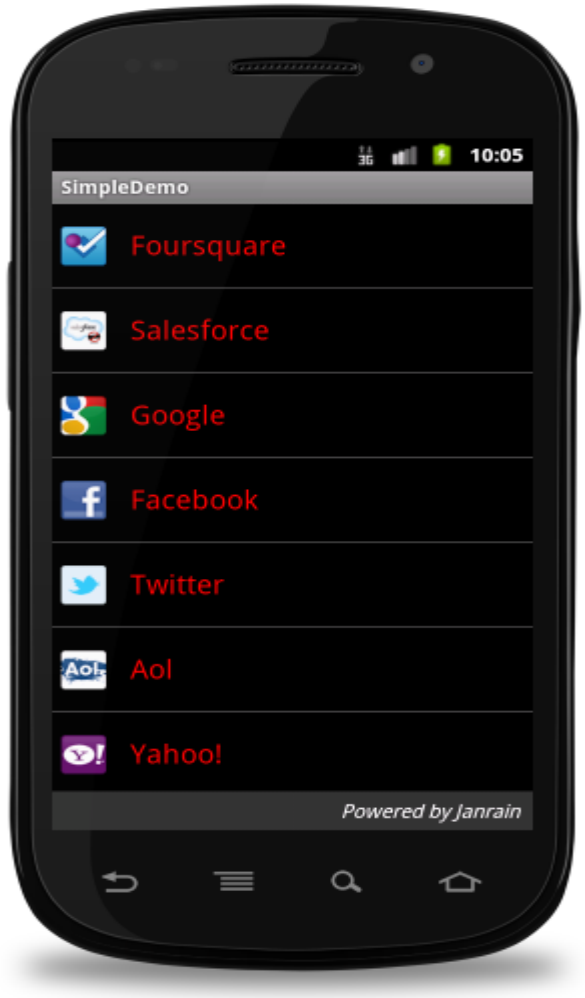
```
<resources>
  ...
  <style name="MyCustomTheme" parent="android:style/Theme">
  <item name="android:textColorPrimary">#ffff0000</item>
  </style>
  ...
</resources>
```

In your AndroidManifest.xml apply the theme to the activities you want to style –

```
<activity
  android:name="com.myapp.MyActivity"
  ...
```

```
android:theme="@style/MyCustomTheme"
/>
```

Your new theme will be applied to your activity, and text is now bright red.



Applying Colors to Theme Attributes

Your color resource can then be applied to some theme attributes, such as the window background and the primary text color, by adding elements to your custom theme. These attributes are defined in your styles.xml file. For example, to apply the custom color to the window background, add the following two elements to your custom theme, defined in MyAndroidApp/res/values/styles.xml file –

```
<resources>
  ...
  <style name="MyCustomTheme" ...>
    <item name="android:windowBackground">@color/my_custom_color</item>
    <item name="android:colorBackgroundCacheHint">@color/my_custom_color</item>
  </style>
  ...
</resources>
```

Using a Custom Nine-Patch With Buttons

A nine-patch drawable is a special kind of image which can be scaled in width and height while maintaining its visual integrity. Nine-patches are the most common way to specify the appearance of Android buttons, though any drawable type can be used.

A SAMPLE OF NINE-PATCH BUTTON

Steps to create Nine-Patch Buttons

- Save this bitmap as /res/drawable/my_nine_patch.9.png
- Define a new style
- Apply the new button style to the buttonStyle attribute of your custom theme

DEFINE A NEW STYLE

```
<resources>
  ...
    <style name="MyCustomButton" parent="android:Widget.Button">
    <item name="android:background">@drawable/my_nine_patch</item>
    </style>
  ...
</resources>
```

APPLY THE THEME

```
<resources>
  ...
    <style name="MyCustomTheme" parent=...>
      ...
      <item name="android:buttonStyle">@style/MyCustomButton</item>
    </style>
  ...
</resources>
```

Android Themes

Hope you understood the concept of Style, so now let's try to understand what is a **Theme**. A theme is nothing but an Android style applied to an entire Activity or application, rather than an individual View. Thus, when a style is applied as a theme, every **View** in the Activity or application will apply each style property that it supports. For example, you can apply the same **CustomFontStyle** style as a theme for an Activity and then all text inside that **Activity** will have green monospace font.

To set a theme for all the activities of your application, open the**AndroidManifest.xml** file and edit the **<application>** tag to include the**android:theme** attribute with the style name. For example −

```
<application android:theme="@style/CustomFontStyle">
```

But if you want a theme applied to just one Activity in your application, then add the android:theme attribute to the <activity> tag only. For example −

```
<activity android:theme="@style/CustomFontStyle">
```

There are number of default themes defined by Android which you can use directly or inherit them using **parent** attribute as follows −

```
<style name="CustomTheme" parent="android:Theme.Light">
  ...
</style>
```

To understand the concept related to Android Theme, you can check Theme Demo Example.

Styling the colour palette

The layout design can implementable based on them based colours, for example as following design is designed based on them colour(blue)

Above layout has designed based on style.xml file,Which has placed at**res/values/**

```
<resource>
  <style name="AppTheme" parent="android:Theme.Material">
    <item name ="android:color/primary">@color/primary</item>
    <item name ="android:color/primaryDark">@color/primary_dark</item>
    <item name ="android:colorAccent/primary">@color/accent</item>
  </style>
<resource>
```

9. Intents

Android Intent is the *message* that is passed between components such as activities, content providers, broadcast receivers, services etc.

It is generally used with startActivity() method to invoke activity, broadcast receivers etc.

The **dictionary meaning** of intent is *intention or purpose*. So, it can be described as the intention to do action.

The LabeledIntent is the subclass of android.content.Intent class.

Android intents are mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

There are two types of intents in android: implicit and explicit.

1) Implicit Intent

Implicit Intent doesn't specifiy the component. In such case, intent provides information of available components provided by the system that is to be invoked.

For example, you may write the following code to view the webpage.

```
Intent intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://androindian.com/"));
startActivity(intent);
```

Program for Implicit Intent

Activity_main.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"
   tools:context="${relativePackage}.${activityClass}" >

   <TextView
      android:id="@+id/textView1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Implicit Intent Example"
      android:gravity="center"/>

   <Button
      android:id="@+id/button1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
```

```
    android:layout_below="@+id/textView1"
    android:layout_marginTop="137dp"
    android:text="Click me" />

</RelativeLayout>
```

ActivityMain.Java

```
package com.raj.implicitintent;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {

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

		Button b=(Button) findViewById(R.id.button1);

		b.setOnClickListener(new OnClickListener() {

			@Override
			public void onClick(View v) {
				// TODO Auto-generated method stub
				String url = "http://androindian.com/";
				Intent i = new Intent(Intent.ACTION_VIEW);
				i.setData(Uri.parse(url));
				startActivity(i);

			}
		});
	}
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.raj.implicitintent"
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="21" />
```

```
  <uses-permission android:name="android.permission.INTERNET"/>

  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name=".MainActivity"
      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>
```

2) Explicit Intent

Explicit Intent specifies the component. In such case, intent provides the external class to be invoked.

```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);
startActivity(i);
```

To get the full code of explicit intent, visit the next page

Program for Explicit Intents

Activity _Main.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"
  tools:context="${relativePackage}.${activityClass}" >

  <TextView
    android:id="@+id/textView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Explicit Intent Example"
    android:gravity="center" />

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="88dp"
```

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

            </RelativeLayout>
package com.raj.explicitintent;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {

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

                bt=(Button) findViewById(R.id.button1);

                bt.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                // TODO Auto-generated method stub

                                Intent intent=new Intent(getApplicationContext(), Second.class);
                                startActivity(intent);

                        }
                });
        }
}

            Second_Activity.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"
  tools:context="${relativePackage}.${activityClass}" >

  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hi Mr. Raj.....Welcome to SEcond Activity"
    android:textSize="20sp"
    android:gravity="center"/>

</RelativeLayout>
```

SecondActivity.java

```
package com.raj.explicitintent;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class Second extends Activity {

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

Android Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.raj.explicitintent"
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="21" />

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

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity
      android:name=".Second"
      android:label="@string/title_activity_second" >
    </activity>
  </application>

</manifest>
```

9.1. Passing Data Through Intents

Using intents data sharing between different activities.

```
<?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">
<TextView
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:layout_marginTop="10dp"
 android:gravity="center_horizontal"
 android:text="Input your Name">
</TextView>
<TableLayout
 android:id="@+id/tableLayout1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="10dp"
 android:stretchColumns="1">
<TableRow
 android:id="@+id/tableRow1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">
<TextView
 android:id="@+id/textView1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="First Name">
</TextView>
<EditText
 android:id="@+id/etFName"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginLeft="5dp">
<requestFocus>
</requestFocus>
</EditText>
</TableRow>
<TableRow
 android:id="@+id/tableRow2"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">
<TextView
 android:id="@+id/textView2"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Last Name">
</TextView>
<EditText
 android:id="@+id/etLName"
```

```
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginLeft="5dp">
</EditText>
</TableRow>
</TableLayout>
<Button
 android:id="@+id/btnSubmit"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center_horizontal"
 android:text="Submit">
</Button>
</LinearLayout>
```

Main Activity

```
package ru.startandroid.develop.p0281intentextras;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity implements OnClickListener {

 EditText etFName;
 EditText etLName;
 Button btnSubmit;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    etFName = (EditText) findViewById(R.id.etFName);
    etLName = (EditText) findViewById(R.id.etLName);

    btnSubmit = (Button) findViewById(R.id.btnSubmit);
    btnSubmit.setOnClickListener(this);

  }


 @Override
 public void onClick(View v) {
  Intent intent = new Intent(this, ViewActivity.class);
  intent.putExtra("fname", etFName.getText().toString());
  intent.putExtra("lname", etLName.getText().toString());
  startActivity(intent);
 }
```

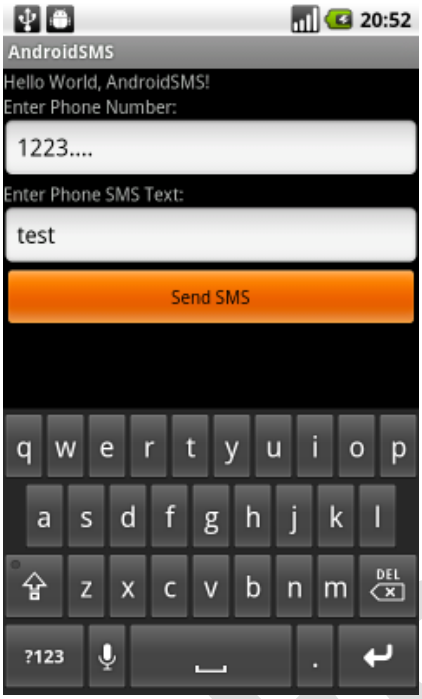
```
}
```

```
<?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:orientation="vertical">
<TextView
 android:id="@+id/tvView"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_gravity="center_horizontal"
 android:layout_marginTop="20dp"
 android:text="TextView"
 android:textSize="20sp">
</TextView>
</LinearLayout>
```

```
package ru.startandroid.develop.p0281intentextras;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class ViewActivity extends Activity {

 TextView tvView;

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

  tvView = (TextView) findViewById(R.id.tvView);

  Intent intent = getIntent();

  String fName = intent.getStringExtra("fname");
  String lName = intent.getStringExtra("lname");

  tvView.setText("Your name is: " + fName + " " + lName);
 }
}
```

SMS Manager :

Android OS provide android.telephony.SmsManager, to manages SMS operations such as sending data, text, and pdu SMS messages. Get this object by calling the static method SmsManager.getDefault(). Here is a example to send SMS sing android.telephony.SmsManager.



Modify main.xml to have two EditText for user to enter phone number and text.

```
<?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"
 >
<TextView
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="@string/hello"
 />
<TextView
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Enter Phone Number:"
 />
<EditText
 android:id="@+id/smsnumber"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:inputType="phone"
 />
<TextView
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text="Enter Phone SMS Text:"
```

```
 />
<EditText
 android:id="@+id/smstext"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 />
<Button
 android:id="@+id/sendsms"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 android:text=" Send SMS "
 />
</LinearLayout>
```

main code, AndroidSMS.java

```
package com.exercise.AndroidSMS;
import android.app.Activity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class AndroidSMS extends Activity {
 /** Called when the activity is first created. */
 @Override
 public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    final EditText edittextSmsNumber = (EditText)findViewById(R.id.smsnumber);
    final EditText edittextSmsText = (EditText)findViewById(R.id.smstext);
    Button buttonSendSms = (Button)findViewById(R.id.sendsms);
    buttonSendSms.setOnClickListener(new Button.OnClickListener(){
@Override
public void onClick(View arg0) {
 SmsManager smsManager = SmsManager.getDefault();
 String smsNumber = edittextSmsNumber.getText().toString();
 String smsText = edittextSmsText.getText().toString();
 smsManager.sendTextMessage(smsNumber, null, smsText, null, null);
}});
 }
}
```

Modify AndroidManifest.xml to grant permission of "android.permission.SEND_SMS".

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.exercise.AndroidSMS"
   android:versionCode="1"
   android:versionName="1.0">
 <uses-sdk android:minSdkVersion="4" />
 <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".AndroidSMS"
            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>
 <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
</manifest>
```

Monitoring SMS :

```
package com.androidbook.telephony;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.SmsMessage;
import android.util.Log;
public class MySMSMonitor extends BroadcastReceiver {
        private static final String ACTION = "android.provider.Telephony.SMS_RECEIVED";
        @Override
   public void onReceive(Context context, Intent intent)
   {
            Object[]pduArray = (Object[])
                    intent.getExtras().get("pdus");//pdus is predefined variable of Bundle object conist
incoming SMS information.
            SmsMessage[] messages = new SmsMessage[pduArray.length];
            for (int i = 0; i<pduArray.length; i++) {
               messages[i] = SmsMessage.createFromPdu((byte[])pduArray [i]);
               Log.d("MySMSMonitor", "From: " + messages[i].getOriginatingAddress());
               Log.d("MySMSMonitor", "Msg: " + messages[i].getMessageBody());
            }
            Log.d("MySMSMonitor","SMS Message Received.");
        }
}
```

Making Call & Phone State Listener :

Example :

Android Manifest.xml :

```
        <uses-permission android:name="android.permission.CALL_PHONE" />
        <uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

main.xml :

```
<?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/buttonCall"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="call 0377778888" />
</LinearLayout>
```

MainActivity.java:

```
public class MainActivity extends Activity {
        final Context context = this;
        private Button button;
```

```
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		button = (Button) findViewById(R.id.buttonCall);
		// add PhoneStateListener
		PhoneCallListener phoneListener = new PhoneCallListener();
		TelephonyManager telephonyManager = (TelephonyManager)
this.getSystemService(Context.TELEPHONY_SERVICE);
		telephonyManager.listen(phoneListener,PhoneStateListener.LISTEN_CALL_STATE);

		// add button listener
		button.setOnClickListener(new OnClickListener() {
			@Override
			public void onClick(View arg0) {
				Intent callIntent = new Intent(Intent.ACTION_CALL);
				callIntent.setData(Uri.parse("tel:0377778888"));
				startActivity(callIntent);
			}
		});
	}
	private class PhoneCallListener extends PhoneStateListener {
		private boolean isPhoneCalling = false;
		String LOG_TAG = "LOGGING 123";
		@Override
		public void onCallStateChanged(int state, String incomingNumber) {
			if (TelephonyManager.CALL_STATE_RINGING == state) {
				// phone ringing
				Log.i(LOG_TAG, "RINGING, number: " + incomingNumber);
			}

			if (TelephonyManager.CALL_STATE_OFFHOOK == state) {
				// active
				Log.i(LOG_TAG, "OFFHOOK");
				isPhoneCalling = true;
			}
			if (TelephonyManager.CALL_STATE_IDLE == state) {
				// run when class initial and phone call ended, need detect flag
				// from CALL_STATE_OFFHOOK
				Log.i(LOG_TAG, "IDLE");
				if (isPhoneCalling) {
					Log.i(LOG_TAG, "restart app");
					// restart app
					Intent i = getBaseContext().getPackageManager()
	.getLaunchIntentForPackage(getBaseContext().getPackageName());
					i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
					startActivity(i);
					isPhoneCalling = false;
				}
			}
		}
	}
}
```

Sending Email :

We can use Intent.ACTION_SEND to call an existing email client to send an Email.

See following code snippets :

```
Intent email = new Intent(Intent.ACTION_SEND);
email.putExtra(Intent.EXTRA_EMAIL, new String[]{"youremail@yahoo.com"});
email.putExtra(Intent.EXTRA_SUBJECT, "subject");
email.putExtra(Intent.EXTRA_TEXT, "message");
email.setType("message/rfc822");
startActivity(Intent.createChooser(email, "Choose an Email client :"));
```

Example :

main.xml

```
<?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/btnSendEmail"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Send Email"
   android:onClick="onClick" />
</LinearLayout>
```

Activity Code :

```
public class EmailsActivity extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
   }

   public void onClick(View v) {
      String[] to = {"Raj2708@gmail.com"};
      String[] cc = {"Raj2708@gmail.com"};
      sendEmail(to, cc, "Hello", "Hello my friends!");
   }

   //—-sends an SMS message to another device—-
   private void sendEmail(String[] to, String[] cc,String subject, String message) {
      Intent emailIntent = new Intent(Intent.ACTION_SEND);
      emailIntent.setData(Uri.parse("mailto:"));
      emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
      emailIntent.putExtra(Intent.EXTRA_CC, cc);
      emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
      emailIntent.putExtra(Intent.EXTRA_TEXT, message);
      emailIntent.setType("message/rfc822");
      startActivity(Intent.createChooser(emailIntent, "Email"));
   }
}
```

10. Dialogs
    10.1. Toast

Andorid Toast can be used to display information for the short period of time. A toast contains message to be displayed quickly and disappears after sometime.

The android.widget.Toast class is the subclass of java.lang.Object class.

You can also create custom toast as well for example toast displaying image. You can visit next page to see the code for custom toast.

Toast class

Toast class is used to show notification for a particular interval of time. After sometime it disappears. It doesn't block the user interaction.

Constants of Toast class

There are only 2 constants of Toast class which are given below.

Constant	Description
public static final int LENGTH_LONG	displays view for the long duration of time.
public static final int LENGTH_SHORT	displays view for the short duration of time.

Methods of Toast class

The widely used methods of Toast class are given below.

Method	Description
public static Toast makeText(Context context, CharSequence text, int duration)	makes the toast containing text and duration.
public void show()	displays toast.
public void setMargin (float horizontalMargin, float verticalMargin)	changes the horizontal and vertical margin difference.

Android Toast Example

```
Toast.makeText(getApplicationContext(),"Hello Androiindian",Toast.LENGTH_SHORT).show();
```

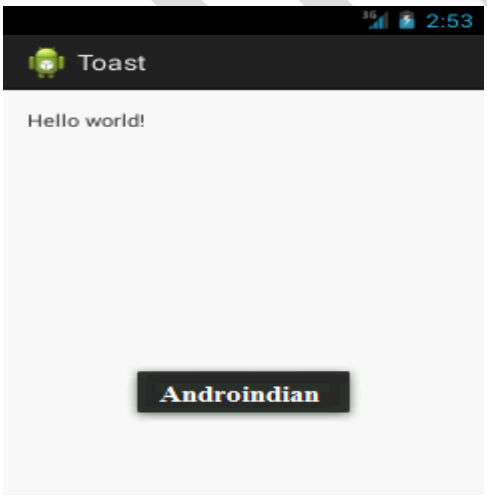
Another code:

```
Toast toast=Toast.makeText(getApplicationContext(),"Hello Androindian",Toast.LENGTH_SHORT);
toast.setMargin(50,50);
toast.show();
```

Example Program

```
package com.example.toast;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends Activity {
    @Override
      public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
 //Displaying Toast with Hello Javatpoint message
         Toast.makeText(getApplicationContext(),"Androindian",Toast.LENGTH_SHORT).show();
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
         getMenuInflater().inflate(R.menu.activity_main, menu);
         return true;
      }

}
```

Output:



10.2. Alert Dialogs

A dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for modal events that require users to take an action before they can proceed.

Working with Dialog Boxes

The Android Dialog class (android.app.Dialog) is the base class for all types of dialog controls that we can use within our Activity classes. Dialogs exists within the lifecycle of our Activity (android.app.Activity). They pop up in the foreground, blocking of our Activity screen, to catch the user's attention for a variety of reasons.

The Purpose of Dialogs :

- Inform the user of some event or progress (e.g. "You have mail!" Or "Downloading Message 1 of 200,000")
- Force the user to confirm an action (e.g. "Are you sure you want to delete all your contacts? Really sure?")
- Prompt the user for further information and collect it (e.g. "Please enter your username and password.")

A Note on Toast Messages: Some developers also use Toast messages (android.widget.Toast) for sending simple notifications or messages to the user. A Toast message displays over our Activity screen for a few seconds and then disappears automatically.

Toast over a Dialog is as follows: if the user is being informed of non-essential information, use a Toast, but if information is essential, use a Dialog. We use Toasts as very lightweight, informational notifications.

Displaying Custom Toast :

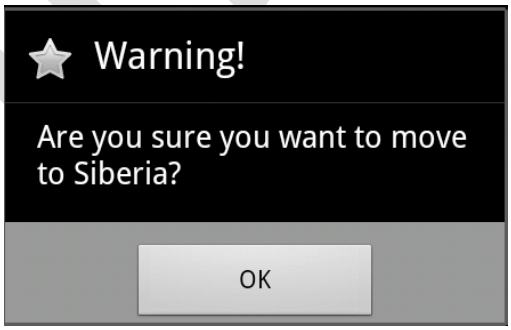
```
Toast toast = new Toast(getApplicationContext());

toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);

toast.setDuration(Toast.LENGTH_LONG);

toast.show();
```

Structure of Dialogs :

Dialogs have a number of different components, most of which are optional. A basic Dialog has:

- ✓ A title
- ✓ A message
- ✓ Buttons to collect user responses (e.g. Yes, No, Cancel, etc.)

A common Dialog configuration is shown below:



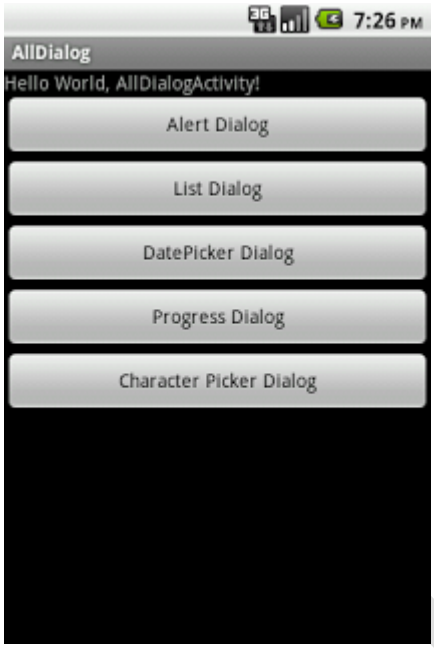
We can also create custom Dialog controls,

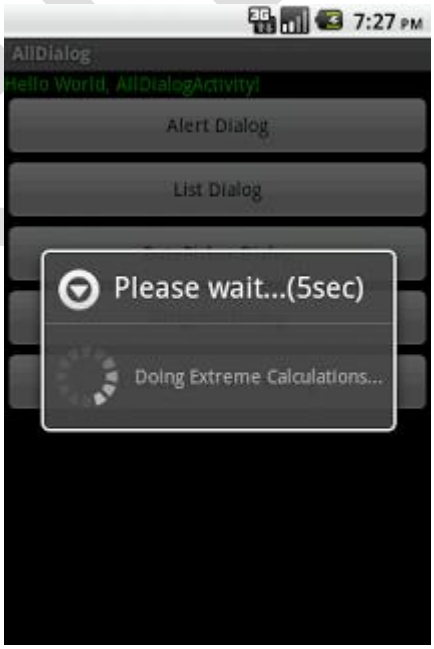
Types of Dialogs :

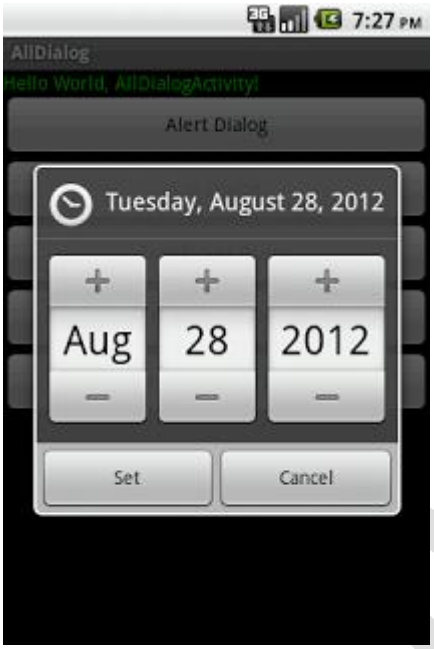
Dialog: Base class for all dialogs. It is usually used to inform user and is the simplest form of a dialog.
AlertDialog: This dialog contains a number of buttons as shown in figure. It is used to get user confirmation

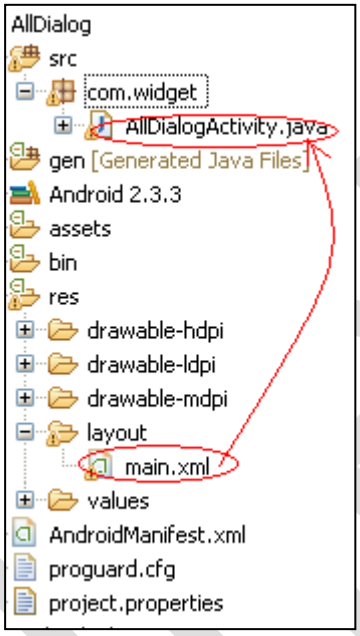
on some specific operations like delete a file.
CharacterPickerDialog: It is used to pick an accented character that is associated with base character.
DatePickerDialog: It contains DatePicker control. It is used to collect data from user.
ProgressDialog: It contains a ProgressBar control. It is used to update the status of an operation. For example, file transfer is in progress.
TimePickerDialog: It contains a TimePicker.

Example : Develop Android app to display all dialogs ?









main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
        <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
```

```
android:id="@+id/tvShow"/>
<Button android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Alert Dialog"
android:id="@+id/btnAlert"/>

<Button android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="List Dialog"
android:id="@+id/btnList"/>

<Button android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="DatePicker Dialog"
android:id="@+id/btnDtPicker" />

<Button android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Progress Dialog"
android:id="@+id/btnProgress"/>

<Button android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Character Picker Dialog"
android:id="@+id/btnChPicker"/>
        </LinearLayout>
```

AllDialogActivity.java

```
public class AllDialogActivity extends Activity {
        /** Called when the activity is first created. */
        TextView tv;
        Button btnAlert, btnList, prgDilog = null, btnDtPicker, btnChButton;
```

```
	int year, month, day;
	CharacterPickerDialog cpd;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		tv = (TextView) findViewById(R.id.tvShow);
		cpd_init();
		/********
		 * Alert Dialog Demo
		 **********/
		btnAlert = (Button) findViewById(R.id.btnAlert);
		btnAlert.setOnClickListener(new View.OnClickListener() {
			public void onClick(View v) {
				// TODO Auto-generated method stub
				AlertDialog.Builder builder = new AlertDialog.Builder(
						AllDialogActivity.this);
				builder.setMessage("Are you Sure you want to Exit?");
				builder.setCancelable(false);
				builder.setPositiveButton("Yes",
						new DialogInterface.OnClickListener() {
				public void onClick(DialogInterface dialog, int which) {
				// TODO Auto-generated method stub
				AllDialogActivity.this.finish();
							}
						});
				builder.setNegativeButton("No",
						new DialogInterface.OnClickListener() {
							public void onClick(DialogInterface dialog,	int
which) {
```

```
                                        // TODO Auto-generated method stub
                                        dialog.cancel();
                                                }
                                        });
                                AlertDialog alert = builder.create();
                                alert.show();
                        }
                });
                /********
                 * List Dialog Demo
                 **********/
                btnList = (Button) findViewById(R.id.btnList);
                btnList.setOnClickListener(new View.OnClickListener() {

                        public void onClick(View v) {
                                // TODO Auto-generated method stub
                                AlertDialog.Builder builder = new AlertDialog.Builder(
                                                AllDialogActivity.this);

                                CharSequence[] items = { "Red", "Green", "Blue" };
                                builder.setTitle("Chose Color..");
                                builder.setItems(items, new DialogInterface.OnClickListener() {

                                        public void onClick(DialogInterface dialog, int which) {
                                                // TODO Auto-generated method stub
                                                switch (which) {
                                                case 0:
                                                        tv.setTextColor(Color.RED);
                                                        break;
                                                case 1:
                                                        tv.setTextColor(Color.GREEN);
```

```
                                        break;
                                case 2:
                                        tv.setTextColor(Color.BLUE);
                                        break;
                                }
                        }
                        });
                        AlertDialog alert = builder.create();
                        alert.show();
                }
        });
        /********
         * Progress Dialog Demo
         **********/
        prgDilog = (Button) findViewById(R.id.btnProgress);
        prgDilog.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        // TODO Auto-generated method stub
                        final ProgressDialog myProgressDialog = ProgressDialog.show(
        AllDialogActivity.this, "Please wait...(5sec)",
"Doing Extreme Calculations...", true);
                        new Thread() {
                                public void run() {
                                        try {
                                                // Do some Fake-Work
                                                sleep(5000);
                                        } catch (Exception e) {
                                        }
                                        // Dismiss the Dialog
                                        myProgressDialog.dismiss();
                                }
```

```
                }.start();
            }
        });

        /********
         * DatePicker Dialog Demo
         **********/
        btnDtPicker = (Button) findViewById(R.id.btnDtPicker);
        btnDtPicker.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
                // TODO Auto-generated method stub

                showDialog(0);
            }
        });

        /********
         * Character Picker Dialog Demo
         **********/
        btnChButton = (Button) findViewById(R.id.btnChPicker);
        btnChButton.setOnClickListener(new View.OnClickListener() {

            public void onClick(View arg0) {
                // TODO Auto-generated method stub
                cpd_init();
                cpd.show();
            }
        });

    }
```

```
//*** Overrided for Intialize DatePicker Dialog***/
@Override
protected Dialog onCreateDialog(int id) {
        Calendar c = Calendar.getInstance();
        year = c.get(Calendar.YEAR);
        month = c.get(Calendar.MONTH);
        day = c.get(Calendar.DAY_OF_MONTH);
        // TODO Auto-generated method stub
        switch (id) {
        case 0:
                return new DatePickerDialog(AllDialogActivity.this,
                                datePickerListener, year, month, day);
        }
        return new DatePickerDialog(AllDialogActivity.this, datePickerListener,
                        year, month, day);
}

//**** DatePicker Set Listener which call when we click on Set Button ****//
private DatePickerDialog.OnDateSetListener datePickerListener = new
DatePickerDialog.OnDateSetListener() {

        // when dialog box is closed, below method will be called.
        public void onDateSet(DatePicker view, int selectedYear,
                        int selectedMonth, int selectedDay) {

                year = selectedYear;
                month = selectedMonth;
                day = selectedDay;

                // set selected date into textview
```

```
                        tv.setText(new StringBuilder().append(day).append("-")
                                        .append(month + 1).append("-").append(year).append(" "));

                        // set selected date into datepicker also
                        // dpResult.init(year, month, day, null);

                }
        };

        // Character Picker Dialog initialization
        void cpd_init() {
                EditText et = new EditText(this);
                et.setLayoutParams(new LinearLayout.LayoutParams(-1, -2));
                final String options = "0123456789ABCDEF";
                cpd = new CharacterPickerDialog(this, new View(this), null, options,
                                false) {
                        public void onClick(View v) {
                                tv.setText("????:"+((Button)v).getText().toString());
                                dismiss();
                        }
                };

        }
}
```

10.3. Notifications

11. Other widgets

11.1 Inflatter

Activity_Main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/LinearLayout1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   tools:context="${relativePackage}.${activityClass}"  >

        </LinearLayout>
```

Button.xml

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


</Button>
```

Textview.xml

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


</TextView>
```

Activitymain.Java

```
package com.example.inflatter;

import java.util.zip.Inflater;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

public class MainActivity extends Activity {

        LinearLayout layout;
```

```
		TextView tv;
		Button bt;
		LayoutInflater Inflater;


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

			Inflater=(LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
			bt=(Button) Inflater.inflate(R.layout.button, null);

			layout=(LinearLayout) findViewById(R.id.LinearLayout1);
			layout.addView(bt);
			bt.setOnClickListener(new OnClickListener() {

				@Override
				public void onClick(View v) {
					// TODO Auto-generated method stub
					tv=(TextView) Inflater.inflate(R.layout.textview, null);
					layout.addView(tv);

				}
			});
		}

	}
```

Strings.xml

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

   <string name="app_name">Inflatter</string>
   <string name="hello_world">Hello world!</string>

</resources>
```

11.2 **VerticalScroolview**

Activity_Main.xml

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/ScrollView1"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="${relativePackage}.${activityClass}" >

   <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical" >

      <Button
```

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />

  <Button
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="Button" />
```

```
    <Button
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:text="Button" />

    <Button
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:text="Button" />

    <Button
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:text="Button" />
  </LinearLayout>

        </ScrollView>
        Activitymain.Java
        package com.example.verticalscroolview;

        import android.app.Activity;
        import android.os.Bundle;
        import android.view.Menu;
        import android.view.MenuItem;

        public class MainActivity extends Activity {

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

11.3 Horizentol ScroolView

```
        Activity_Main.xml
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/ScrollView1"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="${relativePackage}.${activityClass}" >

  <HorizontalScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="horizontal" >
```

```xml
<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:text="Button" />

<Button
   android:layout_width="fill_parent"
```

```
        android:layout_height="fill_parent"
        android:text="Button" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="Button" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="Button" />

    <Button
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="Button" />
  </LinearLayout>
</HorizontalScrollView>

    </ScrollView>
```

Activitymain.Java

```
package com.example.horizontalscroolview;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

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

11.4 Nested Layouts

Activity_Main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/LinearLayout1"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="${relativePackage}.${activityClass}" >

  <ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher" />
  <LinearLayout
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Good"/>
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Morning"/>
  </LinearLayout>

</LinearLayout>
```

Activitymain.Java

```
package com.example.nestedlayout;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

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

Nested Layouts1

Activity_Main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/LinearLayout1"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="${relativePackage}.${activityClass}" >

  <ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher" />
  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Good"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Morning"/>
  </LinearLayout>

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <ImageView
    android:id="@+id/imageView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher" />



  </LinearLayout>

</LinearLayout>
```

Activitymain.Java

```
package com.example.nestedlayout1;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

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

10.11 Spinner

Spinner allows you to select an item from a drop down menu

```
<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"
  tools:context="${relativePackage}.${activityClass}" >

  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />

  <Spinner
    android:id="@+id/spinner1"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="16dp" />

</RelativeLayout>
```

```
package com.example.spinner;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

public class SpinnerActivity extends Activity {

        Spinner spin;
        String Course[]={"Android,"Java","PHP","Android","Java","PHP","Android","Java","PHP"};

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

                spin=(Spinner)findViewById(R.id.spinner1);

                ArrayAdapter adp=new ArrayAdapter(getApplicationContext(),
android.R.layout.simple_dropdown_item_1line,Course);
                spin.setAdapter(adp);

        }
}
```

- ✓ Multiautocomplete TextView

MultiAutoCompleteTextView is same as autocompletetextview but there is a major difference between both of them that autocompletetextview can hold or select only single value at single time but MultiAutoCompleteTextView can hold multiple string words value at single time. These all values are separated by comma(,).

```
<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.android_examples.com.multiautocompletetextview.MainActivity" >

            <MultiAutoCompleteTextView
            android:id="@+id/MultiAutoCompleteTextView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="108dp"
            android:ems="10"
            >

            <requestFocus />
            </MultiAutoCompleteTextView>

</RelativeLayout>

            package com.android_examples.com.multiautocompletetextview;
            import android.app.Activity;
            import android.os.Bundle;
            import android.widget.ArrayAdapter;
            import android.widget.MultiAutoCompleteTextView;

            public class MainActivity extends Activity {

             MultiAutoCompleteTextView MultipleValuesholdt;
             String[] MultipleTextStringValue = { "Android","Android-MultiAutoCompleteTextView","Android Top
            Tutorials" };

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

             MultipleValuesholdt =
            (MultiAutoCompleteTextView)findViewById(R.id.MultiAutoCompleteTextView1);
             ArrayAdapter<String> TopicName = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, MultipleTextStringValue);
             MultipleValuesholdt.setAdapter(TopicName);
             MultipleValuesholdt.setThreshold(3);
             MultipleValuesholdt.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
             }

}
```

Other

Adapter

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter can be used to supply the data to like spinner, list view, grid view etc.

- ✓ ListView

Android ListView is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.

activity_list.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"
   tools:context="${relativePackage}.${activityClass}"
   android:background="#ccddaa">

   <TextView
      android:id="@+id/textView1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="ListView Demo"
      android:gravity="center" />

   <ListView
      android:id="@+id/listView1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_below="@+id/textView1"
      android:layout_marginTop="16dp" >
   </ListView>

</RelativeLayout>
```

Custom.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:textStyle="bold"
   android:textColor="#335566"
   android:textSize="34sp">


</TextView>
```

ListActivity.java

```
package com.example.listview;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class ListActivity extends Activity {
```

```
        ListView list;
        String
Course[]={"Android","2","PHP","Android","Java","PHP","Android","Java","PHP","Android","Java","PHP","Andro
id","Java","PHP","Android","Java","PHP",

        "Android","Java","PHP","Android","Java","PHP","Android","Java","PHP","Android","Java","PHP","And
roid","Java","PHP","Android","Java","PHP",
                        "Android","Java","PHP","Android","Java","PHP","Android","Java","PHP"};

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

                list=(ListView)findViewById(R.id.listView1);

                ArrayAdapter adapter=new ArrayAdapter(getApplicationContext(),
android.R.layout.simple_list_item_checked, Course);
                list.setAdapter(adapter);
                list.setOnItemClickListener(new OnItemClickListener() {

                        @Override
                        public void onItemClick(AdapterView<?> parent, View view,
                                        int position, long id) {
                                // TODO Auto-generated method stub

                                Toast.makeText(getApplicationContext(), "You Clicked"+""+position+""+id,
Toast.LENGTH_LONG).show();

                        }
                });

//              list.setOnClickListener(new OnClickListener() {
//
//                      @Override
//                      public void onClick(View v) {
//                              // TODO Auto-generated method stub
//                              Toast.makeText(getApplicationContext(), "Hiii",
Toast.LENGTH_LONG).show();
//
//                      }
//              });
        }
}
```

- ✓ GridView

GridView shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a ListAdapter

Activity_main.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"
   tools:context="${relativePackage}.${activityClass}" >
```

```
   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/hello_world" />

   <GridView
      android:id="@+id/gridView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_below="@+id/textView1"
      android:numColumns="auto_fit" >
   </GridView>

</RelativeLayout>
```

MainActivity.java

```
package com.example.gridview;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ArrayAdapter;
import android.widget.GridView;

public class MainActivity extends Activity {

	GridView grid;
	String
Course[]={"Androidwwwwwwwwwwww","Java","PHP","Android","Java","PHP","Android","Java","P
HP"};
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		grid=(GridView)findViewById(R.id.gridView1);
		ArrayAdapter adapter=new ArrayAdapter(getApplicationContext(),
android.R.layout.simple_list_item_1, Course);

		grid.setAdapter(adapter);
	}
}
```

12. Services

Android service is a component that is *used to perform operations on the background* such as playing music, handle network transactions, interacting content providers etc. It doesn't has any UI (user interface).

The service runs in the background indefinitely even if application is destroyed.

Moreover, service can be bounded by a component to perform interactivity and inter process communication (IPC).

The android.app.Service is subclass of ContextWrapper class.

Life Cycle of Android Service

There can be two forms of a service.The lifecycle of service can follow two different paths: started or bound.

1. Started
2. Bound

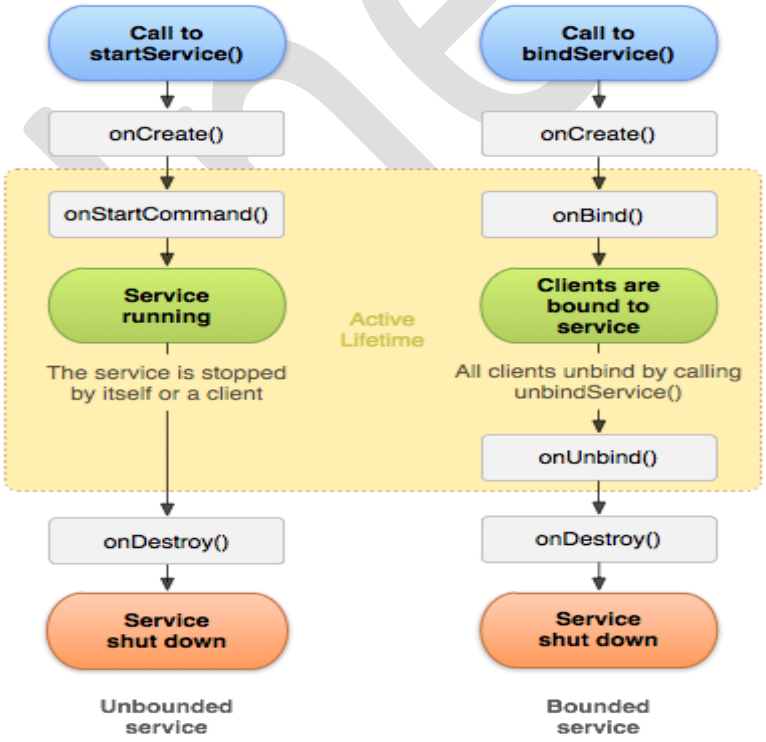
1) Started Service

A service is started when component (like activity) calls **startService()** method, now it runs in the background indefinitely. It is stopped by **stopService()** method. The service can stop itself by calling the **stopSelf()** method.

2) Bound Service

A service is bound when another component (e.g. client) calls **bindService()** method. The client can unbind the service by calling the **unbindService()** method.

The service cannot be stopped until all clients unbind the service.



Understanding Started and Bound Service by background music example

Suppose, I want to play music in the background, so call startService() method. But I want to get information of the current song being played, I will bind the service that provides information about the current song.

Example Program

Actvity_Main.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" >

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="68dp"
    android:text="Start Service" />

  <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/button1"
    android:layout_below="@+id/button1"
    android:layout_marginTop="136dp"
    android:text="Stop service" />

  </RelativeLayout>
```

ActivityMain.Java

```
package com.example.service;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {

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

Button startServiceButton = (Button)findViewById(R.id.button1);
```

```
        startServiceButton.setOnClickListener(new OnClickListener() {


    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        Intent in = new Intent(getApplicationContext(), MyService.class);
      startService(in);

    }
     });
         Button stopServiceButton = (Button)findViewById(R.id.button2);

         stopServiceButton.setOnClickListener(new OnClickListener() {

      @Override
      public void onClick(View v) {
       // TODO Auto-generated method stub
       Intent in = new Intent(getApplicationContext(), MyService.class);
              stopService(in);
      }
     });
        }
    }
```

MyService.Java

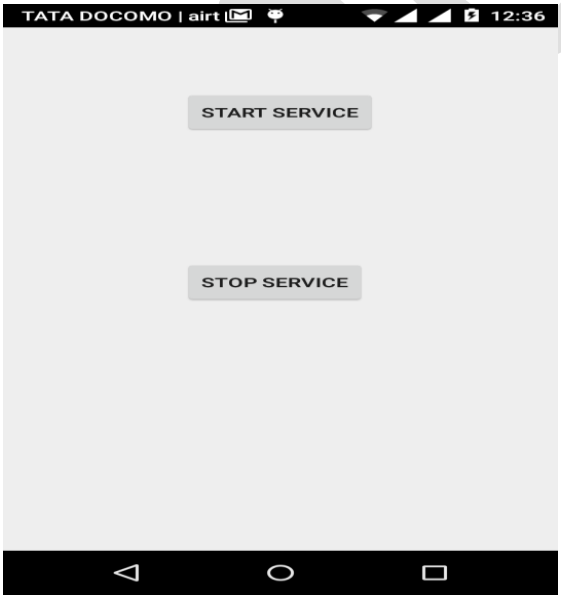
```
package com.example.service;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service{

        @Override
        public IBinder onBind(Intent intent) {
                // TODO Auto-generated method stub
                return null;
        }
        @Override
         public int onStartCommand(Intent intent, int flags, int startId) {
          // TODO Auto-generated method stub
          Toast.makeText(getApplicationContext(), "service started", 3000).show();
          return super.onStartCommand(intent, flags, startId);
         }
         @Override
         public void onDestroy() {
          Toast.makeText(getApplicationContext(), "service stopped", 3000).show();
          super.onDestroy();
         }
```

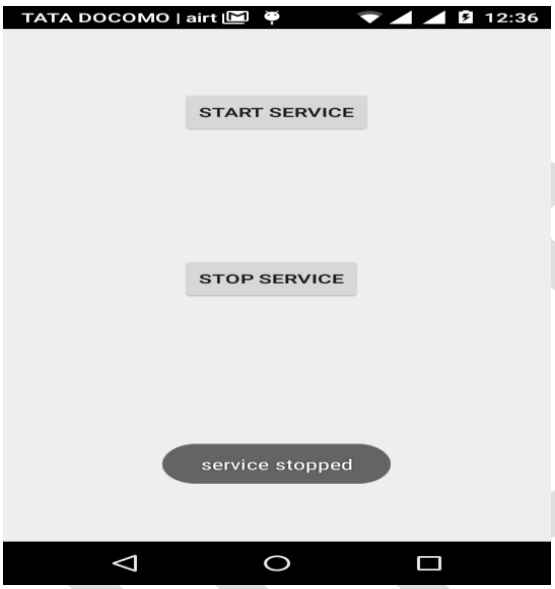
```
}
```

AndroidManifest.XMl

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.service"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="21" />

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

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <service android:name="MyService"></service>
   </application>

</manifest>
```

OutPut





IntentService Example in Android

IntentService is a base class for Services that handle asynchronous requests (expressed as Intents) on demand. Clients send requests through startService(Intent) calls; the service is started as needed, handles each Intent in turn using a worker thread, and stops itself when it runs out of work.

Check output in LogCat.

step1: create a new project.
Step2: create new java class that extends with IntentService class.
step3: Call IntentService in Activity using startService() method.

ProjectName : IntentService
XML Layouts : activity_main.xml
Java Classes: MainActivity, MyService

strings.xml:

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

   <string name="app_name">IntentService</string>
   <string name="action_settings">Settings</string>
   <string name="hello_world">Hello world!</string>
   <string name="button_title">Start Service</string>

</resources>
```

activity_main.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=".MainActivity" >

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/hello_world" />

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/textView1"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="94dp"
      android:onClick="startService"
      android:text="@string/button_title" />

</RelativeLayout>
```

MyService.java:

```
package com.ram.intentservice;

import android.app.IntentService;
import android.content.Intent;
import android.util.Log;

public class MyService extends IntentService {

 public MyService() {
  super("helloservice");
```

```
 // TODO Auto-generated constructor stub
 }

 @Override
 public void onCreate() {
  super.onCreate();
  Log.d("Intent Service", "service created");
 }

 @Override
 protected void onHandleIntent(Intent intent) {
  Log.d("Intent Service", "service started");
  synchronized (intent) {

  }
 }

}
```

MainActivity.java:

```
package com.raj.intentservice;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

public class MainActivity extends Activity {

 @Override

 protected void onCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);

  setContentView(R.layout.activity_main);

 }

 public void startService(View v) {

  Intent in = new Intent(getApplicationContext(), MyService.class);

  startService(in);

 }


}
```

13. Data Storage

13.1. Shared Preferences

Android provides several options for us to save persistent application data. The solution us choose depends on our specific needs, such as whether the data should be private to our application or accessible to other applications (and the user) and how much space our data requires.

User data storage options are the following:

- Shared Preferences
  - Store private primitive data in key-value pairs.
- Internal Storage
- External Storage
- SQLite Databases

Shared Preferences

Preferences are an important part of an Android application. It is important to let the users have the choice to modify and personalize their application depending on their needs.

The SharedPreferences class provides a general framework that allows us to save and retrieve persistent key-value pairs of primitive data types. We can use SharedPreferences to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if our application is killed).

The main importance of preferences is even we comes out the application, the preferences available even for next start of the application also.

To get a SharedPreferences object for user application, use one of two methods:

- getSharedPreferences()
  - Use this if we need multiple preferences files identified by name, which we specify with the first parameter.
- getPreferences()
  - Use this if we need only one preferences file for our Activity. Because this will be the only preferences file for our Activity, we no need to supply a name.

To write values:

4. Call edit() to get a SharedPreferences.Editor.
5. Add values with methods such as putBoolean() and putString().
6. Commit the new values with commit()

To read values, use SharedPreferences methods such as getBoolean() and getString().

Example:

ActvityMain.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"

   tools:context="${relativePackage}.${activityClass}" >
```

```
<TextView
   android:id="@+id/textView1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/hello_world" />
<EditText
   android:id="@+id/editText1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentLeft="true"
   android:layout_below="@+id/textView1"
   android:ems="10" >
   <requestFocus />
</EditText>
<EditText
   android:id="@+id/editText2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentLeft="true"
   android:layout_below="@+id/editText1"
   android:layout_marginLeft="20dp"
   android:ems="10"
   android:inputType="number"/>
<EditText
   android:id="@+id/editText3"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignLeft="@+id/editText2"
   android:layout_below="@+id/editText2"
   android:layout_marginTop="41dp"
```

```
    android:ems="10" />
  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editText3"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="29dp"
    android:layout_toRightOf="@+id/textView1"
    android:text="Button" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.sharedprefences;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity {

        EditText et1,et2,et3;
```

```
Button b;
 public static final String MyPREFERENCES = "MyPrefs";
 public static final String Name = "nameKey";
   public static final String Phone = "phoneKey";
   public static final String mail = "emailKey";

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

        et1=(EditText)findViewById(R.id.editText1);
        et2=(EditText)findViewById(R.id.editText2);
        et3=(EditText)findViewById(R.id.editText3);

        b=(Button)findViewById(R.id.button1);

        b.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                String name=et1.getText().toString();
 String num=et2.getText().toString();
 String Email=et3.getText().toString();


                SharedPreferences preferences = getSharedPreferences(MyPREFERENCES,
Context.MODE_PRIVATE);
 SharedPreferences.Editor editor = preferences.edit();
```

```
            editor.putString(Name, name);
            editor.putString(Phone, num);
            editor.putString(mail, Email);

            editor.commit();

            Intent in=new Intent(MainActivity.this, SecondActivity.class);
            startActivity(in);

                                }
           });

           }
}
```

activity_second <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"
   tools:context="${relativePackage}.${activityClass}" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/hello_world"
      android:id="@+id/tv1" />

</RelativeLayout>
```

SecondActvity.Java

```
package com.example.sharedprefences;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class SecondActivity extends Activity {

        public static final String MyPREFERENCES = "MyPrefs";
         public static final String Name = "";
           public static final String Phone = "";
           public static final String mail = "";

           TextView t;
           String n="",p="",e="";

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

                SharedPreferences preferences = getSharedPreferences(MyPREFERENCES, Context.MODE_PRIVATE);

                n=preferences.getString("nameKey", null);
                p=preferences.getString("phoneKey", null);
                e=preferences.getString("emailKey", null);
```

```
            System.out.println("Vales"+p);


            t=(TextView) findViewById(R.id.tv1);


            t.setText("HIIIIIII"+n+p+e);


        }
}
```

File System in Android

The java.io.File class abstracts a filename and directory pathnames. It provides useful methods to perform some basic file system operations such as, verifying the existence of a file, getting the absolute or relative path of a file, verifying if a path refers to a directory or file, verifying the permissions on a File, checking the file length, create/remove folders, etc.

Example : Display List of Files & Directories exists in SD Card Location

```
public class FilesDirectoryListActivity extends ListActivity {
 private List<String> fileList = new ArrayList<String>();
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      File root = new File(Environment.getExternalStorageDirectory().getAbsolutePath());
      ListDir(root);
   }
   void ListDir(File f){
                  File[] files = f.listFiles();
            fileList.clear();
                  for (File file : files){
             fileList.add(file.getPath());
                  }
    ArrayAdapter<String> directoryList = new ArrayAdapter<String>(this,android.R.laust.simple_list_item_1,
fileList);
    setListAdapter(directoryList);
   }
```

```
}
```

Internal Storage

Internal Storage means it will store in the phone memory. Internal Storage remain until the application is available, if we uninstall the application the memory will be lost.

Example : Develop Android application to write given settings data & to Read the written setting data from android internal storage ?

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLaust xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:laust_width="fill_parent"
   android:laust_height="fill_parent"   >
<TextView
   android:laust_width="fill_parent"
   android:laust_height="wrap_content"
   android:text="Enter File Name" />
<EditText
   android:id="@+id/filename"
   android:laust_width="fill_parent"
   android:laust_height="wrap_content" />
<TextView
   android:laust_width="fill_parent"
   android:laust_height="wrap_content"
   android:text="Enter Content" />
<EditText
   android:id="@+id/content"
   android:laust_width="fill_parent"
   android:laust_height="wrap_content" />
<Button
   android:id="@+id/save"
   android:laust_width="fill_parent"
   android:laust_height="wrap_content"
   android:text="Save"/>
```

```
<Button
   android:id="@+id/load"
   android:laust_width="fill_parent"
   android:laust_height="wrap_content"
   android:text="Load"/>
<TextView
   android:id="@+id/data"
   android:laust_width="fill_parent"
   android:laust_height="wrap_content"
   android:text="Usr Data :" />
</LinearLaust>
```

```
public class AndroidInternalStorageActivity extends Activity implements OnClickListener {
    EditText edFileName, edContent;
    Button btnSave,btnLoad;
    TextView tv;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.laust.main);
      edFileName = (EditText)findViewById(R.id.filename);
      edContent = (EditText)findViewById(R.id.content);
         btnSave = (Button)findViewById(R.id.save);
         btnLoad = (Button)findViewById(R.id.load);
         tv = (TextView)findViewById(R.id.data);
         btnSave.setOnClickListener(this);
         btnLoad.setOnClickListener(this);
     }
   @Override
   public void onClick(View arg0) {
```

```
		if(arg0.getId()==R.id.save){
		try{
WriteSettings(this,edContent.getText().toString(),edFileName.getText().toString());
		}catch(Exception e){
			System.out.println(e);
		}
		}else {
			try{
			  String s=ReadSettings(this,edFileName.getText().toString());
			  tv.setText(s);
			}catch(Exception e){
			   System.out.println(e);
			}
		}
	}
	public void WriteSettings(Context context, String data, String file) throws IOException {
		FileOutputStream fos= null;
		OutputStreamWriter osw = null;
		fos= context.openFileOutput(file,Context.MODE_PRIVATE);
		osw = new OutputStreamWriter(fos);
		osw.write(data);
		Toast t=Toast.makeText(this, "Content Saved Successfully in to Internal Memory", 1000);
		t.show();
		osw.close();
		fos.close();
		}
		public String ReadSettings(Context context, String file) throws IOException {
			FileInputStream fis = null;
			InputStreamReader isr = null;
			String data = null;
			fis = context.openFileInput(file);
```

```
                isr = new InputStreamReader(fis);
                char[] inputBuffer = new char[fis.available()];
                isr.read(inputBuffer);
                data = new String(inputBuffer);
                isr.close();
                fis.close();
                return data;
            }
        }
```

To see the storage location of the files open FileExplorer window select “data/data/packagename/filename”

External-SD Card.

- 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. Also, files on the external storage card will remain there even after the application that wrote the files is uninstalled. These limitations can compromise sensitive information written out to storage or allow attackers to inject malicious data into the program by modifying an external file it relies on.
- The class android.os.Environment provides access to environment variables.
  The method getExternalStorageDirectory() can be used to get the Android external storage directory. This directory may not currently be accessible if it has been mounted by the user on their computer, has been removed from the device, or some other problem has happened. Us can determine its current state with getExternalStorageState().
- Building apps for Android devices often includes the need to store large amounts of information on the device. Since many devices have limited internal storage for apps, it is recommended to use External Storage.

To get started us need to add a uses permission into the AndroidManifest.xml file for our app. The permission us need to add is shown below:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
```

Example : \Develop android application to write res/raw/text file data into External SD Card ?

Save a text file into “ res/raw/textfile.txt “ and run the following code.

```
public class WriteSDCard extends Activity {
   private TextView tv;
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

```
    setContentView(R.laust.main);
    tv = (TextView) findViewById(R.id.TextView01);
    checkExternalMedia();
    writeToSDFile();
    readRaw();
}
/** Method to check whether external media available and writable. */
private void checkExternalMedia(){
    boolean mExternalStorageAvailable = false;
    boolean mExternalStorageWriteable = false;
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        // Can read and write the media
        mExternalStorageAvailable = mExternalStorageWriteable = true;
    } else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        // Can only read the media
        mExternalStorageAvailable = true;
        mExternalStorageWriteable = false;
    } else {
        // Can't read or write
        mExternalStorageAvailable = mExternalStorageWriteable = false;
    }
    tv.append("\n\nExternal Media: readable="
                +mExternalStorageAvailable+" writable="+mExternalStorageWriteable);
}
/** Method to write ascii text characters to file on SD card. Note that us must add a
   WRITE_EXTERNAL_STORAGE permission to the manifest file or this method will throw
   a FileNotFound Exception because us won't have write permission. */
   private void writeToSDFile(){
         // Find the root of the external storage.
    File root = android.os.Environment.getExternalStorageDirectory();
```

```
    tv.append("\nExternal file system root: "+root);
    File dir = new File (root.getAbsolutePath() + "/download");
    dir.mkdirs();
    File file = new File(dir, "myData.txt");
    try {
        FileOutputStream f = new FileOutputStream(file);
        PrintWriter pw = new PrintWriter(f);
        pw.println("This is First Line.");
        pw.println("Here is a second line.");
        pw.flush();
        pw.close();
        f.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
        tv.append("\n\nFile written to "+file);
}
        /** Method to read in a text file placed in the res/raw directory of the application. The
          method reads in all lines of the file sequentially. */
        private void readRaw(){
                tv.append("\nData read from res/raw/textfile.txt:");
                InputStream is = this.getResources().openRawResource(R.raw.textfile);
                InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr, 8192);    // 2nd arg is buffer size

                // More efficient (less readable) implementation of above is the composite expression
                /*BufferedReader br = new BufferedReader(new InputStreamReader(
                                this.getResources().openRawResource(R.raw.textfile)), 8192);*/
```

```
                try {
                        String test;
                        while (true){
                                test = br.readLine();
                                // readLine() returns null if no more lines in the file
                                if(test == null) break;
                                tv.append("\n"+"    "+test);
                        }
                        isr.close();
                        is.close();
                        br.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                tv.append("\n\nThat is all");
        }
}
```

14. SQLITE Database

14.1. introduction about SQLITE

What is SQLite ?

- ✓ SQLite is an Open Source Database which is embedded into Android.
- ✓ SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. In addition it requires only little memory at runtime (approx. 250 KByte).

DataTypes in SQLite:

- ✓ SQLite supports the data types TEXT (similar to String in Java), INTEGER (similar to long in Java) and REAL (similar to double in Java).
- ✓ All other java types must be converted into one of these fields before saving them in the database.

How SQLite is in Android ?

SQLite is available on every Android device.

SQLite database in Android does not require any database setup or administration.

We only need to define the SQL statements for creating and updating the database, then the Android Platform automatically manages the database to use by us.

If your application creates a database, this database is saved in the directory DATA/data/APP_NAME/databases/FILENAME.

The parts of the above directory are constructed based on the following rules.

a) DATA is the path which the Environment.getDataDirectory() method returns.
b) APP_NAME is our application name.
c) FILENAME is the name what we specified in our application code for the database.

How to interact with SQLite?

To Interact with SQLiteDB the following things we should know before going to use, are

i) Its Packages :
a) android.database.*; contains all general classes for working with databases.
b) android.database.sqlite.*; contains the SQLite specific classes.

ii) SQLiteOpenHelper:

- ✓ To create and upgrade a database in our Android application usually we should use subclass of SQLiteOpenHelper.
- ✓ In the constructor of our subclass call the super() method of SQLiteOpenHelper, by specifying the database name and the current database version.
- ✓ For Example : Refer API for SQLiteOpenHelper class.

```
public class MySQLiteHelper extends SQLiteOpenHelper{

        public MySQLiteHelper(Context context) {

                super(context, “Contact Mgt”, null, 1);

        }

}
```

- ✓ In this class we need to override the onCreate() and onUpgrade() methods.
- ✓ onCreate() is called by the framework, if the database does not exists.
- ✓ onUpgrade() is called, if the database version is increased in our application code. This method allows us to update the database schema.

Sample Code:

```
@Override

        public void onCreate(SQLiteDatabase db) {

                db.execSQL(“CREATE TABLE contacts(id INTEGER PRIMARY KEY,name
TEXT,phone_number TEXT)”);

        }

        // Upgrading database

        @Override

        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

                // Drop older table if existed

                db.execSQL("DROP TABLE IF EXISTS contacts");

                // Create tables again
```

```
        onCreate(db);

    }
```

- ✓ Both methods receive an SQLiteDatabase object as parameter which represents the database.
- ✓ SQLiteOpenHelper provides the methods getReadableDatabase() and getWriteableDatabase() to get access to an SQLiteDatabase object; either in read or write mode.
  Note : The best practice in realtime is create a separate class per table.

iii) SQLiteDatabase :

- ✓ SQLiteDatabase is the base class for working with a SQLite database in Android and provides methods to open, query, update and close the database.
- ✓ For queries are specifically SQLiteDatabase provides the insert(), update() and delete() methods.
- ✓ This class also provides the execSQL() method, which allows to execute SQL queries directly.

iv) ContentValues :

- ✓ Defined in android.content.*; package.
- ✓ The object ContentValues allows to define key/values. The "key" represents the table column identifier and the "value" represents the content for the table record in this column.
- ✓ ContentValues can be used for inserts and updates of database entries.
- ✓ Queries can be created via the rawQuery() and query() methods or via the SQLiteQueryBuilder class.
  What is rawQuery() and query() methods :

rawQuery() directly accepts an SQL statement as input, where as query() method provides a structured interface for specifying the SQL query.

Sample Code :

```
String selectQuery = "SELECT  * FROM Contacts";

SQLiteDatabase db = this.getWritableDatabase();

Cursor cursor = db.rawQuery(selectQuery, null);
```

Sample Code :

```
SQLiteDatabase db = super.getReadableDatabase();

Cursor cursor = db.query("Contacts", new String[] { KEY_ID,

                                KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",

                                new String[] { String.valueOf(id) }, new String[]{"KYE_NAME"}, null,
null, null);
```

SQLiteQueryBuilder is a convenience class that helps to build SQL queries.

v) Cursor :

- ✓ A query returns a Cursor object.
- ✓ A Cursor represents the result of a query and basically points to one row of the query result.
- ✓ Using this Cursor object we can navigate list of DB records.
- ✓ To get the number of elements of the resulting query use the getCount() method.
- ✓ To move between individual data rows, we can use the moveToFirst() and moveToNext() methods. The isAfterLast() method allows to check if the end of the query result has been reached.
  Refer Cursor API methods .

Cursor provides typed getXXXX() methods, e.g. getLong(columnIndex), getString(columnIndex) to access the column data for the current position of the result. The "columnIndex" is the number of the column you are accessing.

Cursor also provides the getColumnIndexOrThrow(String) method which allows to get the column index for a column name of the table.

vi) Listviews, ListActivity and Simple Cursor Adapters :

ListViews are Views which allow to display a list of elements.

ListActivities are specialized Activities which make the usage of ListViews easier.

To work with databases and ListViews we can use the SimpleCursorAdapter.

The SimpleCursorAdapter allows to set a layout for each row of the ListViews.

We can also define an array which contains the column names and another array which contains the IDs of Views which should be filled with the data.

The SimpleCursorAdapter class will map the columns to the Views based on the Cursor passed to it.

To obtain the Cursor we should use the Loader class.

Sample Code :

```
1. public List<Contact> getAllContacts() {
2. List<Contact> contactList = new ArrayList<Contact>();
3. SQLiteDatabase db = this.getWritableDatabase();
4. //Getting all contacts.
5. Cursor cursor = db.rawQuery("SELECT * FROM contacts",null);
6. // looping through all rows and adding to list
7. if (cursor.moveToFirst()) {
8. do {
9. Contact contact = new Contact();
10. contact.setID(Integer.parseInt(cursor.getString(0)));
11. contact.setName(cursor.getString(1));
12. contact.setPhoneNumber(cursor.getString(2));
13. // Adding contact to list
14. contactList.add(contact);
15. } while (cursor.moveToNext());
16. }
17. // return contact list
18. return contactList;
19. }
```

Creating SQLite Database

db = this.openOrCreateDatabase(SAMPLE_DB_NAME, MODE_PRIVATE, null);

Note : While reading data from the Database we should open the database in readable mode, while inserting record we should open database in writable mode.

For Ex:

```
SQLiteDatabase db = this.getWritableDatabase(); // Write Mode

SQLiteDatabase db = super.getReadableDatabase();// Readable Mode
```

Creating Table :

```
1. @Override
2. public void onCreate(SQLiteDatabase db) {
3. db.execSQL("CREATE TABLE contacts(id INTEGER PRIMARY KEY,name TEXT,phone_number TEXT)");
4. }
5. // Upgrading database
6. @Override
7. public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
8. // Drop older table if existed
9. db.execSQL("DROP TABLE IF EXISTS contacts");
10. // Create tables again
11. onCreate(db);
12. }
```

(or)

```
sampledb.execSQL("CREATE TABLE IF NOT EXISTS Contacts(LastName VARCHAR, FirstName
VARCHAR,Country VARCHAR, Age INT(3));");
```

Closing the database

```
db.close();
```

Dropping Database

```
db.execSQL("DROP TABLE IF EXISTS contacts");
```

DML Operations on Data

To Add a new Record, 1st create ContentValue object, and put the values as key, value pair and insert the record using insert() method.

```
1. // Adding new contact
2. void addContact(Contact contact) {
3. SQLiteDatabase db = this.getWritableDatabase();
4. ContentValues values = new ContentValues();
5. values.put(KEY_NAME, contact.getName()); // Contact Name
6. values.put(KEY_PH_NO, contact.getPhoneNumber()); // Contact Phone
7. // Inserting Row
8. db.insert("contacts", null, values);
9. db.close(); // Closing database connection
10. }
```

(or)

```
sampleDB.execSQL("INSERT INTO contacts values ('Makam','Sai Geetha','India',25);");
```

To delete a record invoke delete() method.

```
// Deleting single contact

1. public void deleteContact(Contact contact) {
2. SQLiteDatabase db = this.getWritableDatabase();
3. db.delete("Contacts", KEY_ID + " = ?",
4. new String[] { String.valueOf(contact.getID()) });
5. db.close();
6. }
```

To update a record, define ContentValue object, put the values, invoke update() method, will return int value means count of the record.

```
// Updating single contact

1. public int updateContact(Contact contact) {
2. SQLiteDatabase db = this.getWritableDatabase();
3. ContentValues values = new ContentValues();
4. values.put(KEY_NAME, contact.getName());
5. values.put(KEY_PH_NO, contact.getPhoneNumber());
6. // updating row
7. return db.update("contacts", values, KEY_ID + " = ?",
8. new String[] { String.valueOf(contact.getID()) });
9. }
```

Querying the database.

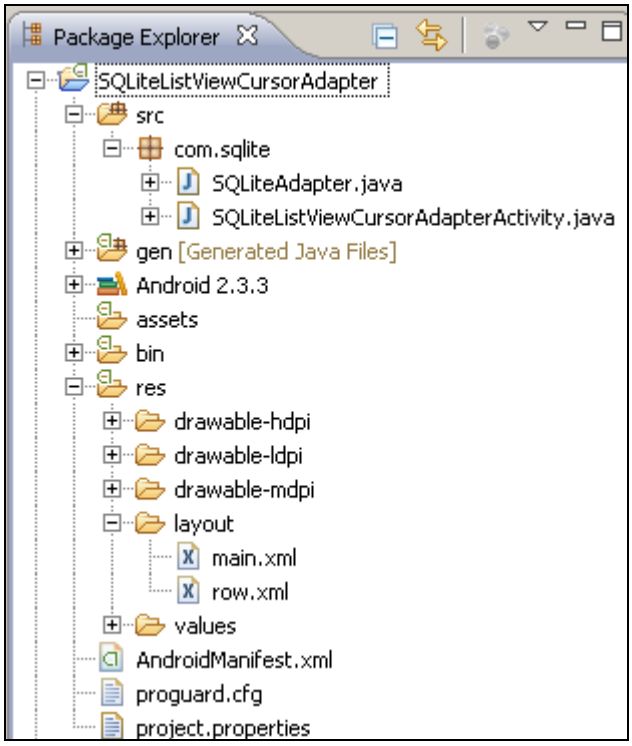
To Select a record 1st invoke query() method, it returns Cursor object, if Cursor object not equals null means if requested record exist, then get data from Cursor using getXXX() method, and return Contact object.

```
// Selecting a record

1. Contact getContact(int id) {
2. SQLiteDatabase db = super.getReadableDatabase();
3. Cursor cursor = db.query("contacts", new String[] { KEY_ID,
4. KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
5. new String[] { String.valueOf(id) }, null, null, null, null);
6. if (cursor != null)
7. cursor.moveToFirst();
8. Contact contact = new Contact(Integer.parseInt(cursor.getString(0)),
9. cursor.getString(1), cursor.getString(2));
10. return contact;
11. }
```

(or)

```
1. Cursor c = sampleDB.rawQuery("SELECT FirstName, Age FROM contacts where Age > 10 LIMIT 5",
   null);
2. if (c != null ) {
3. if  (c.moveToFirst()) {
4. do {
5. String firstName = c.getString(c.getColumnIndex("FirstName"));
6. int age = c.getInt(c.getColumnIndex("Age"));
7. results.add("" + firstName + ",Age: " + age);
8. }while (c.moveToNext());
9. }
10. }
```

Example : Develop an android project to store data into SQLite Database, retrieve and display the data on ListView using CursorAdapter?



Step-1) Create a new Android Project .

Step-2) Define UI in layout xml file.

main.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3. android:orientation="vertical"
4. android:layout_width="fill_parent"
5. android:layout_height="fill_parent">
6. <ListView
7. android:id="@+id/contentlist"
8. android:layout_width="fill_parent"
9. android:layout_height="fill_parent"/>
10. </LinearLayout>
```

Step-3) Define another UI layout xml file, under layout file with textview as root element, to display the text content.

row.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <TextView xmlns:android="http://schemas.android.com/apk/res/android"
3. android:id="@+id/text"
4. android:layout_width="fill_parent"
5. android:layout_height="fill_parent"
6. android:padding="10dip"/>
```

Step-4) Define Sub class for SQLiteHelper class, to connect to Database, and to perform CRUD operations towards Database.

SQLiteAdapter.java

```
1.  package com.sqlite;
2.  import android.content.ContentValues;
3.  import android.content.Context;
4.  import android.database.Cursor;
5.  import android.database.sqlite.SQLiteDatabase;
6.  import android.database.sqlite.SQLiteOpenHelper;
7.  import android.database.sqlite.SQLiteDatabase.CursorFactory;
8.  public class SQLiteAdapter extends SQLiteOpenHelper{
9.  public static final String MYDATABASE_NAME = "MY_DATABASE";
10. public static final String MYDATABASE_TABLE = "MY_TABLE";
11. public static final int MYDATABASE_VERSION = 1;
12. public static final String KEY_ID = "_id";
13. public static final String KEY_CONTENT = "Content";
14. //create table MY_DATABASE (ID integer primary key, Content text not null);
15. private static final String SCRIPT_CREATE_DATABASE =
16. "create table " + MYDATABASE_TABLE + " ("
17. + KEY_ID + " integer primary key autoincrement, "
18. + KEY_CONTENT + " text not null);";
19. private SQLiteDatabase sqLiteDatabase;
20. public SQLiteAdapter(Context ctx){
21. super(ctx,MYDATABASE_NAME,null,MYDATABASE_VERSION);
22. }
23. public SQLiteAdapter(Context context, String name,
24. CursorFactory factory, int version) {
25. super(context, name, factory, version);
26. }
27. @Override
28. public void onCreate(SQLiteDatabase db) {
29. // TODO Auto-generated method stub
30. db.execSQL(SCRIPT_CREATE_DATABASE);
31. }
32. @Override
33. public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
34. // TODO Auto-generated method stub
35. }
36. public void close(){
37. super.close();
38. }
39. public long insert(String content){
40. sqLiteDatabase = this.getWritableDatabase();
41. ContentValues contentValues = new ContentValues();
42. contentValues.put(KEY_CONTENT, content);
43. return sqLiteDatabase.insert(MYDATABASE_TABLE, null, contentValues);
44. }
45. public int deleteAll(){
46. sqLiteDatabase = this.getWritableDatabase();
47. return sqLiteDatabase.delete(MYDATABASE_TABLE, null, null);
48. }
49. public Cursor queueAll(){
50. sqLiteDatabase = super.getReadableDatabase();
51. String[] columns = new String[]{KEY_ID, KEY_CONTENT};
52. Cursor cursor = sqLiteDatabase.query(MYDATABASE_TABLE,columns,null,null,null,null,null);
```

```
53. return cursor;
54. }
55. }
```

Step-5) Define Activity class

SQLiteListViewCursorAdapterActivity.java

```
1.  package com.sqlite;
2.  import android.app.Activity;
3.  import android.database.Cursor;
4.  import android.os.Bundle;
5.  import android.widget.ListView;
6.  import android.widget.SimpleCursorAdapter;
7.  public class SQLiteListViewCursorAdapterActivity extends Activity {
8.  private SQLiteAdapter mySQLiteAdapter;
9.  /** Called when the activity is first created. */
10. @Override
11. public void onCreate(Bundle savedInstanceState) {
12. super.onCreate(savedInstanceState);
13. setContentView(R.layout.main);
14. ListView listContent = (ListView)findViewById(R.id.contentlist);
15. /*Create/Open a SQLite database and fill with dummy content and close it*/
16. mySQLiteAdapter = new SQLiteAdapter(this);
17. mySQLiteAdapter.deleteAll();
18. mySQLiteAdapter.insert("A for Android");
19. mySQLiteAdapter.insert("B for Boy");
20. mySQLiteAdapter.insert("C for Cat");
21. mySQLiteAdapter.insert("D for Dog");
22. mySQLiteAdapter.insert("E for Egg");
23. mySQLiteAdapter.insert("F for Fish");
24. mySQLiteAdapter.insert("G for Girl");
25. mySQLiteAdapter.insert("H for Hat");
26. mySQLiteAdapter.insert("I for Ice-scream");
27. mySQLiteAdapter.insert("J for Java");
28. mySQLiteAdapter.insert("K for Kite");
29. mySQLiteAdapter.insert("L for Lamp");
30. mySQLiteAdapter.insert("M for Raj");
31. mySQLiteAdapter.insert("N for Nose");
32. mySQLiteAdapter.insert("O for Orange");
33. mySQLiteAdapter.insert("P for Pen");
34. mySQLiteAdapter.insert("Q for Queen");
35. mySQLiteAdapter.insert("R for Rain");
36. mySQLiteAdapter.insert("S for Sathya");
37. mySQLiteAdapter.insert("T for Tree");
38. mySQLiteAdapter.insert("U for Umbrella");
39. mySQLiteAdapter.insert("V for Van");
40. mySQLiteAdapter.insert("W for Water");
41. mySQLiteAdapter.insert("X for X'mas");
42. mySQLiteAdapter.insert("Y for Yellow");
43. mySQLiteAdapter.insert("Z for Zoo");
44. mySQLiteAdapter.close();
45. /*
46. Open the same SQLite database and read all it's content.
47. */
```

```
mySQLiteAdapter = new SQLiteAdapter(this);
Cursor cursor = mySQLiteAdapter.queueAll();
startManagingCursor(cursor);
String[] from = new String[]{SQLiteAdapter.KEY_CONTENT};
int[] to = new int[]{R.id.text};
SimpleCursorAdapter cursorAdapter=new SimpleCursorAdapter(this, R.layout.row, cursor, from, to);
listContent.setAdapter(cursorAdapter);
mySQLiteAdapter.close();
}
}
```

14.2. Creating Database

activity_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

</LinearLayout>
```

MainActivity.java:

```
package com.ram.simplesqlitedatabase;

import android.app.Activity;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends Activity {

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

SQLiteDatabase db = openOrCreateDatabase("ramsdb", MODE_PRIVATE, null);

db.execSQL("create table if not exists sampletable(firstname varchar,lastname varchar)");

db.execSQL("insert into sampletable values('raj','shekar')");

Cursor c = db.rawQuery("select * from sampletable", null);
```

```
51.
52. c.moveToFirst();
53.
54. //Getting data using column name
55. String fname = c.getString(c.getColumnIndex("firstname"));
56.
57. //Getting data using column index number
58. String lname = c.getString(1);
59.
60. Toast.makeText(getApplicationContext(), "firstname :"+fname+"\n"+"lastname :"+lname,
    Toast.LENGTH_LONG).show();
61.
62. }
63. }
```

63.1. Working with data CRUD operations

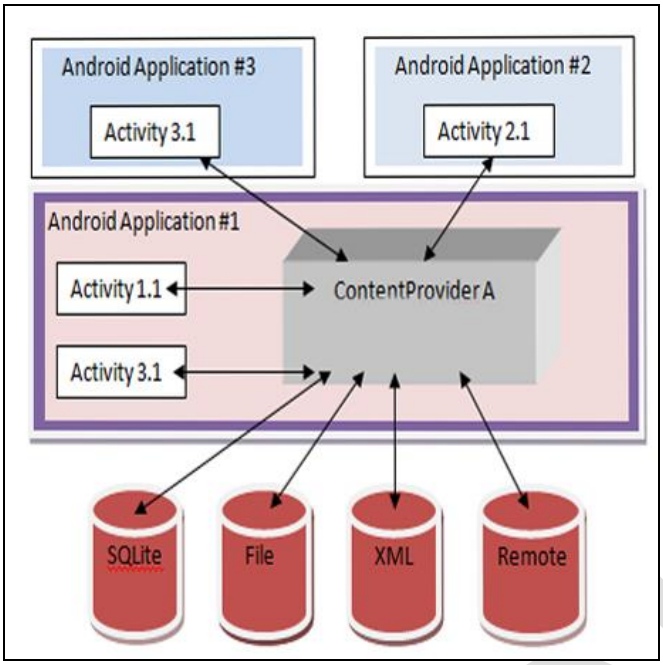
64. Broad cast Receiver
    64.1. Implementation
    64.2. Registering in Manifest

65. Content Providers
    65.1. Creating Content Providers

Content Providers

What is Android Content provider ? Purpose of Content Provider ?

- ✓ An SQLite database is private to the application which creates it.
- ✓ ContentProviders are used to share data among the different apps. i.e If you don't need to share data among multiple applications we can use a database directly via SQLiteDatabase, otherwise we can use ContentProviders.
- ✓ A ContentProvider allows applications to access data. In most cases this data is stored in an SQlite database.
- ✓ We can use content providers to store or retrieve data of one application from any other application.
- ✓ Many Android datasources, e.g. the contacts, are accessible using ContentProviders in different apps. Typically the implementing classes for a ContentProviders provide public constants for the URI's.



Android default content providers ?

There are many built-in content providers supplied by Android OS. They are defined in the android.provider.*; package, they are:

- Browser.
- Calllog.
- Contacts.
- Media Store.
- Settings.

Content Provider	Intended Data
Browser	Browser bookmarks, browser history, etc.
CallLog	Missed calls, call details, etc.
Contacts	Contact details
MediaStore	Media files such as audio, video and images
Settings	Device settings and preferences

Content Provier Uri :

Any content provider is invoked by a URi in the form of content://provider_name . for example the URI of the Contacts content provider that retrieves all contacts is in the following form content://contacts/people. If we want to retrieve a particular contact (by its ID) then it would be in this form: content://contacts/people/5.

Own ContentProvider :

To create our own ContentProvider we should define a class which extends android.content.ContentProvider. Once we define user content provider we should configure in the "AndroidManifest.xml" file and the entry must be with android:authorities attribute, to identify the ContentProvider by the Android system. This authority is the basis for the URI to access data and must be unique.

```
<provider

    android:authorities="package name of content provider"

    android:name=".MyContentProvider" >

</provider>
```

Own ContentProvider must implement several methods, e.g. query(), insert(), update(), delete(), getType() and onCreate(). *In case if you don't want support certain methods its good practice to throw an UnsupportedOperationException().*

The query() method must return a Cursor object.

Security :

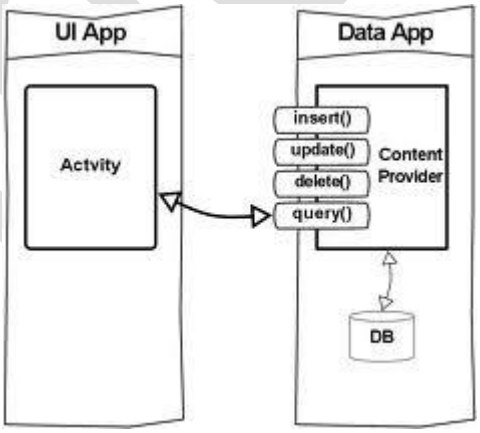
By default a ContentProvider will be available to other programs. If we want to use our ContentProvider only with in the application use android:exported=false attribute in the definition of ContentProvider in the AndroidManifest.xml.

Thread Safety :

Multiple threads can access Content provider concurrently to write data i.e the ContentProvider can be accessed from several programs at the same time.

Set the attribute for android:multiprocess=true in "AndroidManifest.xml" file. This permits an instance of the provider to be created in each client process.

Content Providers are a simple interface with the standard insert(), update(), delete() and query() methods. These methods look like standard database methods, so these are used to implement a content provider as a proxy to the database.



Content Providers feature full permission control and are accessed using a simple URI model, any application with the appropriate permissions can add, remove, and update data from any other applications.

In this example we will demonstrate how to query the contacts content provider to get all the contacts stored.

To query a content provider, we should specify the query string in the form of a URI, with an optional specifier for a particular row. The format of the query URI is as follows:

<standard_prefix>://<authority>/<data_path>/<id>

The various parts of the URI are as follows:

The standard prefix for content providers is always content://

The authority specifies the name of the content provider.

The data_path specifies the kind of data requested.

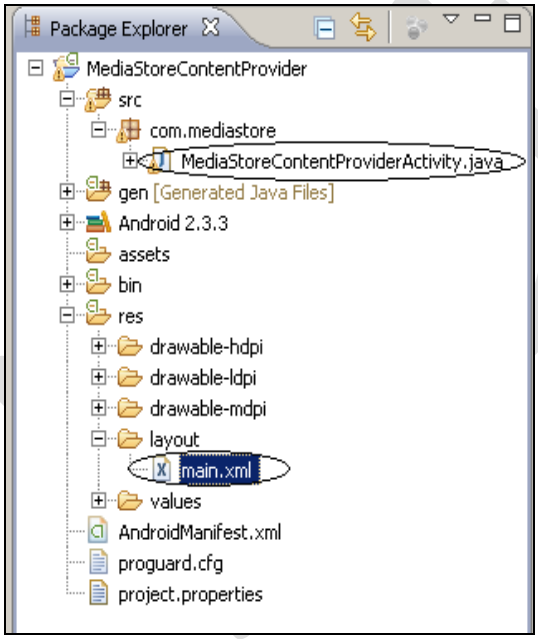
For example, if we are getting all the contacts from the Contacts content provider, then the data path would be people, and the URI look like this: content://contacts/people.

The id specifies the specific record requested. For example, if we are looking for contact number 2 in the Contacts content provider, the URI would look like this: content:// contacts/people/2.

Query String	Description
content://media/internal/images	Returns a list of all the internal images on the device
content://media/external/images	Returns a list of all the images stored on the external storage (e.g., SD card) on the device
content://call_log/calls	Returns a list of all calls registered in the Call Log
content://browser/bookmarks	Returns a list of bookmarks stored in the browser

Working with MediaStore content providers :

Example : Develop android application to retrieves list of audio files exist in the device ?



main.xml

```
<?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" >
        <ListView
   android:id="@+id/PhoneMusicList"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_weight="1"
   android:stackFromBottom="false"
   android:transcriptMode="normal"
  />
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/hello" />
        </LinearLayout>
```

MediaStoreContentProviderActivity.java

```
public class MediaStoreContentProviderActivity extends Activity {
private ArrayList<String> songs = new ArrayList<String>();
private ArrayList<String> albums = new ArrayList<String>();
private ArrayList<String> artists = new ArrayList<String>();
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  Context context = getApplicationContext();
  CharSequence text = "Reading Music...";
  int duration = Toast.LENGTH_LONG;
  Toast toast = Toast.makeText(context, text, duration);
  toast.show();
  String[] projection = new String[] {
       MediaStore.EXTRA_MEDIA_TITLE,
       MediaStore.EXTRA_MEDIA_ARTIST,
```

```
            MediaStore.EXTRA_MEDIA_ALBUM };
        // Get the base URI for the People table in the Contacts content provider.
        Uri media =  MediaStore.getMediaScannerUri();
        // Make the query.
        Cursor mediaCursor = managedQuery(media,
            projection, // Which columns to return
            null,       // Which rows to return (all rows)
            null,       // Selection arguments (none)
            // Put the results in ascending order by name
            MediaStore.EXTRA_MEDIA_TITLE + " ASC");
        if (mediaCursor.moveToFirst()) {
           String title;
           String artist;
           String album;
           int titleColumn = mediaCursor.getColumnIndex(MediaStore.EXTRA_MEDIA_TITLE);
           int artistColumn = mediaCursor.getColumnIndex(MediaStore.EXTRA_MEDIA_ARTIST);
           int albumColumn = mediaCursor.getColumnIndex(MediaStore.EXTRA_MEDIA_ALBUM);
           do {
              // Get the field values
              title = mediaCursor.getString(titleColumn);
              artist = mediaCursor.getString(artistColumn);
              album = mediaCursor.getString(albumColumn);
              artists.add(artist);
              albums.add(album);
              songs.add(title);
           } while (mediaCursor.moveToNext());
        //new MusicGetter().execute();
         }
       }
      }
```

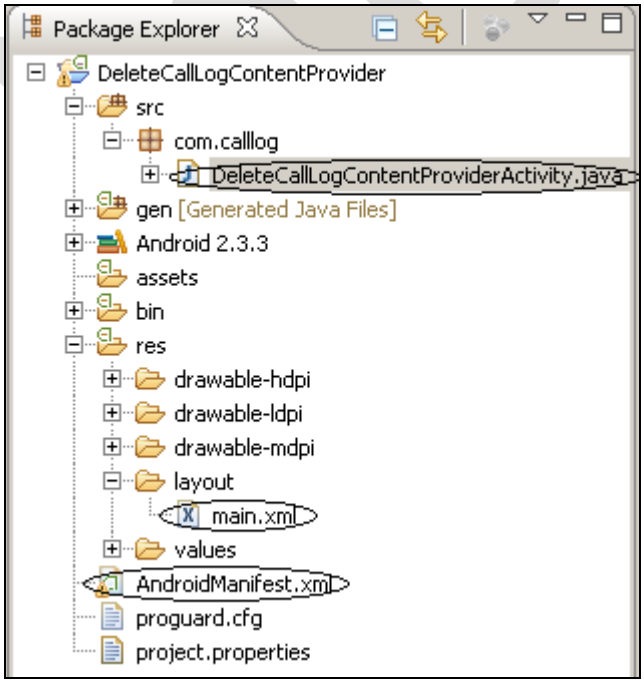
Call Log Content Providers :

- ✓ An Android Application can access the call log of the android phone. This is possible because the call log details are exposed by the Android platform as a content provider.
- ✓ The calllog content provider, stores the log information in its internal database as a content provider – “content://call_log/calls”.
- ✓ The call log data is used by multiple applications and hence android stores it as a content provider. The Call Log provider contains information about placed and received calls.
- ✓ The following Example demonstrates deleting a number from the Call log list using Call log content provider.

Steps :

1. Get uri for call log content provider.
   The call log content provider exposes a public URI “content://call_log/calls” for recent phone calls that uniquely identifies its data set.
   Uri allCalls = Uri.parse(“content://call_log/calls”);
2. Query the content provider.
   To access database we need to query the content provider by using the uri which the given content provider exposes.
   Cursor c = managedQuery(allCalls, null, null, null, null);
3. Search the row in result set(curser) & Delete the row from content provider.
   while(c.moveToNext()) {
   // get particular number for which log entry is to be deleted.
   String strNumber= etNumberForCall.getText().toString();
   // make a selection clause.
   String queryString= “NUMBER=’” + strNumber + “’”;
   Log.*v*(“Number”, queryString);
   CallLogActivity.this.getContentResolver().delete(UriCalls, queryString, null);
   }
4. Update Manifest .xml file
   <uses-permission android:name=”android.permission.GET_ACCOUNTS” />
   <uses-permission android:name=”android.permission.READ_CONTACTS” />
   <uses-permission android:name=”android.permission.WRITE_CONTACTS” />

   Example: Develop Android Application to delete a number from call log list ?



AndroidManifest.xml :

```
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
<uses-permission android:name="android.permission.WRITE_CONTACTS"></uses-permission>
```

main.xml

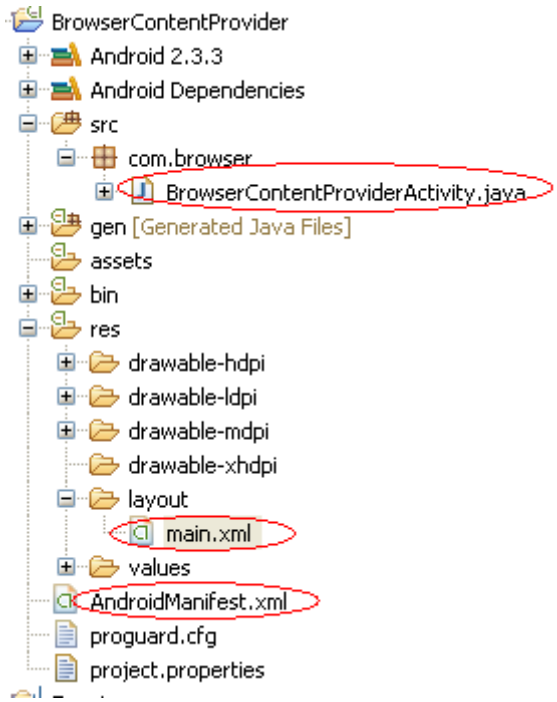
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <EditText
      android:id="@+id/EditText01"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" />
   <Button
      android:id="@+id/Button01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Button" />
</LinearLayout>
```

DeleteCallLogContentProviderActivity.java

```
public class DeleteCallLogContentProviderActivity extends Activity implements OnClickListener {
   /** Called when the activity is first created. */
   EditText etNumberForCall;
   Button btnDeleteNumberFromCallLog;
   @Override
   public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   etNumberForCall=(EditText)findViewById(R.id.EditText01);
```

```
    btnDeleteNumberFromCallLog=(Button)findViewById(R.id.Button01);
    btnDeleteNumberFromCallLog.setOnClickListener(this);
  }
    @Override
      public void onClick(View v) {
        // TODO Auto-generated method stub
        String strUriCalls="content://call_log/calls";
        Uri UriCalls = Uri.parse(strUriCalls);
        Cursor c = DeleteCallLogContentProviderActivity.this.getContentResolver().query(UriCalls, null, null,
null, null);
        if (c.getCount()<=0) {
            Toast.makeText(getApplicationContext(), "Call log empty",Toast.LENGTH_SHORT).show();
            etNumberForCall.setText("");
        }
        while (c.moveToNext()){
            String strNumber= etNumberForCall.getText().toString();
            String queryString= "NUMBER='" + strNumber + "'";
             Log.v("Number", queryString);
             int i=DeleteCallLogContentProviderActivity.this.getContentResolver().delete(UriCalls, queryString,
null);
             etNumberForCall.setText("");
             if(i>=1){
                 Toast.makeText(getApplicationContext(), "Number deleted", Toast.LENGTH_SHORT).show();
             }
             else{
                  Toast.makeText(getApplicationContext(), "No such number in call logs",
Toast.LENGTH_SHORT).show();
             }
       }
    }
}
```

Working with browser Content Provider :



AndroidManiest.xml :

```
<uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
```

main.xml:

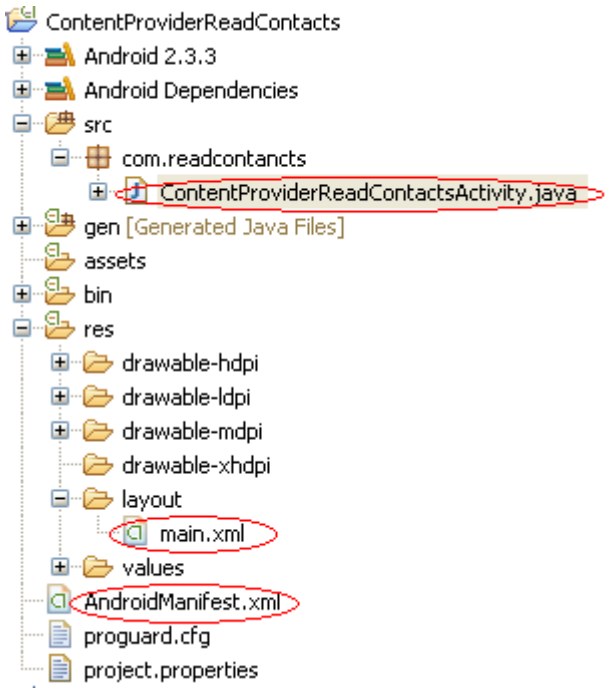
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
      </LinearLayout>
```

BrowserContentProviderActivity.java ?

```
public class BrowserContentProviderActivity
                        extends Activity {
        @Override
        public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
```

```
        // Requested columns to display
        String[] requestedColumns = {
                        Browser.BookmarkColumns.TITLE,
              Browser.BookmarkColumns.VISITS,
              Browser.BookmarkColumns.BOOKMARK
        };
        // make request to browser CP, with URI, with Requested column
        Cursor faves = super.managedQuery(
                        Browser.BOOKMARKS_URI, requestedColumns,
              null, null, null);
        Log.d("BookMark Data:", "Bookmarks count: " +
              faves.getCount());
        int titleIdx = faves.getColumnIndex(
                        Browser.BookmarkColumns.TITLE);
        int visitsIdx = faves.getColumnIndex(
                        Browser.BookmarkColumns.VISITS);
        int bmIdx = faves.getColumnIndex(
                        Browser.BookmarkColumns.BOOKMARK);
        faves.moveToFirst();
        while (!faves.isAfterLast()) {
          Log.d("SimpleBookmarks",
                faves.getString(titleIdx) + " visited "
                        + faves.getInt(visitsIdx) + " times : "
                        + (faves.getInt(bmIdx) != 0 ? "true" : "false"));
          faves.moveToNext();
        }
      }
    }
```

Working with Contacts Content Provider:



AndroidManifest.xml

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

main.xml

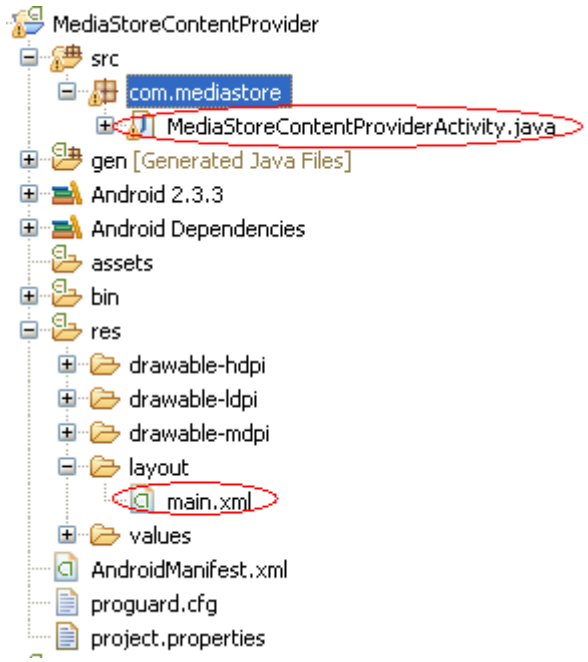
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
  <ListView
    android:id="@+id/android:list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"/>
      <TextView
        android:id="@+id/contactName"
        android:textStyle="bold"
        android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
      <TextView
```

```
            android:id="@+id/contactID"
            android:textStyle="bold"
            android:layout_width="fill_parent"
        android:layout_height="wrap_content"
            />
</LinearLayout>
```

ContentProviderReadContactsActivity.java

```
public class ContentProviderReadContactsActivity extends ListActivity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      //defining requested fields
      String columns[]=new String[]{
           ontactsContract.Contacts.DISPLAY_NAME,ContactsContract.Contacts._ID };
      //contact contentprovider URI
      Uri allContacts=Uri.parse("content://contacts/people");
      // making req to contact content providers with URI
      Cursor c=super.managedQuery(allContacts, null, null, null,null);
      //text view id's to display contact name, contact id
      int []views=new int[]{R.id.contactName,R.id.contactID};
      SimpleCursorAdapter adapter=new SimpleCursorAdapter(
                  this,R.layout.main,c,columns,views);
      this.setListAdapter(adapter);
   }
}
```

Working with MediaStore Content Providers:



main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
         <ListView
      android:id="@+id/PhoneMusicList"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:stackFromBottom="false"
      android:transcriptMode="normal"/>
   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello" />
</LinearLayout>
```

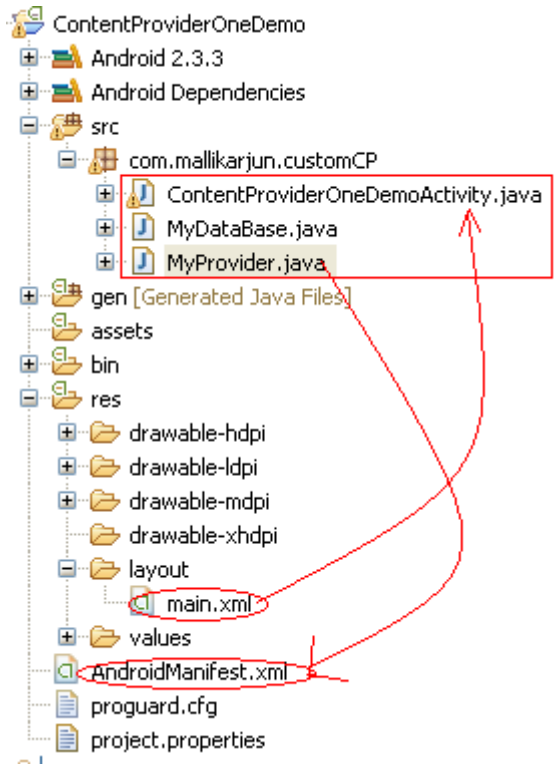
MediaStoreContentProviderActivity.java

```
         public class MediaStoreContentProviderActivity extends Activity {
private ArrayList<String> songs = new ArrayList<String>();
```

```
private ArrayList<String> albums = new ArrayList<String>();
private ArrayList<String> artists = new ArrayList<String>();
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
   Context context = getApplicationContext();
   CharSequence text = "Reading Music...";
   int duration = Toast.LENGTH_LONG;
   Toast toast = Toast.makeText(context, text, duration);
   toast.show();
   String[] projection = new String[] {
         MediaStore.EXTRA_MEDIA_TITLE,
         MediaStore.EXTRA_MEDIA_ARTIST,
         MediaStore.EXTRA_MEDIA_ALBUM };
   // Get the base URI for the People table in the Contacts content provider.
   Uri media =  MediaStore.getMediaScannerUri();
   // Make the query.
   Cursor mediaCursor = managedQuery(media,
         projection, // Which columns to return
         null,      // Which rows to return (all rows)
         null,      // Selection arguments (none)
         // Put the results in ascending order by name
         MediaStore.EXTRA_MEDIA_TITLE + " ASC");
   if (mediaCursor.moveToFirst()) {
      String title;
      String artist;
      String album;
      int titleColumn = mediaCursor.getColumnIndex(MediaStore.EXTRA_MEDIA_TITLE);
      int artistColumn = mediaCursor.getColumnIndex(MediaStore.EXTRA_MEDIA_ARTIST);
      int albumColumn = mediaCursor.getColumnIndex(MediaStore.EXTRA_MEDIA_ALBUM);
      do {
```

```
            // Get the field values
            title = mediaCursor.getString(titleColumn);
            artist = mediaCursor.getString(artistColumn);
            album = mediaCursor.getString(albumColumn);
            artists.add(artist);
            albums.add(album);
            songs.add(title);
        } while (mediaCursor.moveToNext());
    //new MusicGetter().execute();
    }
  }
}
```

Custom Content Providers :



MyDataBase.java

```
/*
 * Global database for all classes . it's used to create
 * custom content provider .
 */
public class MyDataBase extends SQLiteOpenHelper{
	public MyDataBase(Context context, String name,
			CursorFactory factory,int version) {
		super(context, name, factory, version);
	}
/*
 * create a table which contains 4 fields like _id, no, name,email
 */
	@Override
	public void onCreate(SQLiteDatabase db) {
	 db.execSQL("create table contenttable (_id int(3) primary key,"+
				"no int(3),name varchar(20),email varchar(20));");
	}
```

```
	@Override
	public void onUpgrade(SQLiteDatabase db, int oldVersion,
			int newVersion) {
	}
}
```

MyProvider.java

```
/*
 * Creating Custom Content provider :
 * ---------------------------------------------------
 * to create custom content provider  *
 * create a class that class should be extends ContentProvider
 * and override insert() , oncreate() , query() , update(),getType()
 *  methods ,
 */
public class MyProvider extends ContentProvider {
	// Step-1 Define User defined URI for custom CP.
	// CONTENT_URI : BASE URL FOR CONTENT PROVIDER
	public static final Uri CONTENT_URI = Uri.parse(
			"content:// com.Raj.customCP/contenttable");
	private MyDataBase mdb;
	private SQLiteDatabase db;
	/*
	 * allocating memory to database and giving permissions to
	 * database
	 */
	//Step-2 Override onCreate() method to get SQLiteDatabase.
	@Override
	public boolean onCreate() {
		mdb = new MyDataBase(getContext(), "ContentData", null, 2);
		db = mdb.getWritableDatabase();
```

```
            return true;
        }
        // Step-3 override CRUD operations
        /*
         * deleting values from DB using CP
         */
        @Override
        public int delete(Uri uri, String selection, String[] selectionArgs) {
            return 0;
        }
        /*
         * inserting values to database
         */
        @Override
        public Uri insert(Uri uri, ContentValues values) {
            db.insert("contenttable", null, values);
            return null;
        }
        /*
             * get URL type
         */
        @Override
        public String getType(Uri uri) {
            return null;
        }
        /*
          * fetching values from db
         */
        @Override
        public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder) {
```

```
                String col[] = { "_id", "no", "name", "email" };
                Cursor c = db.query("contenttable", col, null, null, null, null, null);
                return c;
        }
        /*
         * for updating values to database
        */
        @Override
        public int update(Uri uri, ContentValues values, String selection,
                        String[] selectionArgs) {
                return 0;
        }
}
```

main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <RelativeLayout
     android:layout_width="fill_parent"
     android:layout_height="wrap_content" >
     <TableLayout
       android:id="@+id/xTb"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:stretchColumns="*" >
       <TableRow
         android:layout_width="fill_parent"
         android:layout_height="fill_parent" >
         <Button
```

```
            android:id="@+id/xBtnSubmit"
            android:layout_span="4"
            android:text="Insert"
            android:onClick="insert"
            />
        <EditText
            android:id="@+id/xEtNum"
            android:layout_span="3"
             android:hint="no"
            android:singleLine="true"
            />
        <EditText
            android:id="@+id/xEtName"
            android:layout_span="4"
            android:hint="name"
            android:singleLine="true"
             />
        <EditText
            android:id="@+id/xEtMail"
            android:layout_span="4"
            android:hint="mail "
            android:singleLine="true"
             />
      </TableRow>
    </TableLayout>
  </RelativeLayout>
</RelativeLayout>
```

ContentProviderOnceDemoActivity.java

```
public class ContentProviderOneDemoActivity
        extends Activity{
        private ContentResolver cr;
```

```
        private Cursor c;
        private EditText mEtNum,mEtName,mEtMail;
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mEtNum=(EditText)findViewById(R.id.xEtNum);
      mEtName=(EditText)findViewById(R.id.xEtName);
      mEtMail=(EditText)findViewById(R.id.xEtMail);
      // getting content resolver object to refer CustomCP
      cr=super.getContentResolver();
    }
    // inserting values which is entered in edit text boxes
    public void insert(View v ){
        String no=mEtNum.getText().toString();
        String name=mEtName.getText().toString();
        String mail=mEtMail.getText().toString();
        ContentValues cv=new ContentValues();
        cv.put("no",no);
        cv.put("name",name);
        cv.put("email",mail);
        cr.insert(MyProvider.CONTENT_URI,cv);
        mEtNum.setText("");
        mEtName.setText("");
        mEtMail.setText("");
      Toast.makeText(this,"Record inserted",3000).show();
    }
}
```

AndroidManifest.xml

```
<provider android:exported="true"
```

```
        android:name="com.Raj.customCP.MyProvider"

        android:authorities=" com.Raj.customCP">

</provider>
```

65.2. Sharing data between Application

Web Services

Understanding of Web services

Web services are open standard (XML, SOAP, HTTP etc.) based Web applications that interact with other web applications for the purpose of exchanging data. Web Services can convert your existing applications into Web-applications. Different books and different organizations provide different definitions to Web Services. Some of them are listed here.

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications.

Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.

Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

To summarize, a complete web service is, therefore, any service that:

- ✓ Is available over the Internet or private (intranet) networks
- ✓ Uses a standardized XML messaging system
- ✓ Is not tied to any one operating system or programming language
- ✓ Is self-describing via a common XML grammar
- ✓ Is discoverable via a simple find mechanism

Components of Web Services

The basic web services platform is XML + HTTP. All the standard web services work using the following components

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

All these components have been discussed in the Web Services Architecture chapter.

How Does a Web Service Work?

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of:

- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.

You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.You can also use C# to build new web services on Windows that can be invoked from your web application that is based on JavaServer Pages (JSP) and runs on Linux.

Example

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.
The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.The steps to perform this operation are as follows:

- The client program bundles the account registration information into a SOAP message.
- This SOAP message is sent to the web service as the body of an HTTP POST request.
- The web service unpacks the SOAP request and converts it into a command that the application can understand.
- The application processes the information as required and responds with a new unique account number for that customer.
- Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
- The client program unpacks the SOAP message to obtain the results of the account registration process.

Architecture

There are two ways to view the web service architecture:
The first is to examine the individual roles of each web service actor.The second is to examine the emerging web service protocol stack.

Web Service Roles

There are three major roles within the web service architecture:

Service Provider

This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

Service Requestor

This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

Service Registry

This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.

Web Service Protocol Stack

A second option for viewing the web service architecture is to examine the emerging web service protocol stack. The stack is still evolving, but currently has four main layers.

Service Transport

This layer is responsible for transporting messages between applications. Currently, this layer includes Hyper Text Transport Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), and newer protocols such as Blocks Extensible Exchange Protocol (BEEP).

XML Messaging

This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP.

Service Description

This layer is responsible for describing the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (WSDL).

Service Discovery

This layer is responsible for centralizing services into a common registry and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI).
As web services evolve, additional layers may be added and additional technologies may be added to each layer.
The next chapter explains the components of web services.

Few Words about Service Transport

The bottom of the web service protocol stack is service transport. This layer is responsible for actually transporting XML messages between two computers.

Hyper Text Transfer Protocol (HTTP)

Currently, HTTP is the most popular option for service transport. HTTP is simple, stable, and widely deployed. Furthermore, most firewalls allow HTTP traffic. This allows XML-RPC or SOAP messages to masquerade as HTTP messages. This is good if you want to integrate remote applications, but it does raise a number of security concerns.

Blocks Extensible Exchange Protocol (BEEP)

This is a promising alternative to HTTP. BEEP is a new Internet Engineering Task Force (IETF) framework for building new protocols. BEEP is layered directly on TCP and includes a number of built-in features, including an initial handshake protocol, authentication, security, and error handling. Using BEEP, one can create new protocols for a variety of applications, including instant messaging, file transfer, content syndication, and network management. SOAP is not tied to any specific transport protocol. In fact, you can use SOAP via HTTP, SMTP, or FTP. One promising idea is therefore to use SOAP over BEEP.

Json

- JSON stands for JavaScript Object Notation
- JSON is a lightweight data-interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand
- Much Like XML Because
- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest
- Much Unlike XML Because
- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

The biggest difference is:

XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

Why JSON?

For AJAX applications, JSON is faster and easier than XML:

Using XML

- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

Using JSON

- Fetch a JSON string
- JSON.Parse the JSON string
- JSON Syntax Rules

JSON syntax is derived from JavaScript object notation syntax:

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays
- JSON Data - A Name and a Value
- JSON data is written as name/value pairs.

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

Example

"firstName":"John"

JSON names require double quotes. JavaScript names don't.

JSON Values

JSON values can be:

A number (integer or floating point)

A string (in double quotes)
A Boolean (true or false)
An array (in square brackets)
An object (in curly braces)
null

JSON Objects

JSON objects are written inside curly braces.
Just like JavaScript, JSON objects can contain multiple name/values pairs:
Example

```
{"firstName":"John", "lastName":"Doe"}
```

JSON Arrays
JSON arrays are written inside square brackets.
Just like JavaScript, a JSON array can contain multiple objects:
Example

```
"employees":[
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter","lastName":"Jones"}
]
```

In the example above, the object "employees" is an array containing three objects. Each object is a record of a person (with a first name and a last name).

Async Task

Purpose of the AsyncTask class

The AsyncTask class allows to run instructions in the background and to synchronize again with the main thread. It also reporting progress of the running tasks. AsyncTasks should be used for short background operations which need to update the user interface. .

Using the AsyncTask class

To use AsyncTask you must subclass it. AsyncTask uses generics and varargs. The parameters are the following AsyncTask <TypeOfVarArgParams , ProgressValue , ResultValue>.
An AsyncTask is started via the execute() method. This execute() method calls the doInBackground() and the onPostExecute() method.
TypeOfVarArgParams is passed into the doInBackground() method as input. ProgressValue is used for progress information and ResultValue must be returned from doInBackground() method. This parameter is passed to onPostExecute() as a parameter.
The doInBackground() method contains the coding instruction which should be performed in a background thread. This method runs automatically in a separate Thread.
The onPostExecute() method synchronizes itself again with the user interface thread and allows it to be updated. This method is called by the framework once the doInBackground() method finishes.

Parallel execution of several AsyncTasks

Android executes AsyncTask tasks before Android 1.6 and again as of Android 3.0 in sequence by default. You can tell Android to run it in parallel with the usage of the executeOnExecutor() method specifying AsyncTask.THREAD_POOL_EXECUTOR as first parameter.
The following code snippet demonstrates that.

```
// ImageLoader extends AsyncTask
ImageLoader imageLoader = new ImageLoader( imageView );
// Execute in parallel
imageLoader.executeOnExecutor( AsyncTask.THREAD_POOL_EXECUTOR, "http://url.com/image.png" );
```

[NOTE} The AsyncTask does not handle configuration changes automatically, i.e. if the activity is recreated. The programmer has to handle that in his coding. A common solution to this is to declare the AsyncTask in a retained headless fragment.

Example: AsyncTask

The following code demonstrates how to use the AsyncTask class to download the content of a webpage.

Create a new Android project called de.vogella.android.asynctask with an activity called ReadWebpageAsyncTask.
Add the android.permission.INTERNET permission to your AndroidManifest.xml file.

Sending Data to server

Sending data to server using Async task.

Retrieving Data from Server

Retryving data to server using Async task.

Registration Example

```
<?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: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="androindian.sampleregistration.Registrion"
   android:orientation="vertical">

   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Welcome"
      android:gravity="center"/>
   <android.support.design.widget.TextInputLayout
      android:id="@+id/input_layout_passwordaa"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      >

   <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Mobile Number"
      android:inputType="number"
      android:id="@+id/mobile"
      android:textColorHint="@color/colorPrimary"/>
   </android.support.design.widget.TextInputLayout>
   <android.support.design.widget.TextInputLayout
      android:id="@+id/input_layout_passwordaaa"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">
   <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Mobile Number"
      android:inputType="textPassword"
      android:id="@+id/pass"/>
   </android.support.design.widget.TextInputLayout>

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

```
            android:text="Login"
            android:id="@+id/login"/>

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="SignUp"
            android:gravity="center"
            android:textColor="@color/colorAccent"
            android:id="@+id/reggg"/>
        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#000"></View>

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

</LinearLayout>
```

Login.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: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="androindian.sampleregistration.Registrion"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Welcome"
        android:gravity="center"/>
    <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="Name"
        android:id="@+id/name"/>
    </android.support.design.widget.TextInputLayout>

    <android.support.design.widget.TextInputLayout
        android:id="@+id/input_layout_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

    <EditText
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Email"
             android:id="@+id/email"/>
        </android.support.design.widget.TextInputLayout>

        <android.support.design.widget.TextInputLayout
            android:id="@+id/input_layout_passwordaa"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Mobile Number"
            android:inputType="number"
            android:id="@+id/mobile"/>
        </android.support.design.widget.TextInputLayout>

        <android.support.design.widget.TextInputLayout
            android:id="@+id/input_layout_passworda"
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Password"
            android:inputType="textPassword"
            android:id="@+id/pass"/>
        </android.support.design.widget.TextInputLayout>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Register"
            android:id="@+id/re"/>


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


</LinearLayout>
```

Registration.Java

```
package androindian.sampleregistration;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class Registrion extends AppCompatActivity {
```

```
    TextView reg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registrion);
        reg=(TextView)findViewById(R.id.reggg);
        reg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent reg=new Intent(Registrion.this,Second.class);
                startActivity(reg);
            }
        });
    }
}
```

Second.Java

```
package androindian.sampleregistration;

import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.ads.AdListener;
import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdView;
import com.google.android.gms.ads.InterstitialAd;

import org.json.JSONException;
import org.json.JSONObject;

/**
 * Created by Raj on 7/6/2016.
 */
public class Second extends AppCompatActivity{

    EditText name,mobile,email,password;
    Button bt;
    String uname, umobile, upass, uemail;
    String respone="";
    JSONObject jsonoBject=null;
    AdView adView;


    InterstitialAd interstitial;
```

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

   name= (EditText) findViewById(R.id.name);
   mobile= (EditText) findViewById(R.id.mobile);
   email= (EditText) findViewById(R.id.email);
   password= (EditText) findViewById(R.id.pass);

   bt=(Button)findViewById(R.id.re);

   try{
      adView = (AdView) this.findViewById(R.id.adView);

      AdRequest adRequest = new AdRequest.Builder().build();
      adView.loadAd(adRequest);
   } catch (Exception e) {
      e.printStackTrace();
   }

   try{
      interstitial = new InterstitialAd(this);
      interstitial.setAdUnitId("ca-app-pub-9827189980172148/3612898112");

      AdRequest adRequest1 = new AdRequest.Builder().build();
      interstitial.loadAd(adRequest1);
   }catch(Exception e){

   }
   interstitial.setAdListener(new AdListener() {
      public void onAdLoaded() {
         displayInterstitial();
      }
   });

   bt.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         uname = name.getText().toString().trim();
         umobile = mobile.getText().toString().trim();
         upass = password.getText().toString().trim();
         uemail = email.getText().toString().trim();

         JSONObject samreg=new JSONObject();
         try {
            samreg.put("name", uname);

         samreg.put("mobile", umobile);
         samreg.put("email", uemail);
         samreg.put("pswrd", upass);
```

```
                samreg.put("baction", "register_user");
                } catch (JSONException e) {
                    e.printStackTrace();
                }

                Reg re = new Reg();
                re.execute(samreg.toString());
                Log.e("Json Values", samreg.toString());
            }
        });
    }

    public class Reg extends AsyncTask<String,String,String> {
        String url = "http://androindian.com/apps/example_app/api.php";

        @Override
        protected String doInBackground(String... params) {

            jsonoBject = JsonFunction.getJsonFromUrlparam(url, params[0]);
            Log.i("json", "" + jsonoBject);
            return String.valueOf(jsonoBject);
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            try {
                JSONObject jsonObject = new JSONObject(jsonoBject.toString());
                respone = jsonObject.getString("response");

                if (respone.trim().equals("success")) {

                    String status = jsonObject.getString("user");
                    Toast.makeText(getApplicationContext(), "" + status, Toast.LENGTH_LONG).show();

                    Intent regintent = new Intent(Second.this, Registrion.class);
                    startActivity(regintent);

                } else if (respone.trim().equals("failed")) {
                    String status1 = jsonObject.getString("user");
                    Toast.makeText(getApplicationContext(), "" + status1, Toast.LENGTH_LONG).show();


                } else if (respone.trim().equals("error")) {
                    String status2 = jsonObject.getString("user");
                    Toast.makeText(getApplicationContext(), "" + status2, Toast.LENGTH_LONG).show();

                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

```
    public void displayInterstitial() {
      if (interstitial.isLoaded()) {
        interstitial.show();
      }
    }
}
```

JsonFunctions.Java

```
package androindian.sampleregistration;

import android.util.Log;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

/**
 * Created by Raj on 17-05-2016.
 */
public class JsonFunction {

    public JSONObject getJsonFromUrl(String url)
    {
        JSONObject jsonObject = null;

        try {

            URL jsonUrl=new URL(url);
            HttpURLConnection connection= (HttpURLConnection) jsonUrl.openConnection();
            InputStream is=new BufferedInputStream(connection.getInputStream());
            BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(is));
            StringBuffer sb=new StringBuffer();
            String line;
            while ((line=bufferedReader.readLine())!=null)
            {
                sb.append(line+"\n");
            }
            is.close();
            Log.i("sb",sb.toString());
            jsonObject=new JSONObject(sb.toString());

        } catch (MalformedURLException e) {
```

```
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return jsonObject;
    }
    public static JSONObject getJsonFromUrlparam(String url, String para)
    {
        JSONObject jsonObject = null;

        try {

            URL jsonUrl=new URL(url);
            HttpURLConnection connection= (HttpURLConnection) jsonUrl.openConnection();

            connection.setDoOutput(true);
            connection.setDoInput(true);
            connection.setConnectTimeout(60000);
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
            connection.setRequestProperty("Accept-Charset", "UTF-8");
            connection.setRequestProperty("Accept", "application/json");
            connection.setUseCaches(false);
            connection.connect();


            OutputStream mOutPut = new BufferedOutputStream(connection.getOutputStream());
            mOutPut.write(para.getBytes());
            mOutPut.flush();

            InputStream is=new BufferedInputStream(connection.getInputStream());
            BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(is));
            StringBuffer sb=new StringBuffer();
            String line;
            while ((line=bufferedReader.readLine())!=null)
            {
                sb.append(line+"\n");
            }
            is.close();

            Log.i("json",sb.toString());
            jsonObject=new JSONObject(sb.toString());
            Log.i("json",""+jsonObject);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return jsonObject;
```

```
    }
    public static String getJsonUrlparamString(String url, String para)
    {
       StringBuffer sb = null;

       try {

          URL jsonUrl=new URL(url);
          HttpURLConnection connection= (HttpURLConnection) jsonUrl.openConnection();

          connection.setDoOutput(true);
          connection.setDoInput(true);
          connection.setConnectTimeout(60000);
          connection.setRequestMethod("POST");
          connection.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
          connection.setRequestProperty("Accept-Charset", "UTF-8");
          connection.setRequestProperty("Accept", "application/json");
          connection.setUseCaches(false);
          connection.connect();


          OutputStream mOutPut = new BufferedOutputStream(connection.getOutputStream());
          mOutPut.write(para.getBytes());
          mOutPut.flush();

          InputStream is=new BufferedInputStream(connection.getInputStream());
          BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(is));
          sb=new StringBuffer();
          String line;
          while ((line=bufferedReader.readLine())!=null)
          {
             sb.append(line+"\n");
          }
          is.close();

          Log.i("json",sb.toString());
       } catch (MalformedURLException e) {
          e.printStackTrace();
       } catch (IOException e) {
          e.printStackTrace();
       }
       return sb.toString();
    }
}
```

DOM Parser

The Document Object Model is an official recommendation of the World Wide Web Consortium (W3C). It defines an interface that enables programs to access and update the style, structure,and contents of XML documents. XML parsers that support the DOM implement that interface.

When to use?

You should use a DOM parser when:

You need to know a lot about the structure of a document

You need to move parts of the document around (you might want to sort certain elements, for example)

You need to use the information in the document more than once

What you get?

When you parse an XML document with a DOM parser, you get back a tree structure that contains all of the elements of your document. The DOM provides a variety of functions you can use to examine the contents and structure of the document.

Advantages

The DOM is a common interface for manipulating document structures. One of its design goals is that Java code written for one DOM-compliant parser should run on any other DOM-compliant parser without changes.

DOM interfaces

The DOM defines several Java interfaces. Here are the most common interfaces:

Node - The base datatype of the DOM.

Element - The vast majority of the objects you'll deal with are Elements.

Attr Represents an attribute of an element.

Text The actual content of an Element or Attr.

Document Represents the entire XML document. A Document object is often referred to as a DOM tree.

Common DOM methods

When you are working with the DOM, there are several methods you'll use often:

Document.getDocumentElement() - Returns the root element of the document.

Node.getFirstChild() - Returns the first child of a given Node.

Node.getLastChild() - Returns the last child of a given Node.

Node.getNextSibling() - These methods return the next sibling of a given Node.

Node.getPreviousSibling() - These methods return the previous sibling of a given Node.

Node.getAttribute(attrName) - For a given Node, returns the attribute with the requested name.

Steps to Using DOM

Following are the steps used while parsing a document using DOM Parser.

Import XML-related packages.

Create a DocumentBuilder

Create a Document from a file or stream

Extract the root element

Examine attributes

Examine sub-elements

SAX Parser

SAX (the Simple API for XML) is an event-based parser for xml documents.Unlike a DOM parser, a SAX parser creates no parse tree. SAX is a streaming interface for XML, which means that applications using SAX receive event notifications about the XML document being processed an element, and attribute, at a time in sequential order starting at the top of the document, and ending with the closing of the ROOT element.

Reads an XML document from top to bottom, recognizing the tokens that make up a well-formed XML document

Tokens are processed in the same order that they appear in the document

Reports the application program the nature of tokens that the parser has encountered as they occur

The application program provides an "event" handler that must be registered with the parser

As the tokens are identified, callback methods in the handler are invoked with the relevant information

When to use?

You should use a SAX parser when:

You can process the XML document in a linear fashion from the top down.The document is not deeply nested. You are processing a very large XML document whose DOM tree would consume too much memory.Typical DOM implementations use ten bytes of memory to represent one byte of XML. The problem to be solved involves only part of the XML document.Data is available as soon as it is seen by the parser, so SAX works well for an XML document that arrives over a stream

Disadvantages of SAX

We have no random access to an XML document since it is processed in a forward-only manner

If you need to keep track of data the parser has seen or change the order of items, you must write the code and store the data on your ownContentHandler Interface

This interface specifies the callback methods that the SAX parser uses to notify an application program of the components of the XML document that it has seen.
void startDocument() - Called at the beginning of a document.
void endDocument() - Called at the end of a document.
void startElement(String uri, String localName, String qName, Attributes atts) - Called at the beginning of an element.
void endElement(String uri, String localName,String qName) - Called at the end of an element.
void characters(char[] ch, int start, int length) - Called when character data is encountered.
void ignorableWhitespace(char[] ch, int start, int length) - Called when a DTD is present and ignorable whitespace is encountered.
void processingInstruction(String target, String data) - Called when a processing instruction is recognized.
void setDocumentLocator(Locator locator)) - Provides a Locator that can be used to identify positions in the document.
void skippedEntity(String name) - Called when an unresolved entity is encountered.
void startPrefixMapping(String prefix, String uri) - Called when a new namespace mapping is defined.
void endPrefixMapping(String prefix) - Called when a namespace definition ends its scope.

Attributes Interface

This interface specifies methods for processing the attributes connected to an element.
int getLength() - Returns number of attributes.
String getQName(int index)
String getValue(int index)
String getValue(String qname)

Restful web services

RESTful Web Services are REST architecture based web services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web based applications.

What is REST ?

REST stands for REpresentational State Transfer. REST is web standards based architecture and uses HTTP Protocol for data communication. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000. In REST architecture, a REST Server simply provides access to resources and REST client accesses and presents the resources. Here each resource is identified by URIs/ global IDs. REST uses various representations to represent a resource like text, JSON and XML. Now a days JSON is the most popular format being used in web services.
HTTP Methods
Following well known HTTP methods are commonly used in REST based architecture.
GET - Provides a read only access to a resource.
PUT - Used to create a new resource.
DELETE - Used to remove a resource.
POST - Used to update a existing resource or create a new resource.
OPTIONS - Used to get the supported operations on a resource.

RESTFul Web Services

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.
Web services based on REST Architecture are known as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

activity_main.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"
   tools:context=".MainActivity" >

   <TextView
      android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:text="@string/hello_world" />

</RelativeLayout>
```

MainActivity.java:

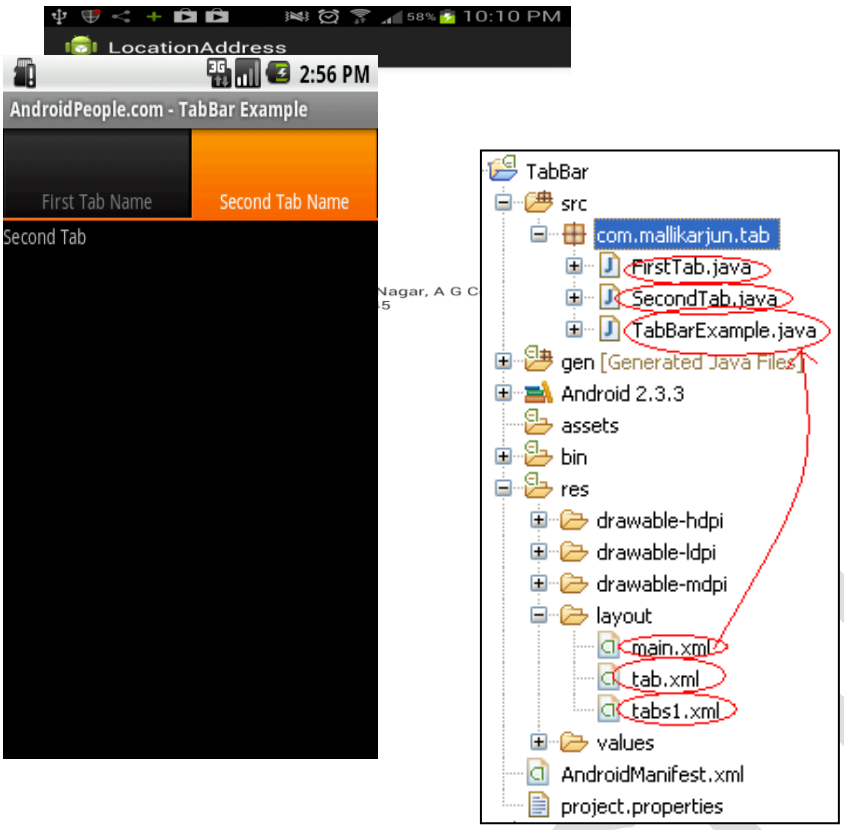
```
package com.ram.locationaddress;

import java.io.IOException;
import java.util.List;
import java.util.Locale;

import android.app.Activity;
import android.content.Context;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends Activity {
 double latti, longi;
 TextView display;

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

  display = (TextView) findViewById(R.id.text);
```

```
 LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

 LocationListener locationListener = new LocationListener() {

  @Override
  public void onStatusChanged(String provider, int status,
    Bundle extras) {
   // TODO Auto-generated method stub

  }

  @Override
  public void onProviderEnabled(String provider) {
   // TODO Auto-generated method stub

  }

  @Override
  public void onProviderDisabled(String provider) {
   // TODO Auto-generated method stub

  }

  @Override
  public void onLocationChanged(Location location) {
   // TODO Auto-generated method stub
   latti = location.getLatitude();
   longi = location.getLongitude();

   getAddress();


  }
 };

 locationManager.requestLocationUpdates(
   LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);

}

void getAddress() {
 Geocoder gc = new Geocoder(getApplicationContext(), Locale.getDefault());

 try {
  // Addresses:A class representing an Address, i.e, a set of Strings
  // describing a location
  List<Address> addresses = gc.getFromLocation(latti, longi, 1);

  String addr = "";

  if (addresses.size() > 0) {
   for (int i = 0; i <       addresses.get(0).getMaxAddressLineIndex(); i++)
```

```
            addr += addresses.get(0).getAddressLine(i) + "\n";
           }

           display.setText(addr);
          } catch (IOException e) {
           // TODO Auto-generated catch block
           e.printStackTrace();
          }
         }

        }
```

1.1.1. AndroidManifest.java:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ram.locationaddress"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
       <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.ram.locationaddress.MainActivity"
            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>
```



Advanced UI Components

Tab Host :

main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
<TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/hello"/>
</LinearLayout>
```

tab.xml

```
<TabHost android:layout_width="fill_parent"
        android:layout_height="fill_parent" xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@android:id/tabhost">
```

```
        <LinearLayout android:id="@+id/LinearLayout01"
                android:orientation="vertical" android:layout_height="fill_parent"
                android:layout_width="fill_parent">
                <TabWidget android:id="@android:id/tabs"
                        android:layout_height="wrap_content"
android:layout_width="fill_parent"></TabWidget>
                <FrameLayout android:id="@android:id/tabcontent"
                        android:layout_height="fill_parent"
android:layout_width="fill_parent"></FrameLayout>
        </LinearLayout>
</TabHost>
```

tabs1.xml

```
<LinearLayout
   android:id="@+id/LinearLayout01"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android">
</LinearLayout>
```

FirstTab.java

```
public class FirstTab extends Activity {
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                /* First Tab Content */
                TextView textView = new TextView(this);
                textView.setText("First Tab");
                setContentView(textView);
        }
}
```

SecondTab.java

```
public class SecondTab extends Activity {
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                /* Second Tab Content */
                TextView textView = new TextView(this);
                textView.setText("Second Tab");
                setContentView(textView);

        }
}
```

TabBarExample.java

```
public class TabBarExample extends TabActivity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.tab);
      /* TabHost will have Tabs */
      TabHost tabHost = (TabHost)findViewById(android.R.id.tabhost);
      /* TabSpec used to create a new tab.
       * By using TabSpec only we can able to setContent to the tab.
       * By using TabSpec setIndicator() we can set name to tab. */
      /* tid1 is firstTabSpec Id. Its used to access outside. */
      TabSpec firstTabSpec = tabHost.newTabSpec("tid1");
      TabSpec secondTabSpec = tabHost.newTabSpec("tid1");
      /* TabSpec setIndicator() is used to set name for the tab. */
      /* TabSpec setContent() is used to set content for a particular tab. */
```

```
        firstTabSpec.setIndicator("First Tab Name").setContent(new Intent(this,FirstTab.class));
        secondTabSpec.setIndicator("Second Tab Name").setContent(new Intent(this,SecondTab.class));
        /* Add tabSpec to the TabHost to display. */
        tabHost.addTab(firstTabSpec);
        tabHost.addTab(secondTabSpec);

    }
}
```

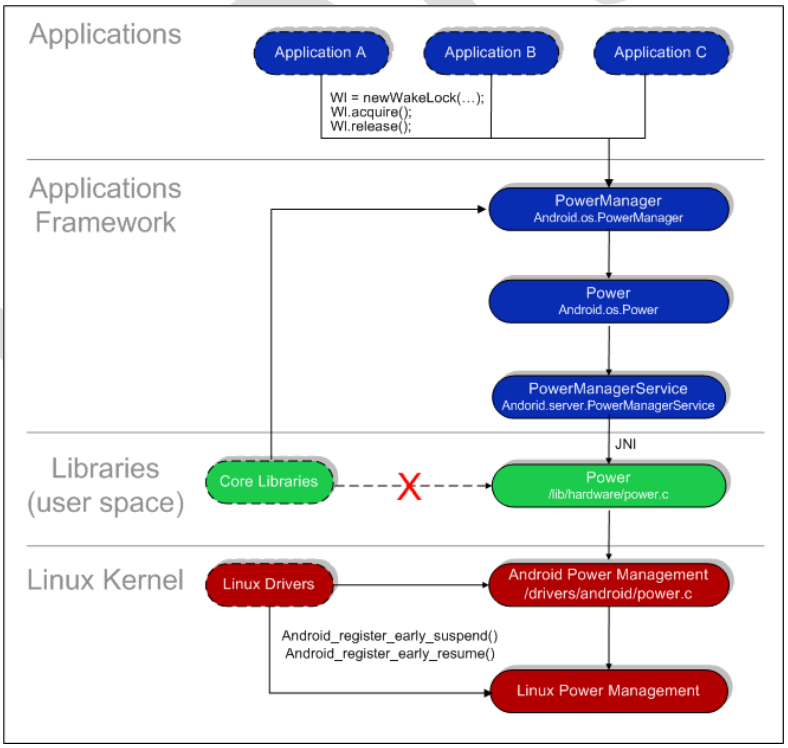
Wake Locks

Android supports its own Power Management (on top of the standard Linux Power Management) designed with the premise that the CPU shouldn't consume power if no applications or services require power. For more information regarding standard Linux power management, please see Linux Power Management Support at http://kernel.org.

(Imp) Reference : Refer PowerManager class in Android API

Android requires that applications and services request CPU resources with "wake locks" through the Android application framework and native Linux libraries. If there are no active wake locks, Android will shut down the CPU.

The image below illustrates the Android power management architecture.



- **Wake Locks**

Wake locks are used by applications and services to request CPU resources.

- *Types of Wake Locks*

Wake Lock	Description
ACQUIRE_CAUSES_WAKEUP	Normally wake locks don't actually wake the device, they just cause it to remain on once it's already on. Think of the video player app as the normal behavior. Notifications that pop up and want the device to be on are the exception; use this flag to be like them.
FULL_WAKE_LOCK	Wake lock that ensures that the screen and keyboard are on at full brightness.
ON_AFTER_RELEASE	When this wake lock is released, poke the user activity timer so the screen stays on for a little longer.
PARTIAL_WAKE_LOCK	Wake lock that ensures that the CPU is running. The screen might not be on.
SCREEN_BRIGHT_WAKE_LOCK	Wake lock that ensures that the screen is on at full brightness; the keyboard backlight will be allowed to go off.
SCREEN_DIM_WAKE_LOCK	Wake lock that ensures that the screen is on, but the keyboard backlight will be allowed to go off, and the screen backlight will be allowed to go dim.

- **Wake Lock Example**

All power management calls follow the same basic format:

3. Acquire handle to the PowerManager service.
4. Create a wake lock and specify the power management flags for screen, timeout, etc.
5. Acquire wake lock.
6. Perform operation (play MP3, open HTML page, etc.).
7. Release wake lock.

I given sample code to work with Wakelocks :

```
PowerManager pm = (PowerManager)mContext.getSystemService(
                                Context.POWER_SERVICE);
PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.FULL_WAKELOCK,
                            TAG);
wl.acquire();
 // in pause() method define the following method.
wl.release();
```

WakeLock Permission :

```
        <uses-permission android:name="android.permission.WAKE_LOCK" />
```

65.3. Adapters
65.4. Navigation Drawers

Fragments

A Fragment is a piece of an activity which enable more modular activity design. It will not be wrong if we say, a fragment is a kind ofsub-activity.

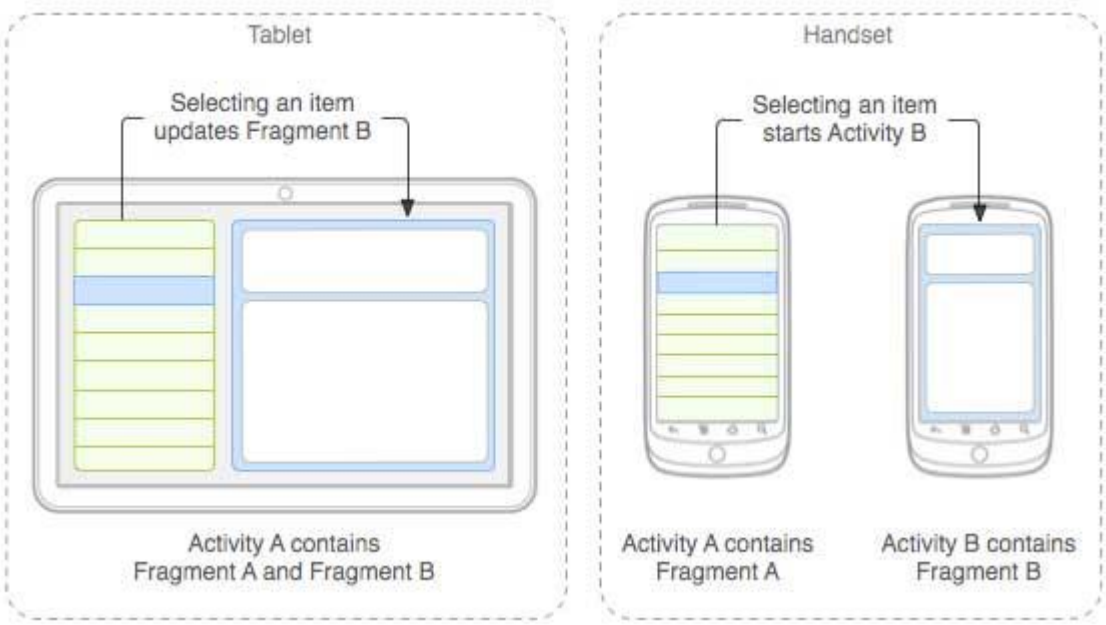
Following are important points about fragment −

- A fragment has its own layout and its own behaviour with its own life cycle callbacks.
- You can add or remove fragments in an activity while the activity is running.
- You can combine multiple fragments in a single activity to build a multi-plane UI.
- A fragment can be used in multiple activities.
- Fragment life cycle is closely related to the life cycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.
- A fragment can implement a behaviour that has no user interface component.
- Fragments were added to the Android API in Honeycomb version of Android which API version 11.

You create fragments by extending Fragment class and You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element.

Prior to fragment introduction, we had a limitation because we can show only a single activity on the screen at one given point in time. So we were not able to divide device screen and control different parts separately. But with the introduction of fragment we got more flexibility and removed the limitation of having a single activity on the screen at a time. Now we can have a single activity but each activity can comprise of multiple fragments which will have their own layout, events and complete life cycle.

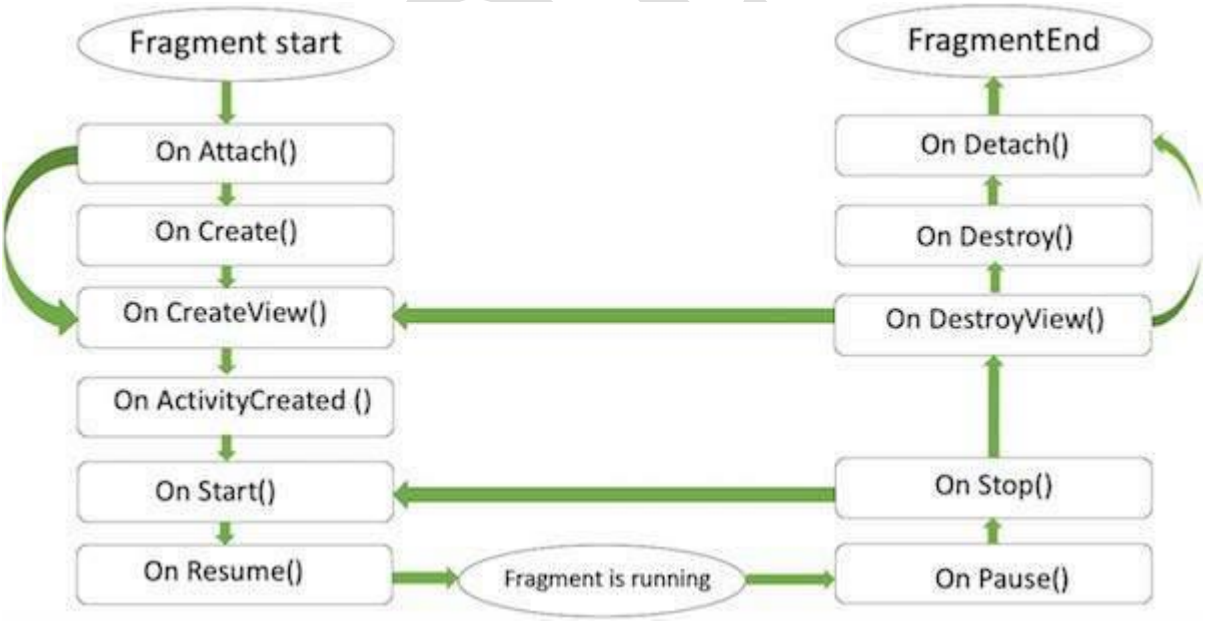
Following is a typical example of how two UI modules defined by fragments can be combined into one activity for a tablet design, but separated for a handset design.



The application can embed two fragments in Activity A, when running on a tablet-sized device. However, on a handset-sized screen, there's not enough room for both fragments, so Activity A includes only the fragment for the list of articles, and when the user selects an article, it starts Activity B, which includes the second fragment to read the article.

Fragment Life Cycle

Android fragments have their own life cycle very similar to an android activity. This section briefs different stages of its life cycle.



1.1.1.1. FRAGMENT LIFECYCLE

Here is the list of methods which you can to override in your fragment class −

- onAttach()The fragment instance is associated with an activity instance.The fragment and the activity is not fully initialized. Typically you get in this method a reference to the activity which uses the fragment for further initialization work.

- onCreate() The system calls this method when creating the fragment. You should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.

- onCreateView() The system calls this callback when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a View component from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.

- onActivityCreated()The onActivityCreated() is called after the onCreateView() method when the host activity is created. Activity and fragment instance have been created as well as the view hierarchy of the activity. At this point, view can be accessed with the findViewById() method. example. In this method you can instantiate objects which require a Context object

- onStart()The onStart() method is called once the fragment gets visible.

- onResume()Fragment becomes active.

- onPause() The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session.

- onStop()Fragment going to be stopped by calling onStop()

- onDestroyView()Fragment view will destroy after call this method

- onDestroy()onDestroy() called to do final clean up of the fragment's state but Not guaranteed to be called by the Android platform.

1.2. How to use Fragments?

This involves number of simple steps to create Fragments.

- First of all decide how many fragments you want to use in an activity. For example let's we want to use two fragments to handle landscape and portrait modes of the device.

- Next based on number of fragments, create classes which will extend the *Fragment* class. The Fragment class has above mentioned callback functions. You can override any of the functions based on your requirements.

- Corresponding to each fragment, you will need to create layout files in XML file. These files will have layout for the defined fragments.

- Finally modify activity file to define the actual logic of replacing fragments based on your requirement.

1.3. Types of Fragments

Basically fragments are divided as three stages as shown below.

- Single frame fragments − Single frame fragments are using for hand hold devices like mobiles, here we can show only one fragment as a view.

- List fragments − fragments having special list view is called as list fragment

- Fragments transaction − Using with fragment transaction. we can move one fragment to another fragment.

65.5. Telephony Manager

Android TelephonyManager provides information about the android telephony system. To use the TelephonyManager first get the instance of the Telephony Service by calling Context.getSystemService(TELEPHONY_SERVICE). This telephony service can be used to retrieve Call State, Cell Location, Operator Name, etc… as well as to listen to the various telephony events. Following are some of the important information we can get from TelephonyManager:

Cell Location :

The getCellLocation method is used to get the Cell Location of the device. This method returns an instance of GsmCellLocation class. The getCid() and getLac() methods of this class can used to retrieve the Cell ID and LAC of the device. This method requires the permission ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION. Following code shows how to retrieve the Cell ID and LAC.

```
TelephonyManager tm = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
GsmCellLocation loc = (GsmCellLocation) tm.getCellLocation();
int cellid = loc.getCid();
int lac = loc.getLac();
```

IMEI/MEID :

The getDeviceId() method is used to get the IMEI/MEID of the device. If the device is a GSM device then IMEI will be returned and if the device is a CDMA device then MEID will be returned. This device id can be used to uniquely identify the device. This method requires the permission READ_PHONE_STATE. Following code retrieve the device id.

```
String deviceid = tm.getDeviceId();
```

Device Phone Number:

The getLine1Number() method returns the device phone number (MSISDN). This method requires the permission READ_PHONE_STATE. Following code retrieve the device phone number:

```
String phonenumber = tm.getLine1Number();
```

Note: This function name not work prior to Android Version 2.0.

Network Name:

The getNetworkOperatorName() method returns the registered network operator's name, getNetworkOperator() method returns the MCC + MNC of the registered network operator and getNetworkCountryIso() returns the registered network operator's country code. This information may not be available on CDMA devices. Following code retrieves these details:

```
String operatorname = tm.getNetworkOperatorName();
String operatorcode = tm.getNetworkOperator();
String operatoriso = tm.getNetworkCountryIso();
```

SIM Card Infomarion:

The getSimCountryIso() returns the SIM operator's country code, getSimOperator() returns the SIM operator's MNC + MCC number, getSimOperatorName() returns the SIM operator's name and getSimSerialNumber() returns the SIM Serial Number. This method requires the permission READ_PHONE_STATE. Following code snippets reads the SIM Card information:

```
String simcountrycode = tm.getSimCountryIso();
String simoperator = tm.getSimOperatorName();
String simserialno = tm.getSimSerialNumber();
```

Network Type :

The getNetworkType() returns the type of the network available in the device. This method returns one of the following values:

```
TelephonyManager.NETWORK_TYPE_UNKNOWN
TelephonyManager.NETWORK_TYPE_GPRS
TelephonyManager.NETWORK_TYPE_EDGE
TelephonyManager.NETWORK_TYPE_UMTS
```

Phone Type :

The getPhoneType() returns the device type. This method returns one of the following values:

```
TelephonyManager.PHONE_TYPE_NONE
TelephonyManager.PHONE_TYPE_GSM
TelephonyManager.PHONE_TYPE_CDMA
```

Subscriber ID :

getSubscriberId() return the subscriber id of the device. This is IMSI if the device is a GSM device. If unavailable the function returns null. This function requires the permission READ_PHONE_STATE.

Neighboring Cell Information :

The getNeighboringCellInfo() function returns a list of NeighboringCellInfo class which represents the neighboring cell information if available otherwise the function returns null. This function returns the permission ACCESS_COARSE_UPDATES. Following code returns the this information:

```
List cellinfo = tm.getNeighboringCellInfo();
for(NeighboringCellInfo info: cellinfo){
   cellid = info.getCid();
   rssi = info.getRssi();
}
```

The listen() method is used to register a phone state listener. It accepts a PhoneStateListener instance and an int value specifying what are the events to listen. Following are the events we can listen:

```
PhoneStateListener.LISTEN_SIGNAL_STRENGTH
PhoneStateListener.LISTEN_DATA_ACTIVITY
PhoneStateListener.LISTEN_CELL_LOCATION
PhoneStateListener.LISTEN_CALL_STATE
PhoneStateListener.LISTEN_CALL_FORWARDING_INDICATOR
PhoneStateListener.LISTEN_DATA_CONNECTION_STATE
PhoneStateListener.LISTEN_MESSAGE_WAITING_INDICATOR
PhoneStateListener.LISTEN_SERVICE_STATE
```

PhoneStateListener class has following method:

```
void onCallForwardingIndicatorChanged(boolean cfi)
void onCallStateChanged(int state, String incomingNumber)
void onCellLocationChanged(CellLocation location)
void onDataActivity(int direction)
void onDataConnectionStateChanged(int state)
void onDataConnectionStateChanged(int state, int networkType)
void onMessageWaitingIndicatorChanged(boolean mwi)
void onServiceStateChanged(ServiceState serviceState)
void onSignalStrengthChanged(int asu)
void onSignalStrengthsChanged(SignalStrength signalStrength)
```

Following table describes important functions and its permission:

Function	Description
getCellLocation()	Returns the the Cell Location of the device ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION
getDeviceId()	Returns the IMEI/MEID of the device. If the device is a GSM device then IMEI will be returned and if the device is a CDMA device then MEID will be returned READ_PHONE_STATE
getLine1Number()	Returns the device phone number (MSISDN) READ_PHONE_STATE
getNetworkOperatorName()	Returns the registered network operator's name
getNetworkOperator()	Returns the MCC + MNC of the registered network operator
getNetworkCountryIso()	Returns the registered network operator's country code
getSimCountryIso()	Returns the SIM operator's country code

	READ_PHONE_STATE
getSimOperator()	Returns the SIM operator's MNC + MCC number READ_PHONE_STATE
getSimOperatorName()	Returns the SIM operator's name READ_PHONE_STATE
getSimSerialNumber()	Returns the SIM Serial Number READ_PHONE_STATE
getNetworkType()	Returns the type of the network available in the device. This will be one of the following values: TelephonyManager.NETWORK_TYPE_UNKNOWN TelephonyManager.NETWORK_TYPE_GPRS TelephonyManager.NETWORK_TYPE_EDGE TelephonyManager.NETWORK_TYPE_UMTS READ_PHONE_STATE
getPhoneType()	Returns the device type. This will be one of the following values: TelephonyManager.PHONE_TYPE_NONE TelephonyManager.PHONE_TYPE_GSM TelephonyManager.PHONE_TYPE_CDMA READ_PHONE_STATE
getSubscriberId()	Return the subscriber id (IMSI) of the device READ_PHONE_STATE
getNeighboringCellInfo()	Returns a list of NeighboringCellInfo class which represents the neighboring cell information if available otherwise the function returns null ACCESS_COARSE_UPDATES

Fragments With Navigation bar and Tabs:
Activity_main.xml

```
<android.support.v4.view.ViewPager xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/pager"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

</android.support.v4.view.ViewPager>
```

Frgament_games.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"
   android:background="#ff8400" >

   <TextView android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:text="Design Games Screen"
      android:textSize="20dp"
      android:layout_centerInParent="true"/>

</RelativeLayout>
```

Fragment_movies.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"
   android:background="#17df0d">

   <TextView android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:text="Design Movies Screen"
      android:textSize="20dp"
      android:layout_centerInParent="true"/>

</RelativeLayout>
```

Fragment_toprated.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"
   android:background="#fa6a6a" >

   <TextView android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:text="Design Top Rated Screen"
      android:textSize="20dp"
      android:layout_centerInParent="true"/>


</RelativeLayout>
```

MainActivity.Java

```
package androindian.tabsswipe;

import android.app.ActionBar;
import android.app.ActionBar.Tab;
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.support.v4.view.ViewPager;

public class MainActivity extends FragmentActivity implements
                ActionBar.TabListener {

        private ViewPager viewPager;
        private TabsPagerAdapter mAdapter;
        private ActionBar actionBar;
        // Tab titles
        private String[] tabs = { "Top Rated", "Games", "Movies" };
```

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

	// Initilization
	viewPager = (ViewPager) findViewById(R.id.pager);
	actionBar = getActionBar();
	mAdapter = new TabsPagerAdapter(getSupportFragmentManager());

	viewPager.setAdapter(mAdapter);
	actionBar.setHomeButtonEnabled(false);
	actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

	// Adding Tabs
	for (String tab_name : tabs) {
		actionBar.addTab(actionBar.newTab().setText(tab_name)
				.setTabListener(this));
	}

	/**
	 * on swiping the viewpager make respective tab selected
	 * */
	viewPager.setOnPageChangeListener(new ViewPager.OnPageChangeListener() {

		@Override
		public void onPageSelected(int position) {
			// on changing the page
			// make respected tab selected
			actionBar.setSelectedNavigationItem(position);
		}

		@Override
		public void onPageScrolled(int arg0, float arg1, int arg2) {
		}

		@Override
		public void onPageScrollStateChanged(int arg0) {
		}
	});
}

@Override
public void onTabReselected(Tab tab, FragmentTransaction ft) {
}

@Override
public void onTabSelected(Tab tab, FragmentTransaction ft) {
	// on tab selected
	// show respected fragment view
	viewPager.setCurrentItem(tab.getPosition());
}
```

```
		@Override
		public void onTabUnselected(Tab tab, FragmentTransaction ft) {
		}

}
```

TabpagerAdapter.java

```
	package androindian.tabsswipe.adapter;

	import info.androidhive.tabsswipe.GamesFragment;
	import info.androidhive.tabsswipe.MoviesFragment;
	import info.androidhive.tabsswipe.TopRatedFragment;
	import android.support.v4.app.Fragment;
	import android.support.v4.app.FragmentManager;
	import android.support.v4.app.FragmentPagerAdapter;

	public class TabsPagerAdapter extends FragmentPagerAdapter {

		public TabsPagerAdapter(FragmentManager fm) {
			super(fm);
		}

		@Override
		public Fragment getItem(int index) {

			switch (index) {
			case 0:
				// Top Rated fragment activity
				return new TopRatedFragment();
			case 1:
				// Games fragment activity
				return new GamesFragment();
			case 2:
				// Movies fragment activity
				return new MoviesFragment();
			}

			return null;
		}

		@Override
		public int getCount() {
			// get item count - equal to number of tabs
			return 3;
		}

}
```

Gamesfragment.java

```
	package androindian.tabsswipe;

	import info.androidhive.tabsswipe.R;
	import android.os.Bundle;
```

```
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class GamesFragment extends Fragment {

	@Override
	public View onCreateView(LayoutInflater inflater, ViewGroup container,
			Bundle savedInstanceState) {

		View rootView = inflater.inflate(R.layout.fragment_games,
				container, false);


		return rootView;
	}
}
```

MoviesFragment.Java

```
package androindian.tabsswipe;

import info.androidhive.tabsswipe.R;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class MoviesFragment extends Fragment {

	@Override
	public View onCreateView(LayoutInflater inflater, ViewGroup container,
			Bundle savedInstanceState) {

		View rootView = inflater.inflate(R.layout.fragment_movies, container, false);

		return rootView;
	}

}
```

TopratedFragment.java

```
package androindian.tabsswipe;

import info.androidhive.tabsswipe.R;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class TopRatedFragment extends Fragment {
```

```
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {

            View rootView = inflater.inflate(R.layout.fragment_top_rated, container, false);

            return rootView;
        }
}
```

Manifestfile.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="info.androidhive.tabsswipe"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="11"
      android:targetSdkVersion="17" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name="info.androidhive.tabsswipe.MainActivity"
         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>
```

65.6. Bluetooth

Android supports to work with different types of networks like Bluetooth, WIFI,..etc , and more it supports to develop socket programming.

Checking Network status of Android Mobile:

AndroidManifest.xml :

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

CheckNetworkStatusActivity :

```
public class CheckNetworkStatusActivity extends Activity {

   /** Called when the activity is first created. */
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void onClick(View v){
            chkConnectionStatus();
    }
    void chkConnectionStatus(){
            //getting ConnctivityManager Service
            ConnectivityManager connMgr = (ConnectivityManager)
            this.getSystemService(Context.CONNECTIVITY_SERVICE);
            // Getting WIFI network
            final android.net.NetworkInfo wifi =
            connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
            // Getting 3G network
            final android.net.NetworkInfo mobile =
            connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
            // Ensuring WIFI network is available or not
            if( wifi.isAvailable() ){
            Toast.makeText(this, "Wifi" , Toast.LENGTH_LONG).show();
            }
            // Ensuring 3G is Available or not
            if( mobile.isAvailable() ){
            Toast.makeText(this,"Mobile 3G ",Toast.LENGTH_LONG).show();
            }
            else{
    Toast.makeText(this,"No Network",Toast.LENGTH_LONG).show();
            }
            }
}
```

Checking WIFI Status :

AndroidManifest.xml :

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.UPDATE_DEVICE_STATS"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
```

WifiStatusActivity.java

```
public class WifiStatusActivity extends Activity {
  /** Called when the activity is first created. */
        private WifiManager wifiManager;
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
        //Get the Wifi service of device
    wifiManager = (WifiManager) this.getSystemService(
                Context.WIFI_SERVICE);
        //Check the wifi is currently turned on or turned off
    if(wifiManager.isWifiEnabled()){
        // Turn on/off our wifi
        wifiManager.setWifiEnabled(false);
        Toast.makeText(this,"Wifi Enabled",Toast.LENGTH_LONG).show();
    }else{
        wifiManager.setWifiEnabled(true);
        Toast.makeText(this,"Wifi Disabled",Toast.LENGTH_LONG).show();
    }
  }
}
```

Introduction to Bluetooth :

The Android platform includes support for the Bluetooth network stack, which allows a device to wirelessly exchange data with other Bluetooth devices. The application framework provides access to the Bluetooth

functionality through the Android Bluetooth APIs. These APIs let applications wirelessly connect to other Bluetooth devices, enabling point-to-point and multipoint wireless features.

Using the Bluetooth APIs, an Android application can perform the following:

- Scan for other Bluetooth devices
- Query the local Bluetooth adapter for paired Bluetooth devices
- Establish RFCOMM channels
- Connect to other devices through service discovery
- Transfer data to and from other devices
- Manage multiple connections

All of the Bluetooth APIs are available in the android.bluetooth package.

BluetoothSocket

Represents the interface for a Bluetooth socket (similar to a TCP Socket). This is the connection point that allows an application to exchange data with another Bluetooth device via InputStream and OutputStream.

BluetoothServerSocket

Represents an open server socket that listens for incoming requests (similar to a TCP ServerSocket). In order to connect two Android devices, one device must open a server socket with this class. When a remote Bluetooth device makes a connection request to the this device, the BluetoothServerSocket will return a connected BluetoothSocket when the connection is accepted.

BluetoothDevice

Represents a remote Bluetooth device. Use this to request a connection with a remote device through a BluetoothSocket or query information about the device such as its name, address, class, and bonding state.

BluetoothAdapter

Represents the local Bluetooth adapter (Bluetooth radio). The BluetoothAdapter is the entry-point for all Bluetooth interaction. Using this, you can discover other Bluetooth devices, query a list of bonded (paired) devices, instantiate a BluetoothDevice using a known MAC address, and create a BluetoothServerSocket to listen for communications from other devices.

BluetoothHeadset

Provides support for Bluetooth headsets to be used with mobile phones. This includes both Bluetooth Headset and Hands-Free (v1.5) profiles.

Bluetooth Permission :

```
<manifest ... >
 <uses-permission android:name="android.permission.BLUETOOTH" />

 <uses-permission android:name="android.permission.BLUETOOTH_ADMIN">
   </uses-permission>
 ...
</manifest>
```

Bluetooth Status :

AndroidManifest.xml :

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
```

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <ToggleButton
      android:id="@+id/toggleButton1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:onClick="onClick" />
</LinearLayout>
```

BlueToothActivity.java

```
public class BlueToothActivity extends Activity {
  ToggleButton tb;
  BluetoothAdapter mBluetoothAdapter;
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     tb = (ToggleButton) findViewById(R.id.toggleButton1);
  }
  public void onClick(View v) {
           if((tb).isChecked()){
                  // getting blue tooth object
              mBluetoothAdapter = BluetoothAdapter.
                              getDefaultAdapter();
              // cofirming bluetooth supported by the device or not
              if (mBluetoothAdapter == null) {
                 System.out.println("Device does not support " +
```

```
                    "Bluetooth");
            }
            // enable the bluetooth if disabled
            if (!mBluetoothAdapter.isEnabled()) {
                //Enable bluetooth
              Intent enableBtIntent = new Intent(
                              BluetoothAdapter.ACTION_REQUEST_ENABLE);
              startActivityForResult(enableBtIntent, 1);
            }
            //Creating intent to discover the other bluetooth devices
            Intent discoverableIntent = new Intent(
                              BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
            discoverableIntent.putExtra(
                              BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION,
                              3000);//available time
            startActivity(discoverableIntent);
          }
          else{
            Toast.makeText(getApplicationContext(),
                              "Bluetooth turned off",Toast.LENGTH_SHORT).show();
            //get bluetooth object,if bluetooth is enable making disable.
            mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
            if (mBluetoothAdapter.isEnabled()) {
                //disable the bluetooth
              if (mBluetoothAdapter != null) {
                  mBluetoothAdapter.disable();
              }
            }
        }
    }
}
```

Graphics :

What is View Class, its important ?

This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for *widgets*, which are used to create interactive UI components (buttons, text fields, etc.).

All of the views in a window are arranged in a single tree. You can add views either from code or by specifying a tree of views in one or more XML layout files. There are many specialized subclasses of views that act as controls or are capable of displaying text, images, or other content.

A View object handles its own measurement, layout, drawing, focus change, scrolling, and key/gesture interactions for the rectangular area of the screen in which it resides.

What is Canvas Class its important and methods ?

The Canvas class holds the "draw" calls. To draw something, you need 4 basic components: A Bitmap to hold the pixels, a Canvas to host the draw calls (writing into the bitmap), a drawing primitive (e.g. Rect, Path, text, Bitmap), and a paint (to describe the colors and styles for the drawing).

void drawRect(Rect r, Paint paint) => Draw the specified Rect using the specified Paint.

void drawRoundRect(RectF rect, float rx, float ry, Paint paint) => Draw the specified round-rect using the specified paint.

void drawOval(RectF oval, Paint paint) => Draw the specified oval using the specified paint.

void drawLine(float startX, float startY, float stopX, float stopY, Paint paint) => Draw a line segment with the specified start and stop x,y coordinates, using the specified paint.

void drawCircle(float cx, float cy, float radius, Paint paint) => Draw the specified circle using the specified paint.

…etc for more methods refer API documentation.

Steps to Draw 2D Graphics :

Step-1) Define a class is-a type of View

Step-2) Create paint object, define paint properties in constructor.

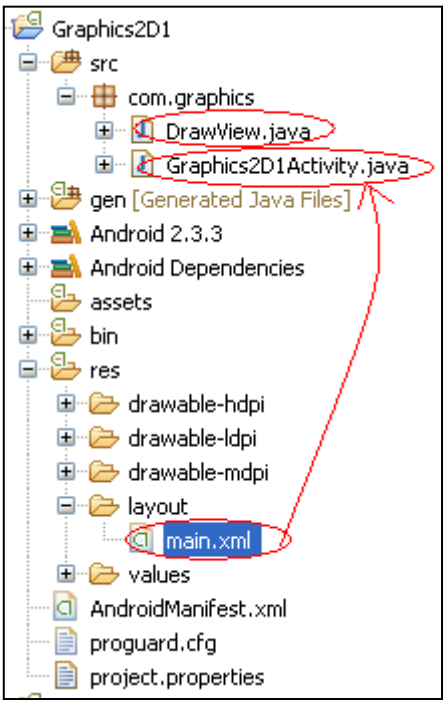
Step-3) Override onDraw() method, and draw the 2D graphics using canvas drawXXX() methods.

Implementing Touch Listener for android App to draw graphics :

Step-1) Define a class is-a type of View class, with OnTouchListener implementation

Step-2) Override the onDraw() , onTouch() methods.

Example : Develop Android app to draw graphics on Android ?



main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello" />
         </LinearLayout>
```

Graphics2D1Activity.java

```
public class Graphics2D1Activity extends Activity {
   DrawView drawView;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      drawView = new DrawView(this);
      drawView.setBackgroundColor(Color.WHITE);
      setContentView(drawView);
```

```
    }
        }
```

DrawView.java

```
public class DrawView extends View {
   Paint paint = new Paint();
   public DrawView(Context context) {
      super(context);
      paint.setColor(Color.RED);
      paint.setStrokeWidth(10);
   }
   @Override
   public void onDraw(Canvas canvas) {
        canvas.drawLine(0, 0, 200, 200, paint);
        canvas.drawLine(200, 0, 0, 200, paint);
        //c.drawRect(100, 100, 200, 200, paint); draws rectangle
        //canvas.drawCircle(75, 75, 75, paint);
   }
}
```

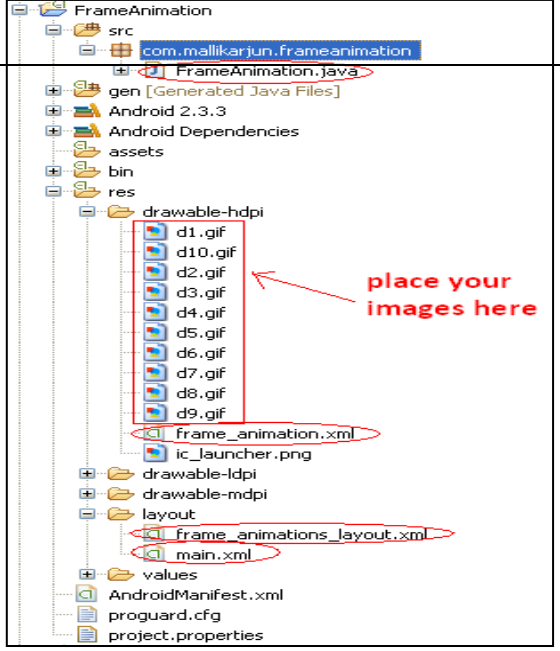
Animations

Android supports three types of animation:

- Frame-by-frame animation
  A series of frames is drawn one after the other at regular intervals by loading a series of Drawable resources one after the other.
- Layout animation
  Animate views inside a container view such as lists and tables.
- View animation
  Animate any general-purpose view.
  The types #2 and #3 are a type of tweeting animation which involves the drawing in between the key drawings. The idea is that knowing the beginning state and ending state of a drawing allows us to vary certain aspect of the drawing in time. This varying aspect could be color, position, size, rotation, and so on. In other words, we tell Android to perform a series of simple transformations to the content of a View.

Both animation types can be used in any View object to provide simple rotating timers, activity icons, and other useful UI elements. Tweened animation is handled by android.view.animation. Frame-by-frame animation is handled by the AnimationDrawable class.

ii) Frame –by- frme animation :



In Android, we can get frame-by-frame animation through a class in the graphics package called AnimationDrawable. We can tell from its name that it is like any other drawable that can work as background for any view. For example the background bitmaps are represented as Drawables.

The class AnimationDrawable, in addition to being a Drawable, can take a list of other Drawable resources like images and render them at specified intervals. This class is really a thin wrapper around the animation support provided by the basic Drawable class.

To make use of the AnimationDrawable class, let's start with a set of Drawable resources in the /res/drawable subdirectory.

Example :

main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello" />
</LinearLayout>
```

frame_animations_layout.xml

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

```
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView android:id="@+id/textViewId1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/frame"/>
  <Button
    android:id="@+id/startFAButtonId"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/start"/>
  <ImageView
    android:contentDescription="@string/desc"
    android:id="@+id/animationImage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```

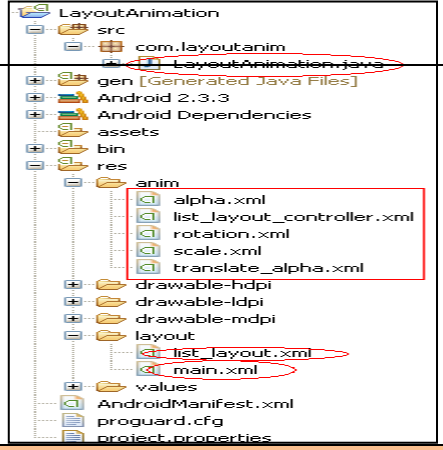
frame_animation.xml

```
<animation-list
  xmlns:android="http://schemas.android.com/apk/res/android"
        android:oneshot="false">
        <item android:drawable="@drawable/d1" android:duration="50" />
        <item android:drawable="@drawable/d2" android:duration="50" />
        <item android:drawable="@drawable/d3" android:duration="50" />
        <item android:drawable="@drawable/d4" android:duration="50" />
        <item android:drawable="@drawable/d5" android:duration="50" />
        <item android:drawable="@drawable/d6" android:duration="50" />
        <item android:drawable="@drawable/d7" android:duration="50" />
        <item android:drawable="@drawable/d8" android:duration="50" />
        <item android:drawable="@drawable/d9" android:duration="50" />
        <item android:drawable="@drawable/d10" android:duration="50" />
```

```
</animation-list>
```

FrameAnimation.java

```
public class FrameAnimation extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.frame_animations_layout);
      this.setupButton();
    }

    private void setupButton() {
    Button b = (Button)this.findViewById(R.id.startFAButtonId);
    b.setOnClickListener(
      new Button.OnClickListener() {
        public void onClick(View v) {
        parentButtonClicked(v);
        }
      });
    }
    private void parentButtonClicked(View v) {
      animate();
    }
    private void animate() {
      ImageView imgView = (ImageView)findViewById(R.id.animationImage);
      //imgView.setVisibility(ImageView.VISIBLE);
      imgView.setBackgroundResource(R.drawable.frame_animation);
      AnimationDrawable frameAnimation =(AnimationDrawable) imgView.getBackground();
      frameAnimation.start();
      /*if (frameAnimation.isRunning()) {
        frameAnimation.stop();
```




```
            }
            else {
              frameAnimation.stop();
              frameAnimation.start();
            }*/
          }
}
```

iii) Layout Animation

- ✓ We'll use layout animation with the ListView and GridView, which are the two most commonly-used controls in Android. Specifically, we'll use layout animation to add visual effects to the way each item in a ListView or GridView is displayed. Actually, we can use this type of animation on all controls derived from a ViewGroup.
- ✓ Layout animation works by applying tweening to each view that is part of the layout being animated. Here are four types of tweening animation:
  - Scale animation (growing or shrinking) : We use this type of animation to make a view smaller or larger either on x axis or on the y axis. We can also specify the pivot point around which we want the animation to take place.
  - Rotate animation (rotations): We use this to rotate a view around a pivot point by a certain number of degrees.
  - Translate animation (position changes): We use this to move a view along the x or y axis.
  - Alpha animation (transparency changes): We use this to change the transparency of a view.

Once we have a ListView, we can attach an animation to it so that each list item will go through that animation.

We can define both the individual animation and the mediator in XML files in the /res/anim subdirectory. Once we have the mediator XML file, we can use that file as an input to the ListView in its own XML layout definition.

main.xml

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

```
      android:layout_height="wrap_content"
      android:text="@string/hello" />
          </LinearLayout>
```

list_layout.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <ListView
    android:id="@+id/list_view_id"
    android:persistentDrawingCache="animation|scrolling"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layoutAnimation="@anim/list_layout_controller" />
          </LinearLayout>
```

list_layout_controller.xml

```
<layoutAnimation xmlns:android="http://schemas.android.com/apk/res/android"
          android:delay="30%"
          android:animationOrder="reverse"
          android:animation="@anim/translate_alpha" />
```

alpha.xml

```
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/accelerate_interpolator"
  android:fromAlpha="0.0"
  android:toAlpha="1.0"
            android:duration="5000" />
```

rotation.xml

```
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/accelerate_interpolator"
  android:fromDegrees="0.0"
```

```
  android:toDegrees="360"
  android:pivotX="70%"
  android:pivotY="70%"
          android:duration="5000" />
```

scale.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator">
  <scale
    android:fromXScale="1"
    android:toXScale="1"
    android:fromYScale="0.1"
    android:toYScale="1.0"
    android:duration="1000"
    android:pivotX="50%"
    android:pivotY="50%"
    android:startOffset="100" />
        </set>
```

translate_alpha.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator">
  <translate android:fromYDelta="-100%" android:toYDelta="0"
    android:duration="500" />
  <alpha android:fromAlpha="0.0" android:toAlpha="1.0"
    android:duration="1000" />
        </set>
        LayoutAnimation.java
public class LayoutAnimation extends Activity {
                @Override
                public void onCreate(Bundle savedInstanceState){
                  super.onCreate(savedInstanceState);
                  setContentView(R.layout.list_layout);
```

```
        setupListView();
    }

    private void setupListView(){
      String[] listItems = new String[] {
        "Item 1", "Item 2", "Item 3",
        "Item 4", "Item 5", "Item 6",
      };

      ArrayAdapter<String> listItemAdapter =
      new ArrayAdapter<String>(this
        ,android.R.layout.simple_list_item_1
        ,listItems);

      ListView lv = (ListView)this.findViewById(R.id.list_view_id);
      lv.setAdapter(listItemAdapter);
    }
}
```

Now that we have the layout needed for the activity, we can write the code for the activity to load the layout file so we can generate our UI.

Activity_Main.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/txtMessage"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is fadeout animation"
      android:layout_centerInParent="true"
      android:textSize="25dp"/>

   <Button android:id="@+id/btnStart"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Start Animation"
```

```
    android:layout_marginTop="30dp"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="20dp"/>

</RelativeLayout>
```

Create one xml file in drawable folder withbelow code

Anim.xml

```
<?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="1.0"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:toAlpha="0.0" />


</set>
```

MainActivity.java

```
package com.example.animation;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.Animation.AnimationListener;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity implements AnimationListener {

        TextView txtMessage;
        Button btnStart;

        // Animation
        Animation animFadeOut;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
         // TODO Auto-generated method stub
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);

         txtMessage = (TextView) findViewById(R.id.txtMessage);
         btnStart = (Button) findViewById(R.id.btnStart);

         // load the animation
         animFadeOut = AnimationUtils.loadAnimation(getApplicationContext(),
          R.drawable.anim);
```

```
        // set animation listener
        animFadeOut.setAnimationListener(this);

        // button click event
        btnStart.setOnClickListener(new View.OnClickListener() {

         @Override
         public void onClick(View v) {
          // start the animation
          txtMessage.startAnimation(animFadeOut);
         }
        });

       }

       @Override
       public void onAnimationEnd(Animation animation) {
        // Take any action after completing the animation

        // check for fade out animation
        if (animation == animFadeOut) {
         Toast.makeText(getApplicationContext(), "Animation Stopped",
           Toast.LENGTH_SHORT).show();
        }

       }

       @Override
       public void onAnimationRepeat(Animation animation) {
        // TODO Auto-generated method stub

       }

       @Override
       public void onAnimationStart(Animation animation) {
        // TODO Auto-generated method stub

       }

}
```

Overview of Playstore Account

Register for Google Play Android account

Google Play is the premier store for distributing Android apps. It's preinstalled on more than 400 million devices worldwide, a number growing by more than a million every day. Android users have downloaded more than **25 billion apps** from Google Play, growing at a rate of more than 1.5 billion per month.
When you publish on Google Play, you put your apps in front of Android's huge base of active customers, in more than 130 countries and territories across the world.

Google Play is a central part of the Android experience. New users personalize their devices with apps, games, and other Google Play content. Existing users return regularly to see what's trending and new. Downloading new apps is

extremely convenient and fast— Google Play pushes apps to the user's devices instantly, over the air. No cable or sync is ever needed.

Register for a publisher account

The first step is to visit the Google Play Android Developer Console and register for a publisher account.

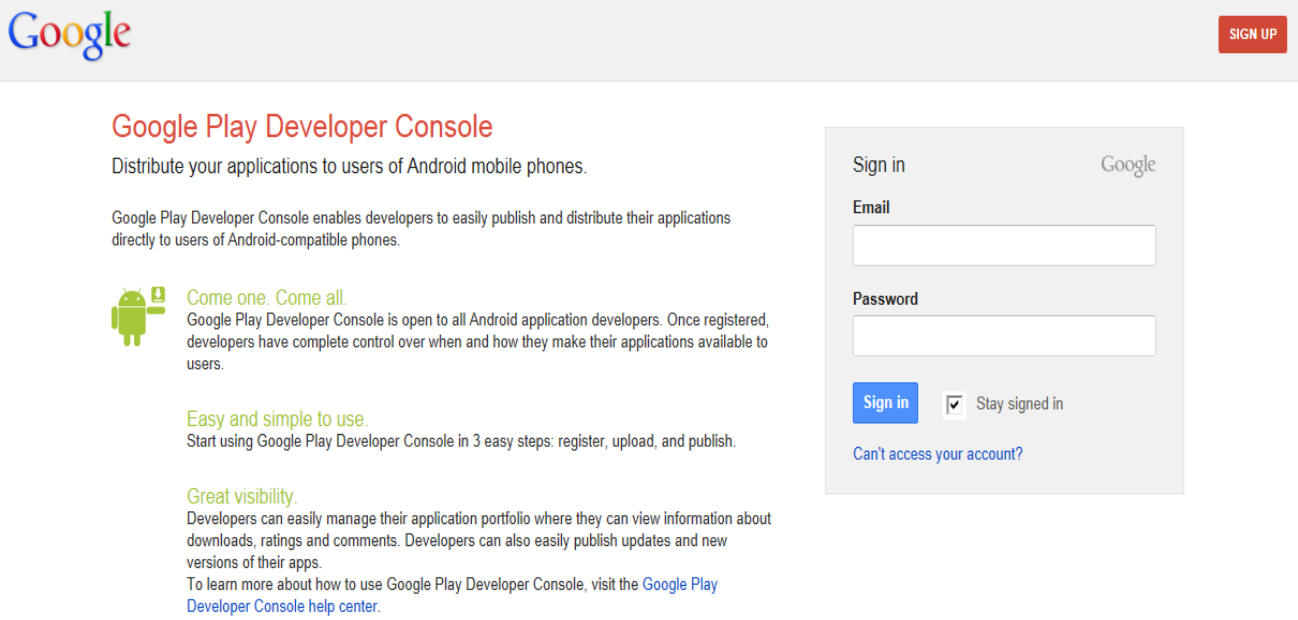
Tips:

- You need a Google account to register. You can create one during the process. It is recommended not to use your main Google account for this purpose, so you can keep this account separate from your other Google services.
- If you are an organization, consider registering a new Google account rather than using a personal account.
- Review the developer countries and merchant countries where you can distribute and sell apps.

1. Visit the Google Play Android Developer Console at

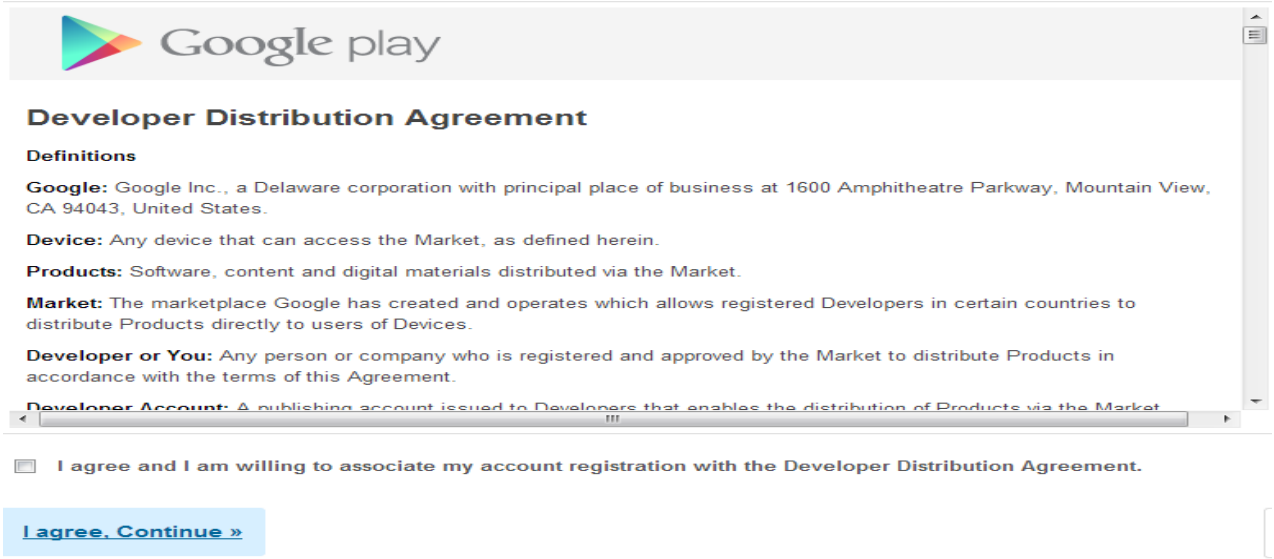
https://play.google.com/apps/publish/

Login to your Google Account, or open a new one by clicking the signup button on the top right.

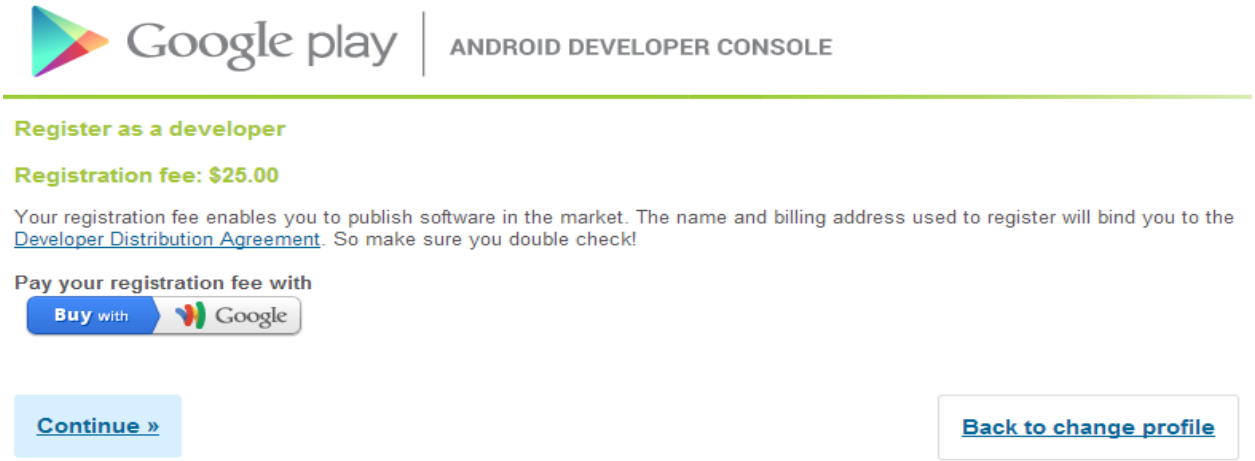


2. Enter basic information about your developer identity — developer name, email address, and so on. You can modify this information later.

3. Read and accept the Developer Distribution Agreement that applies to your country or region. Note that apps and store listings that you publish on Google Play must comply with the Developer Program Policies and US export law.



4. Pay a $25 USD registration fee using Google Checkout. If you don't have a Google Checkout account, you can quickly set one up during the process.



When your registration is verified, you'll be notified at the email address you specified during registration.

Publishing your Android App

You can set up to start publishing on Google Play in only a few minutes. Here's how you do it:

1. Register for a Google Play publisher account (If you already have a Google Play account, you can skip this step).
You need a Google account to register. You can create one during the process.

If you are an organization, consider registering a new Google account rather than using a personal account.

Visit the Google Play Developer Console

1.1 Enter basic information about your developer identity — developer name, email address, and so on. You can modify this information later.

1.2 Read and accept the Developer Distribution Agreement that applies to your country or region. Note that apps and store listings that you publish on Google Play must comply with the Developer Program Policies and US export law,

1.3 Pay a $25 USD registration fee using Google Checkout. If you don't have a Google Checkout account, you can quickly set one up during the process.

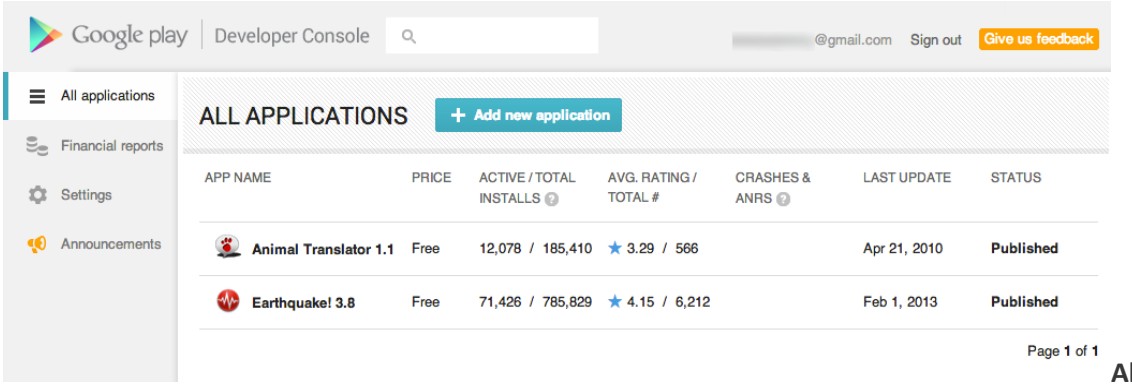
1.4 When your registration is verified, you'll be notified at the email address you specified during registration.

2. Distribute your App as a paid App
Users that would like to sell their Apps needs to set up a Google Checkout Merchant account, this process is not required for users that would like to distribute their App for free, however please note that if you will choose the App to be free you will not be able to change it to a paid one later on.

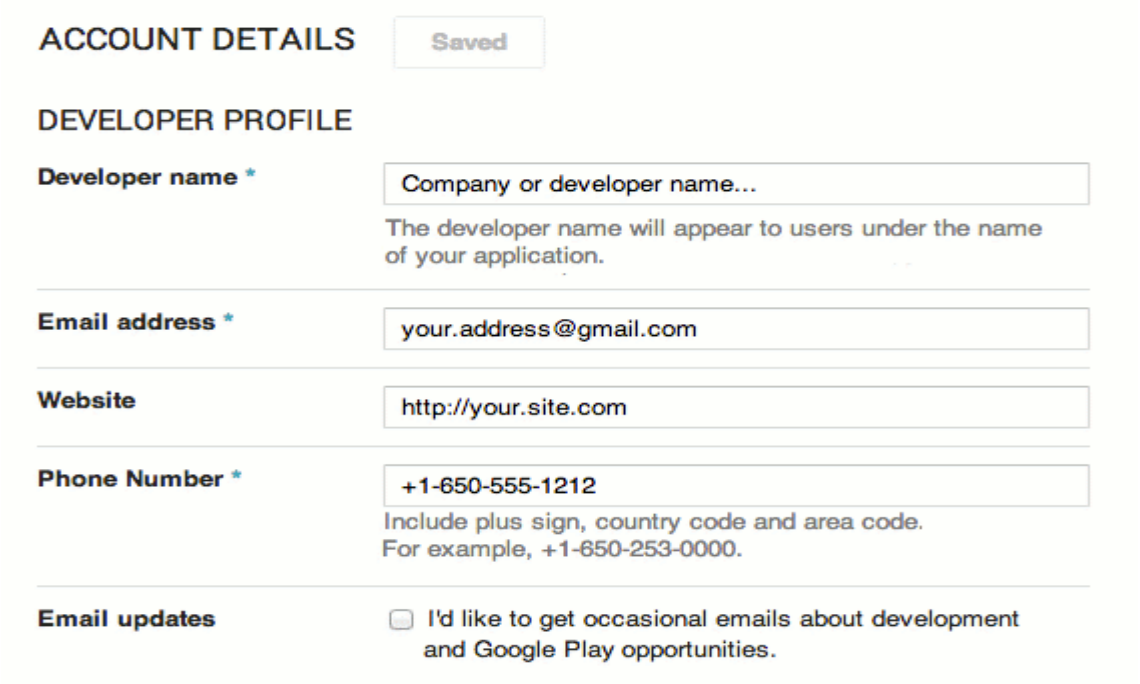
To set up a Merchant account from the Developer Console:

Sign in to your Google Play Developer Console at https://play.google.com/apps/publish/
Open Financial reports on the side navigation.
Click Setup a Merchant Account now.
This takes you to the Google Wallet site to sign up as a Merchant; you'll need information about your business available to complete this step.

3. The Developer Console
Once you've registered and received verification by email, you can sign in to your Google Play Developer Console, which will be the home for your app publishing operations and tools on Google Play. This sections below introduce a few of the key areas you'll find in the Developer Console.



All applications page: Gives you a quick overview of your apps, lets you jump to stats, reviews, and product details, or upload a new app.



Account details page: Specifies your developer identity and contact information, accounts for app testing, and more.

1.3.1. YOUR ACCOUNT DETAILS

The account details page is where you specify basic information about yourself or your company in a developer profile. The information in your developer profile is important because it identifies you to Google Play and also to your customers.

During registration you must provide the information for your profile, but you can go back at any time to edit the information and change your settings.

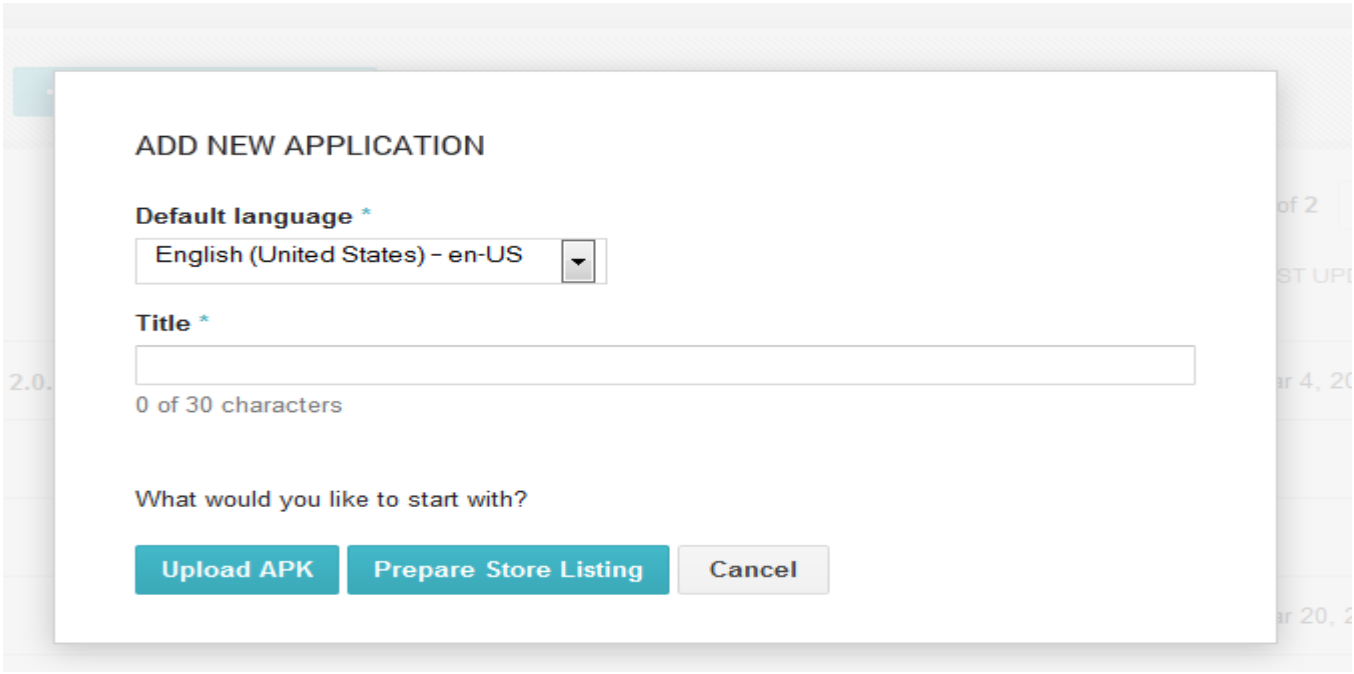
Your developer profile contains:

- Your developer name — the name you want to show users on your store listing page and elsewhere on Google Play.
- Your developer contact information — how Google can contact you if needed (this information isn't exposed to users).
- Your developer website URL — shown to users on your store listing page so they can learn more about your company or products.

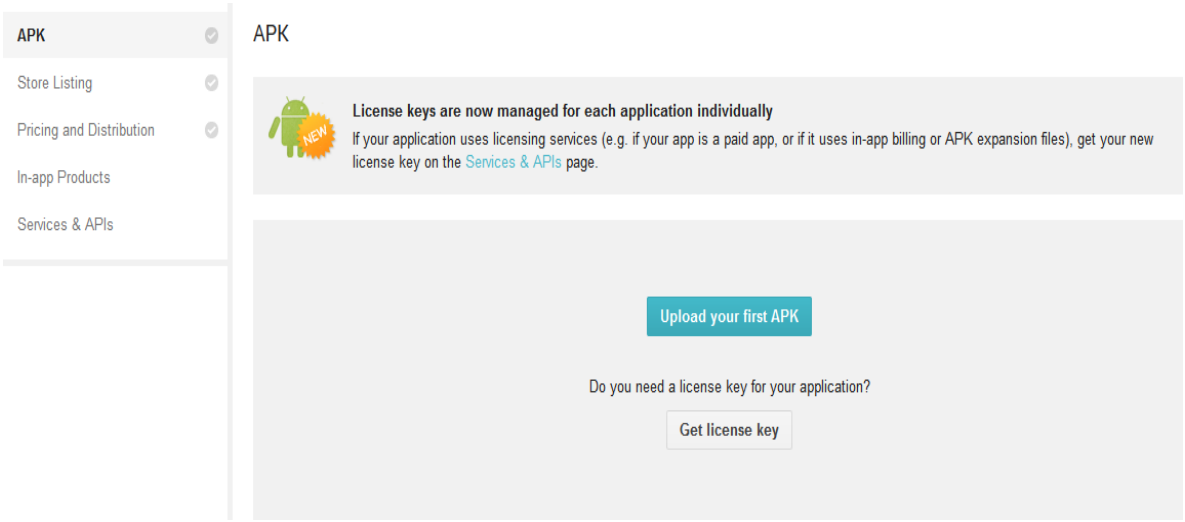
4. Publishing a new App

✓ Please login to the Android developer console and click the "Add new Application" button.

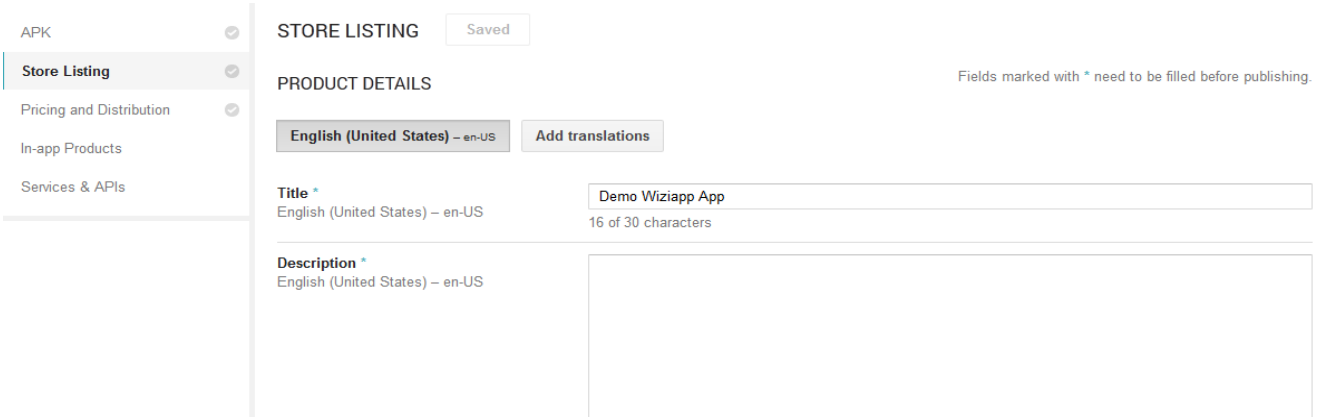
✓ On the popup screen choose your App default language and your App name and click the "Upload APK" button.



4.3 Click the “Upload your first APK” button and upload the APK file which you have created on the Wiziapp WordPress plugin control panel.



4.4 **Choose the “Store Listing” tab on the top left side menu:**



4.4.1 **Fill your App description.**

If you would like to add a disclaimer about the App required permissions, please copy the paragraph below:

Permissions:

- Storage – Allows us to save the App caching resources in order for the App to load faster.
- Network Communication – Enables the App to access the web.
- Your Account – This permission is needed to register your device ID with Google Cloud Messaging for the push notifications service to be enabled. The App will never attempt to place a phone call or use it in any other way.
- System tools – Enable us to add a shortcut to your home screen.
- Development tools – Enable us to add a shortcut to your notification center.
- Phone calls – This permission is needed to register your device ID with Google Cloud Messaging for the push notifications service to be enabled.

4.4.2 **Upload your App screenshots –** You can create your App screenshots from one of the following ways:

-Users that have access to an Android device, just send the APK file to your Email, open it on the device, click install and take screenshots of your preferred App screens.

-Users that have an iPhone device, browse to your Wiziapp powered webapp on your iPhone browser, click to save the App to your iPhone home screen, open it from the iPhone home screen, take the screenshots of your preferred App screens and Email it to your desktop. You will also need to resize it for the Android screenshots requirements.

4.4.3 Upload a High-res icon, **512 x 512, 32-bit PNG.**

4.4.4 Select your application type as “Application”.

4.4.5 Select your App category.

4.4.6 Select your App content ratings.

4.4.7 Fill in your App contact details.

4.4.8 If you wish to provide a privacy policy URL for this application, please enter it or check “the Not submitting a privacy policy URL at this time”.

4.4.9 Click to save (see top button).

4.5 Choose the “Pricing and Distribution” tab on the top left menu.

4.5.1 Choose the App price to be Free or Paid.

4.5.2 Choose the countries to distribute the App.

4.5.3 Check the last 2 checkbox on the bottom of the “Pricing and Distribution” screen as follows:

-Content guidelines

-US export laws

4.5.4 Click to save (see top button).

4.6 Choose the “Services & APIs ” tab on the left menu:

For activating the push notification service – Click the “Link sender ID” and enter your API Key which can be found on the Wiziapp WordPress plugin control panel – “Publish to Google Play”.

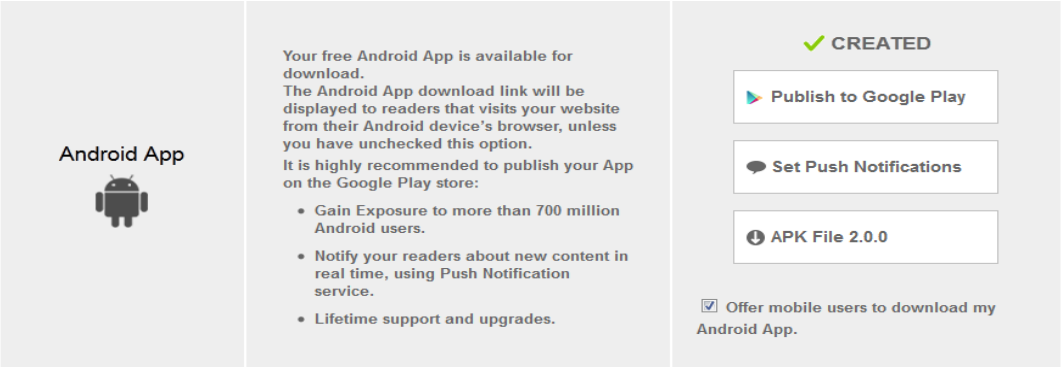
5. Click to publish the App by changing the top right button dropdown menu from “Ready to Publish” to “Publish the App”.

Updating your Android App

In order to publish an update to your Android App, please follow the below steps:

Creating a new APK file:

On the Wiziapp WordPress plugin control panel, edit your App settings and click the “Save Changes” button. Please wait while the system creates your new APK file.



Uploading the new APK file to the Google Play market:

1. Login to your Android developer console and choose the App you would like to update.
2. Choose the APK tab on the left, then click the “Upload new APK to production”, save it and republish the App.

The update will be available on the Google Play market within several hours.

Material Design

Material design is a set of rule built by Google that guide how to develop an Android app. They can be applied not only to Android apps but also to web design. In the process of developing an app, Android provides some libraries that help developers to implement the main material guide line rules. The most important libraries are:

- com.android.support:appcompat-v7:23.1.0
- com.android.support:design:23.1.0

After all, these two libraries are imported by default when a developer starts a new project using Android Studio.

One important aspect of an app is represented by the color schema. Material design rules describe how to cho0se colors.
Let us suppose we create a simple Android project and let us follow the steps to implement an Android app following Material design rules.

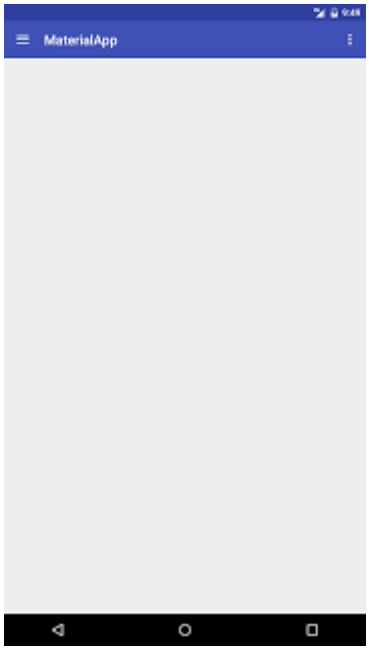
21.1 *Material Design: Colors*

The first step is choosing the color schema for our app. For this purpose there is a great website that can be used to create the color schema according to material design rules.

After the colors are selected we can download colors.xml:

```
<resources>
   <color name="primary">#3F51B5</color>
   <color name="primary_dark">#303F9F</color>
   <color name="primary_light">#C5CAE9</color>
   <color name="accent">#03A9F4</color>
   <color name="primary_text">#212121</color>
   <color name="secondary_text">#727272</color>
   <color name="icons">#FFFFFF</color>
   <color name="divider">#B6B6B6</color>
</resources>
```

You can select the schema you like. The first result is shown in the picture below:



Now it's time to create our theme that uses the colors we selected before. The app should support the largest number of smart phones—not only those running Lollipop or later.

For this reason it is necessary to create two themes: one for the devices that run Android 5 or later and those that run pre-lollipop version.

So let's create two directories under the values:

style

style-v21

The first one is used by all smart phones running pre-Lollipop versions while the second folder is used by smart phones with OS starting from Lollipop.

In the first directory, style.xml we do:

```
<resources>
   <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">   <!-- Customize your theme here. -->
      <item name="colorPrimary">@color/primary</item>
      <item name="colorPrimaryDark">@color/primary_dark</item>
      <item name="colorAccent">@color/accent</item>
   </style>
   <style name="MyAppTheme" parent="@style/AppTheme" />
</resources>
```

While in the second directory we simply add:

```
<resources>
  <style name="MyAppTheme" parent="AppTheme">
    <item name="android:windowContentTransitions">true</item>
    <item name="android:windowAllowEnterTransitionOverlap">true</item>
    <item name="android:windowAllowReturnTransitionOverlap">true</item>
    <item name="android:windowSharedElementEnterTransition">@android:transition/move</item>
    <item name="android:windowSharedElementExitTransition">@android:transition/move</item>
  </style>
</resources>
```

Finally, in Manifest.xml, modify the file like so:

```
<application
    android:theme="@style/MyAppTheme" >
    ...
</application>
```