# Course: 305-02: Mobile Application Development - 1

## Unit-2: Setting up Android Environment:

### 2.1 Android Emulator

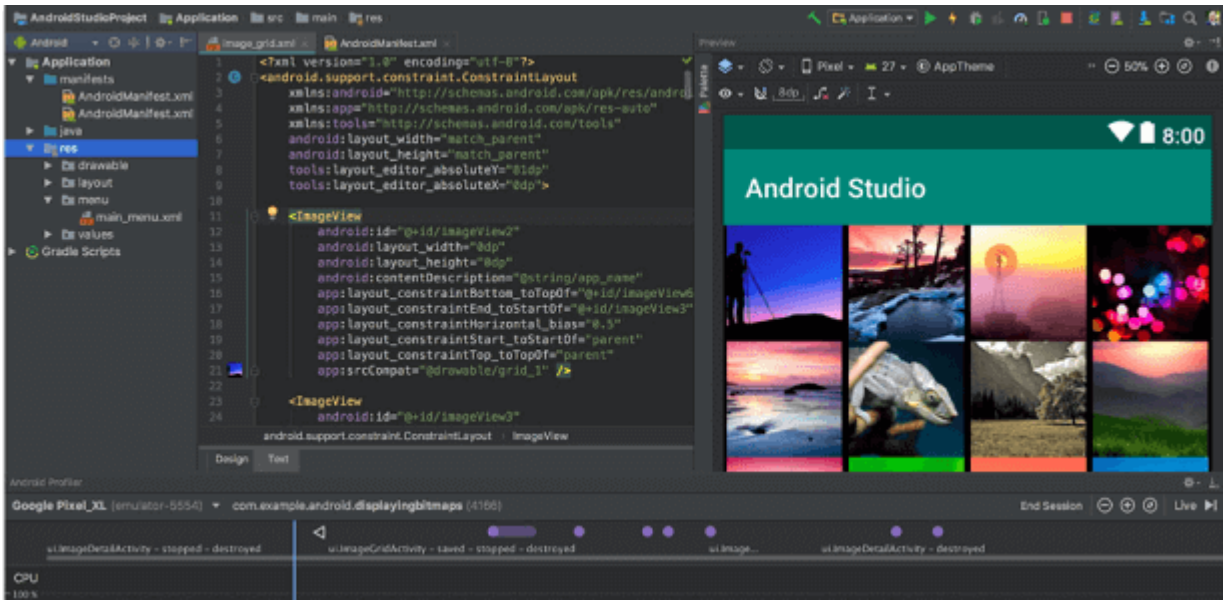
#### 2.1.1 Setting up JDK and Android Studio

*Android Studio*

*Android Studio is the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android apps.*

*Android Studio was announced on 16th May 2013 at the Google I/O conference as an official IDE for Android app development. It started its early access preview from version 0.1 in May 2013. The first stable built version was released in December 2014, starts from version 1.0.*

*Since 7th May 2019, Kotlin is Google's preferred language for Android application development. Besides this, other programming languages are supported by Android Studio.*



## *Features of Android Studio*

- *It has a flexible Gradle-based build system.*
- *It has a fast and feature-rich emulator for app testing.*
- *Android Studio has a consolidated environment where we can develop for all Android devices.*
- *Apply changes to the resource code of our running app without restarting the app.*
- *Android Studio provides extensive testing tools and frameworks.*
- *It supports C++ and NDK.*
- *It provides build-in supports for Google Cloud Platform. It makes it easy to integrate Google Cloud Messaging and App Engine.*

## *Android Studio Version History*

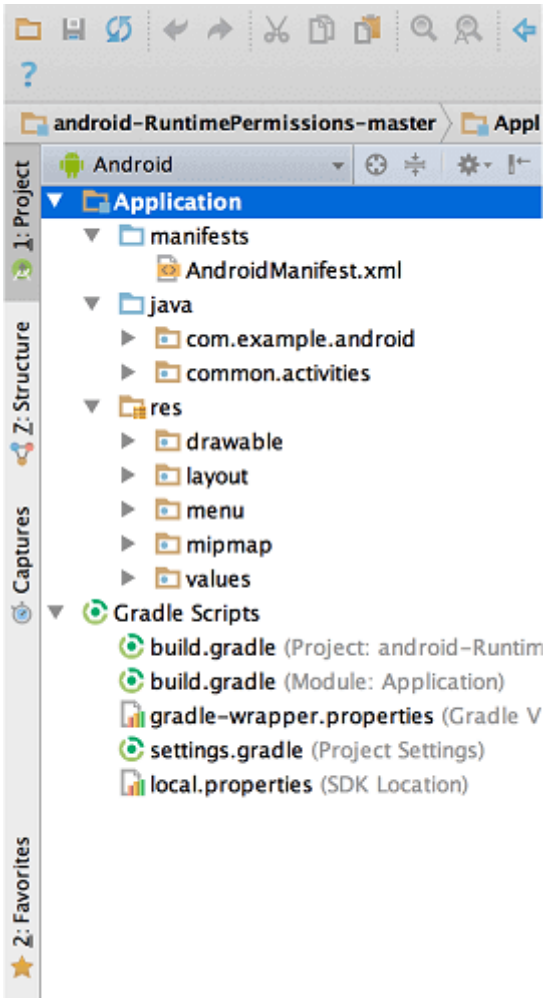
| Version | Release date |
|---|---|
| 1.0 | December 2014 |
| 1.1 | February 2015 |
| 1.2 | April 2015 |
| 1.3 | July 2015 |
| 1.4 | September 2015 |
| 1.5 | November 2015 |
| 2.0 | April 2016 |
| 2.1 | April 2016 |

| | |
|---|---|
| 2.2 | September 2016 |
| 2.3 | March 2017 |
| 3.0 | October 2017 |
| 3.1 | March 2018 |
| 3.2 | September 2018 |
| 3.3 | January 2019 |
| 3.4 | April 2019 |
| 3.5 | August 2019 |

## *Android Studio Project Structure*

*The Android Studio project contains one or more modules with resource files and source code files. These include different types of modules-*

- *Android app modules*
- *Library modules*
- *Google App Engine modules*




*By default, Android Studio displays our project files in the Android project view, as shown in the above image. This view is formed by modules to provide quick access to our project's key source files.*

*These build files are visible to the top-level under Gradle Scripts. And the app module contains the following folders:*

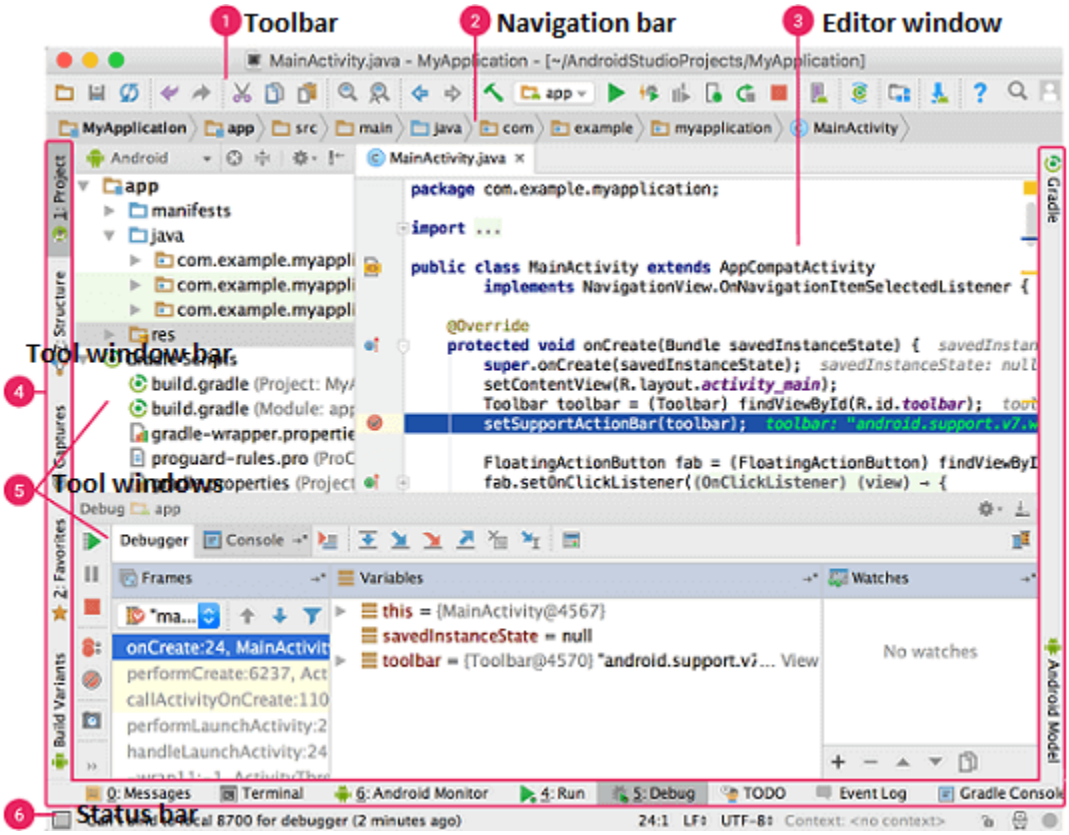
- ***manifests**: It contains the AndroidManifest.xml file.*
- ***java**: It contains the source code of Java files, including the JUnit test code.*

- *res: It contains all non-code resources, UI strings, XML layouts, and bitmap images.*

*We will see the actual file structure of the project by selecting the Project from the Project dropdown.*

## *Android Studio User Interface*

*The Android Studio main window contains the several logical areas which are shown in the below figure:*




1. *The toolbar provides us a wide range of actions, which includes running apps and launching Android tools.*
2. *The navigation bar helps in navigating our project and open files for editing. It gives a compact view of structure visible in the Project window.*

3. *The editor window is a space where we can create and modify our code. On the basis of the current file type, the editor can change. While viewing a layout file, the editor displays the Layout Editor.*
4. *The tool window bar runs around the outside the IDE window and contains buttons that allow as to expand and collapse individual tool windows.*
5. *The tool windows provide us access specific tasks like search, project management, version control, and more. We can expand and collapse them.*
6. *The status bar displays the status of our project and IDE itself, as well as any messages or warnings.*

*We are willing to organize the main window to give ourselves more screen space by moving or hiding toolbars and tool windows. We can also use keyboard shortcuts to access most of the IDE features.*

## *Android Studio Tool window*

*We can use keyboard shortcuts to open tool windows. The below table provides the list of shortcuts for the most common windows.*

| Tool window | Windows and Linux | Mac |
|---|---|---|
| Project | Alt+1 | Command+1 |
| Version Control | Alt+9 | Command+9 |
| Run | Shift+F10 | Control+R |
| Debug | Shift+F9 | Control+D |
| Logcat | Alt+6 | Command+6 |
| Return to Editor | Esc | Esc |
| Hide all Tool Windows | Control+Shift+F12 | Command+Shift+F12 |

## *Gradle build system*

*Gradle build used as the foundation of the build system in Android Studio. It uses more Android-specific capabilities provided by the Android plugin for Gradle. This build system runs independently from the command line and integrated tool from the Android Studio menu. We can use build features for the following purpose:*

- *Configure, customize, and extend the build process.*
- *We can create multiple APKs from our app, with different features using the same project and modules.*
- *Reuse resource and code across source sets.*

### 2.1.2 Android SDK manager

To download and install latest android APIs and development tools from the internet, android provide us with android SDK manager. Android SDK Manager separates the APIs, tools and different platforms into different packages which you can download.

Android SDK manager comes with the Android SDK bundle. You can't download it separately. You can download the android sdk from here.

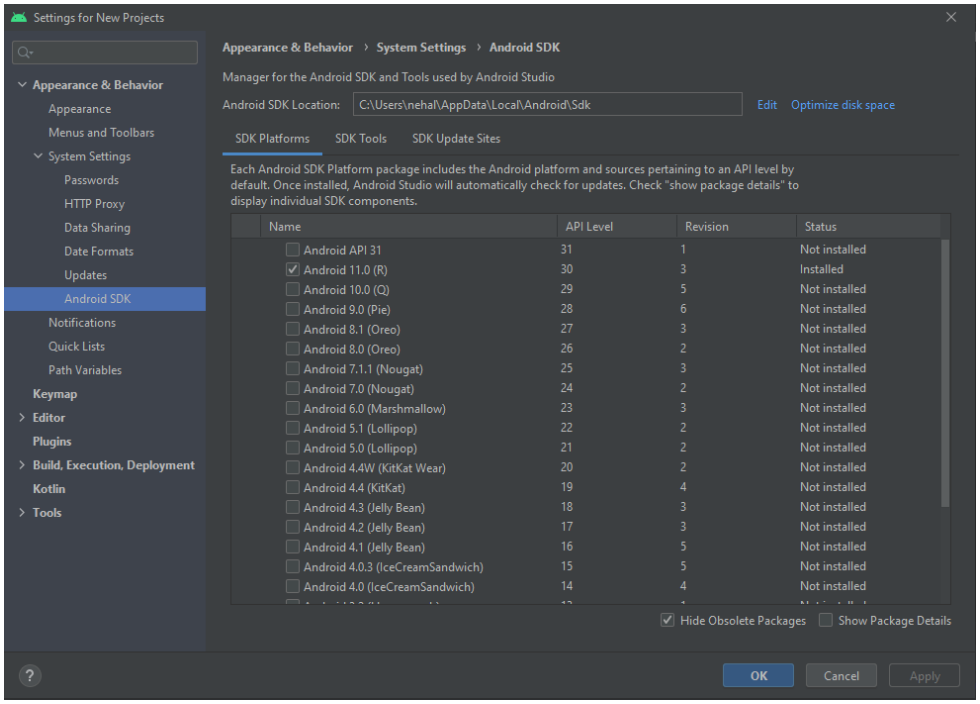
**Running Android SDK Manager**

Once downloaded, you can launch Android SDK Manager in one of the following ways –

Click **tools->Android-> SDK Manager** option in Eclipse.

Double Click on the **SDK Manager.exe** file in the Android SDK folder.

When it runs you will see the following screen –



*You can select which package you want to download by selecting the checkbox and then click Install to install those packages. By default SDK Manager keeps it up to date with latest APIs and other packages.*

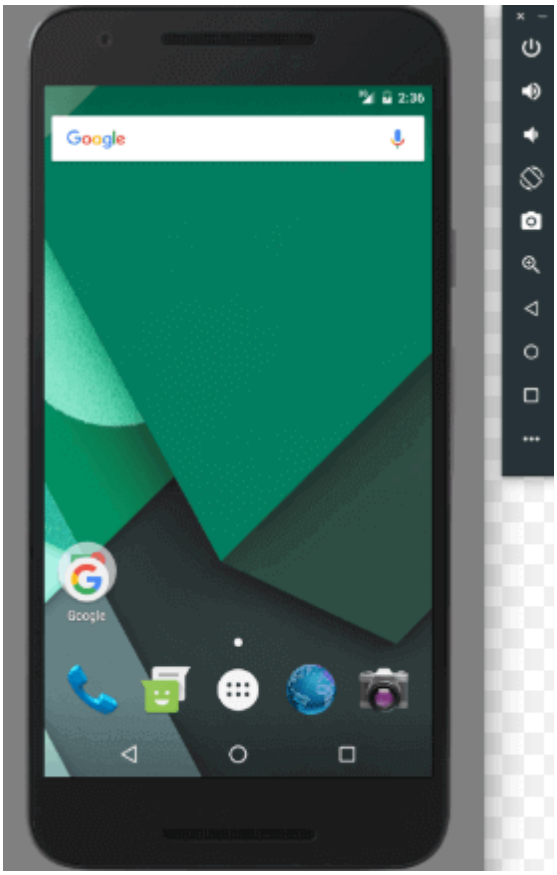
*Once you download the SDK, following packages are available but first three are necessary to run your SDK and others are recommended.*

| Sr.No | Package & Description |
|---|---|
| 1 | **SDK Tools**<br>This is necessary package to run your SDK. |
| 2 | **SDK Platform-tools**<br>This package will be installed once when you first run the SDK manager. |
| 3 | **SDK Platform**<br>At least one platform must be installed in your environment to run your application. |
| 4 | **System Image**<br>It's a good practice to download system images for all of the android versions so you can test your app on them with the Android Emulator. |
| 5 | **SDK Samples**<br>This will give you some sample codes to learn about android. |

*2.2 Creating Android Virtual Device (AVD)*

***Android Emulator***

*The Android emulator is an Android Virtual Device (AVD), which represents a specific Android device. We can use the Android emulator as a target device to execute and test our Android application on our PC. The Android emulator provides almost all the functionality of a real device. We can get the incoming phone calls and text messages. It also gives the location of the device and simulates different network speeds. Android emulator simulates rotation and other hardware sensors. It accesses the Google Play store, and much more*



*Testing Android applications on emulator are sometimes faster and easier than doing on a real device. For example, we can transfer data faster to the emulator than to a real device connected through USB.*

*The Android emulator comes with predefined configurations for several Android phones, Wear OS, tablet, Android TV devices.*

*Requirement and recommendations*

*The Android emulator takes additional requirements beyond the basic system requirement for Android Studio. These requirements are given below:*

*SDK Tools 26.1.1 or higher*

*64-bit processor*

*Windows: CPU with UG (unrestricted guest) support*

*HAXM 6.2.1 or later (recommended HAXM 7.2.0 or later)*

***Install the emulator***

*The Android emulator is installed while installing the Android Studio. However some components of emulator may or may not be installed while installing Android Studio. To install the emulator component, select the Android Emulator component in the SDK Tools tab of the SDK Manager.*

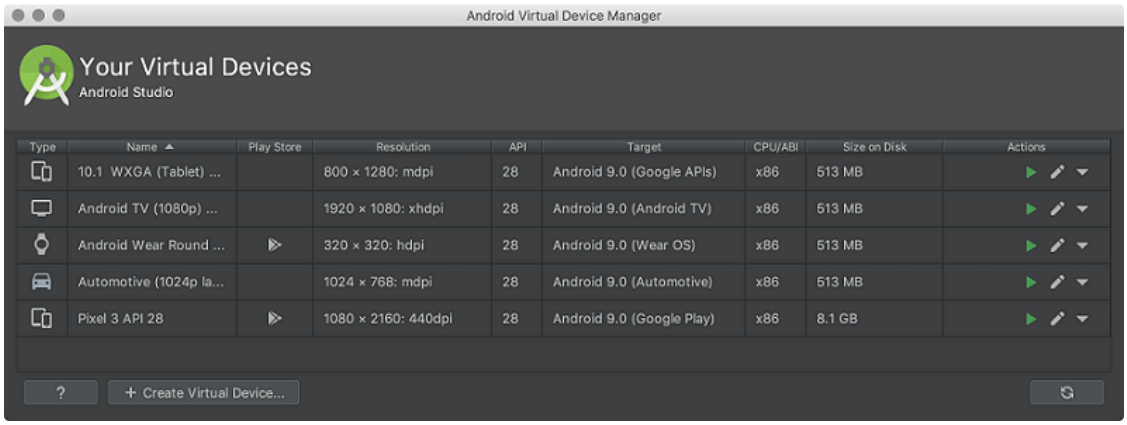
***Run an Android app on the Emulator***

*We can run an Android app form the Android Studio project, or we can run an app which is installed on the Android Emulator as we run any app on a device.*

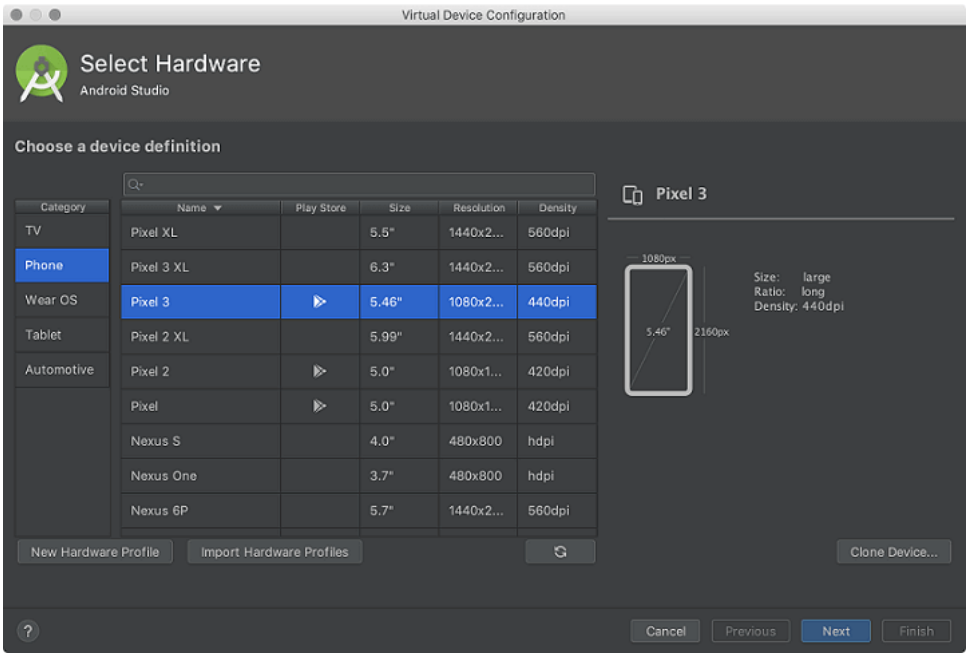
*To start the Android Emulator and run an application in our project:*

1. In Android Studio, we need to create an Android Virtual Device (AVD) that the emulator can use to install and run your app. To create a new AVD:-

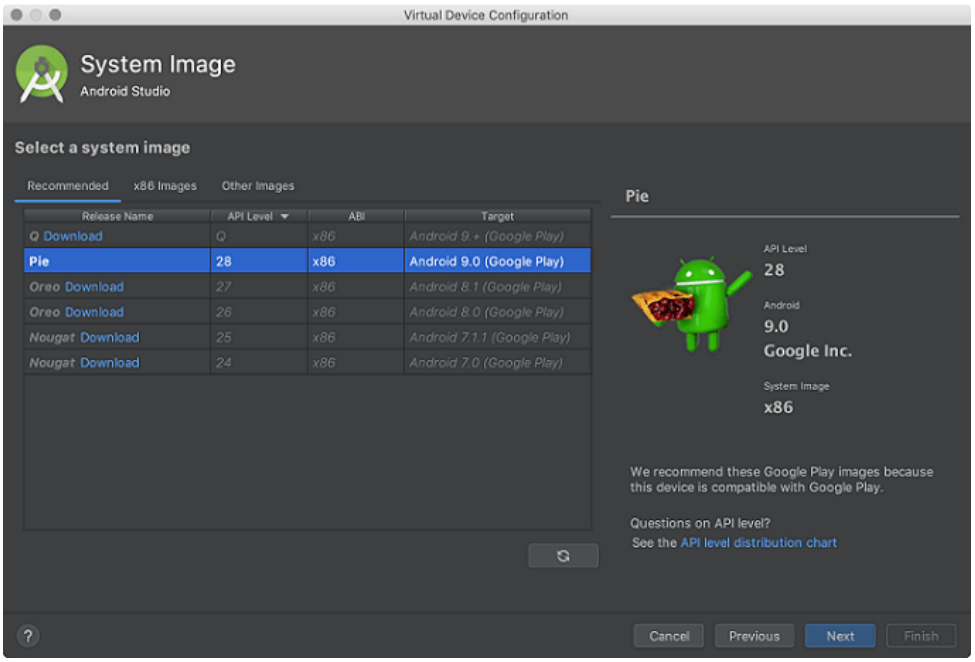
1.1 Open the AVD Manager by clicking Tools > AVD Manager.



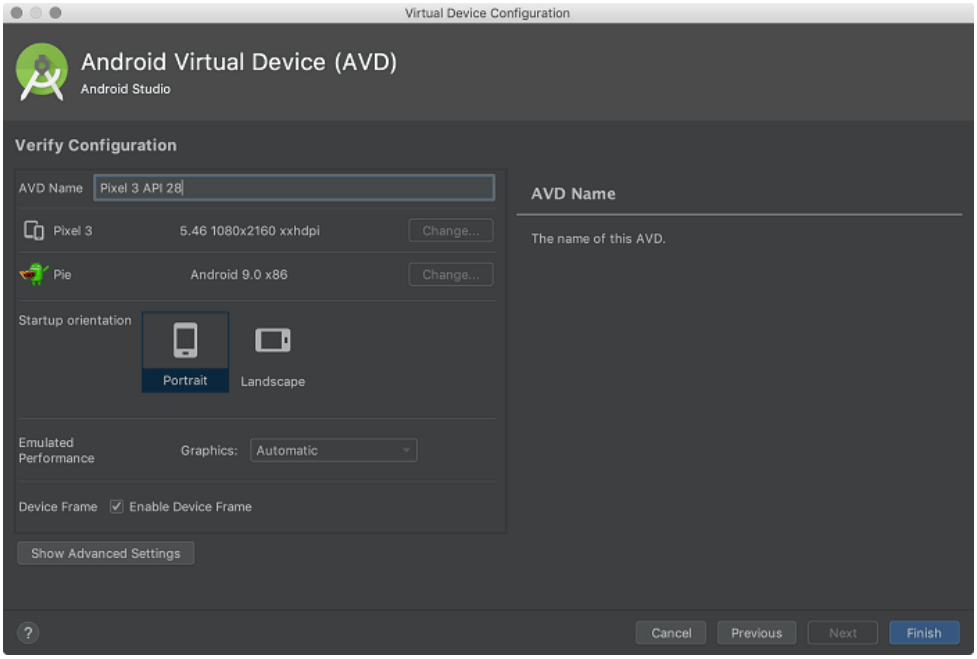

1.2 Click on Create Virtual Device, at the bottom of the AVD Manager dialog. Then Select Hardware page appears.




1.3 Select a hardware profile and then click Next. If we don?t see the hardware profile we want, then we can create or import a hardware profile. The System Image page appears.

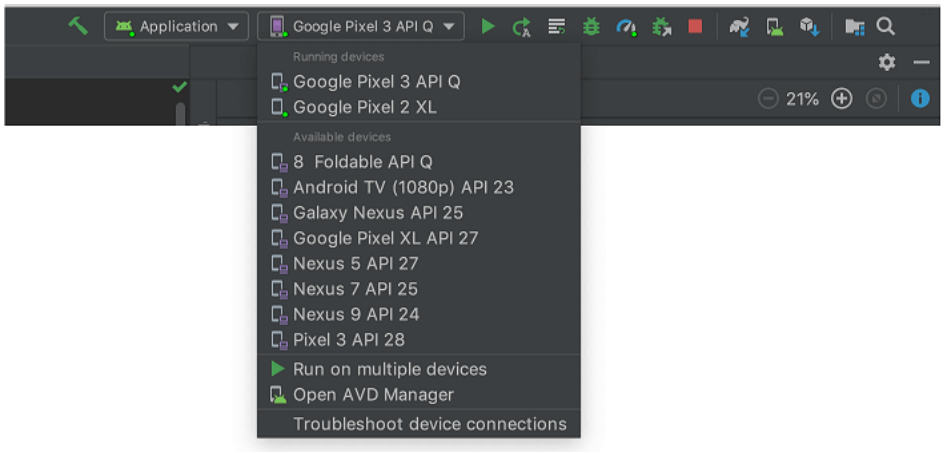



1.4 *Select the system image for the particular API level and click Next. This leads to open a Verify Configuration page.*




1.5 *Change AVD properties if needed, and then click Finish.*

2 *In the toolbar, choose the AVD, which we want to run our app from the target device from the drop-down menu.*




3 Click Run.

Launch the Emulator without first running an app

To start the emulator:

- Open the AVD Manager.
- Double-click an AVD, or click Run

While the emulator is running, we can run the Android Studio project and select the emulator as the target device. We can also drag an APKs file to install on an emulator, and then run them.

Run and stop an emulator, and clear data

From the Virtual Device page, we can perform the following operation on emulator:


- *To run an Android emulator that uses an AVD, double-click the AVD, or click Launch*
- *To stop the running emulator, right-click and select Stop, or click Menu ▼ and select Stop.*
- *If we want to clear the data from an emulator and return it to the initial state when it was first defined, then right-click an AVD and select Wipe Data. Or click menu ▼ and select Wipe Data.*

## *2.3 Creating first App:*

### *2.3.1 Activity*

***How to make android apps***

*In this page, you will know how to create the simple hello android application. We are creating the simple example of android using the Eclipse IDE. For creating the simple example:*

*1 Create the new android project*
*2 Write the message (optional)*
*3 Run the android application*

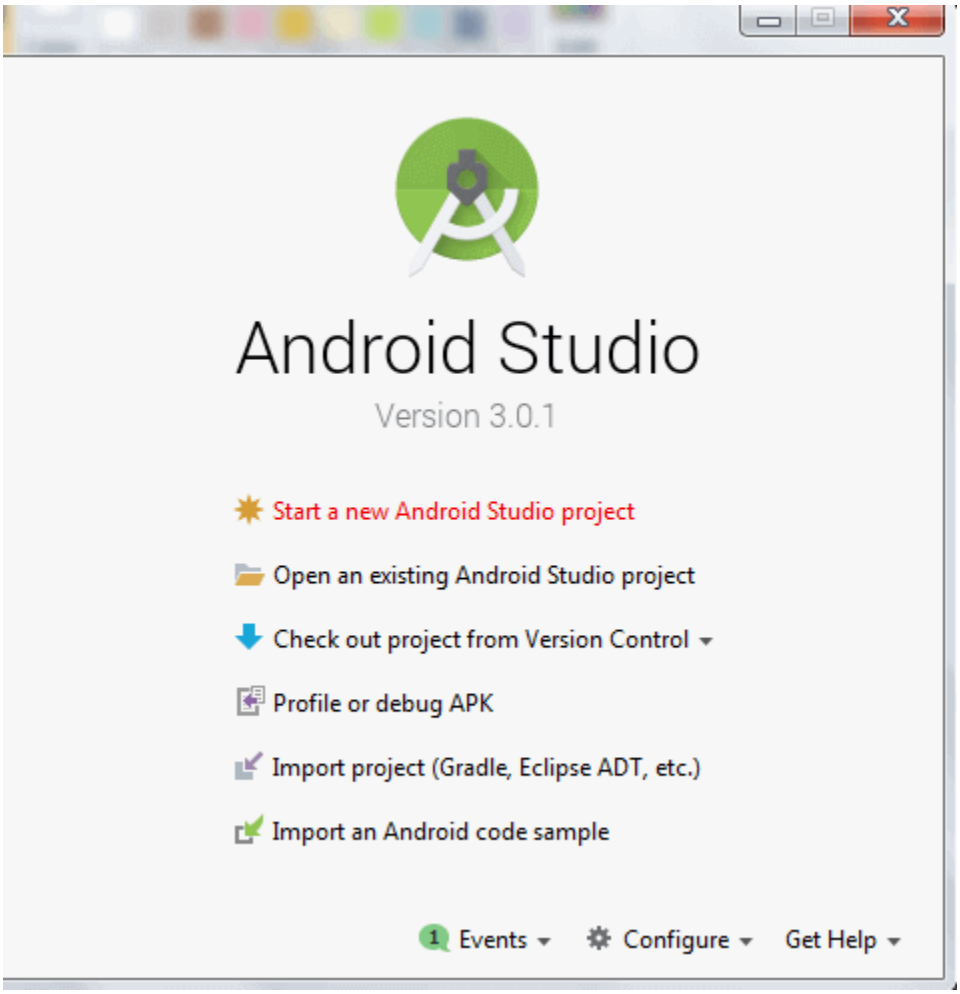
*Hello Android Example*

*You need to follow the 3 steps mentioned above for creating the Hello android application.*

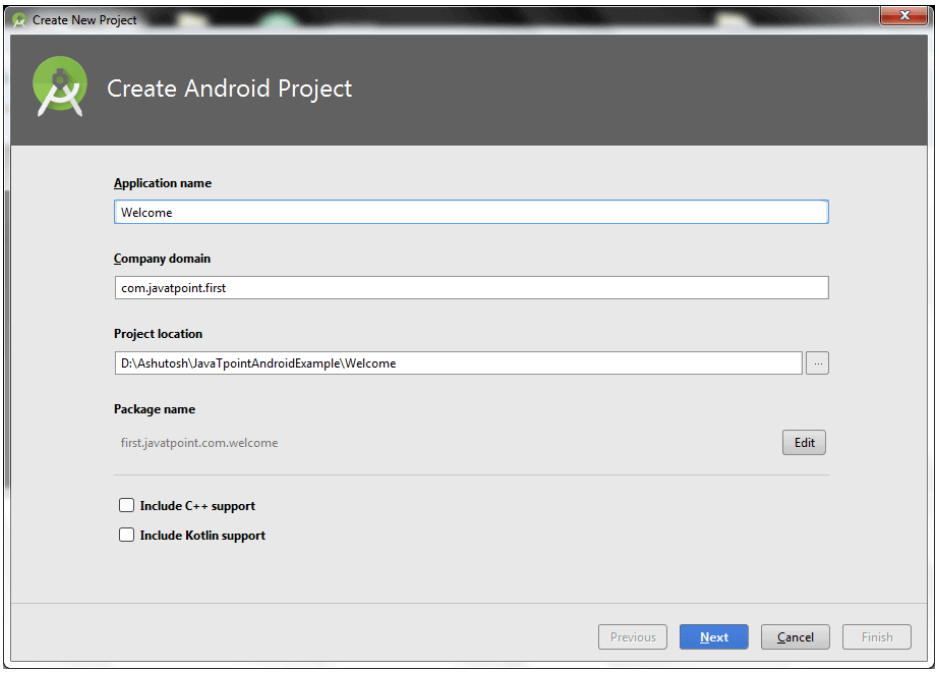
*1) Create the New Android project*

*For creating the new android studio project:*

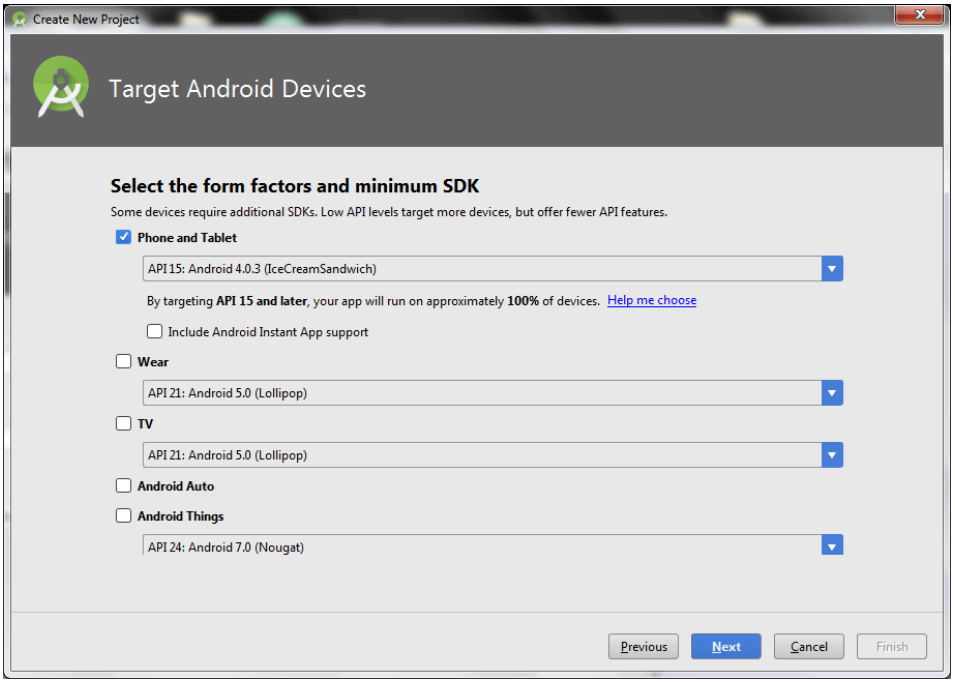
*1) Select Start a new Android Studio project*



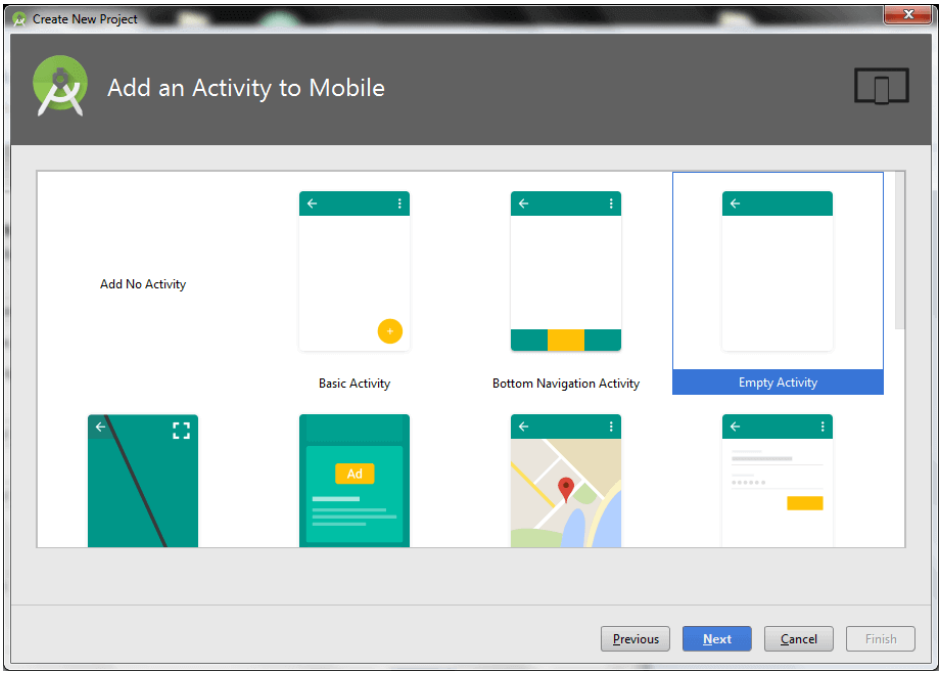

*2) Provide the following information: Application name, Company domain, Project location and Package name of application and click next.*



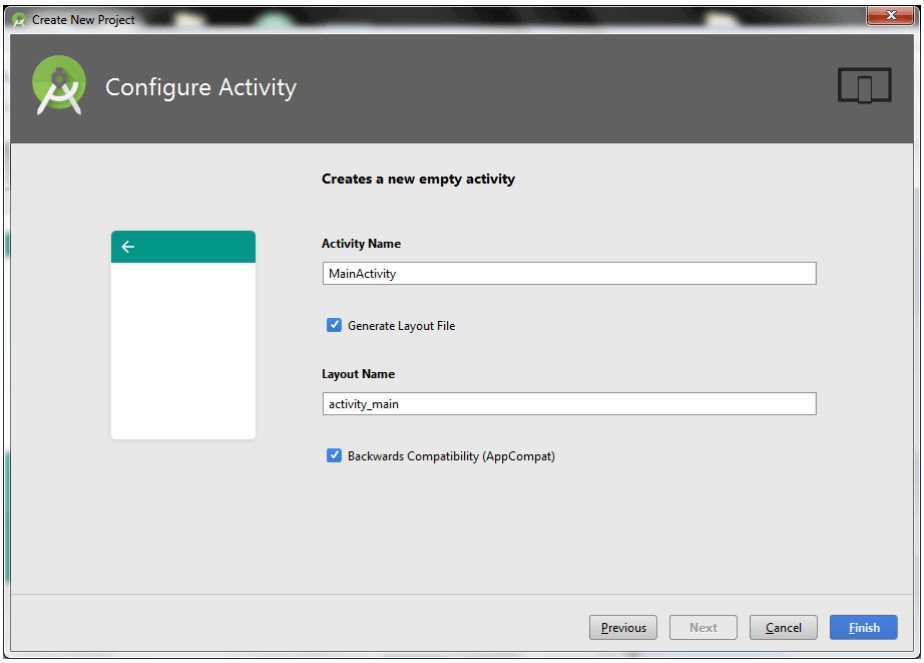

3) Select the API level of application and click next.




4) Select the Activity type (Empty Activity).

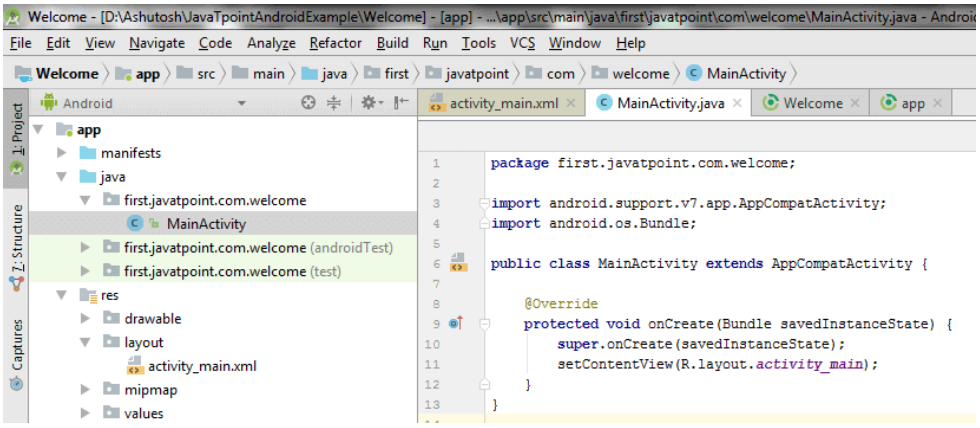



5) Provide the Activity Name and click finish.




After finishing the Activity configuration, Android Studio auto generates the activity class and other required configuration files.

Now an android project has been created. You can explore the android project and see the simple program, it looks like this:




*2) Write the message*

*File: activity_main.xml*

*Android studio auto generates code for activity_main.xml file. You may edit this file according to your requirement.*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="first.javatpoint.com.welcome.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Android!"
```

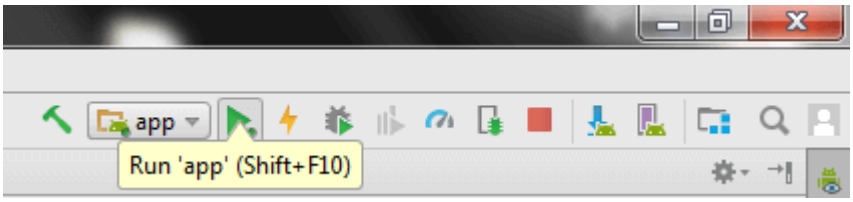
```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

*File: MainActivity.java*

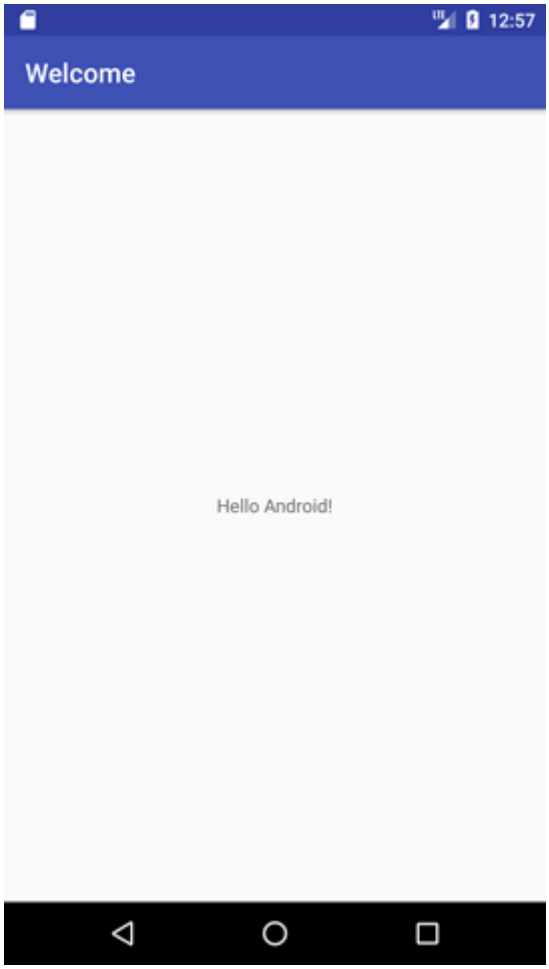
```
package first.test.com.welcome;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

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

*3) Run the android application*

*To run the android application, click the run icon on the toolbar or simply press Shift + F10.*



*The android emulator might take 2 or 3 minutes to boot. So please have patience. After booting the emulator, the android studio installs the application and launches the activity. You will see something like this:*



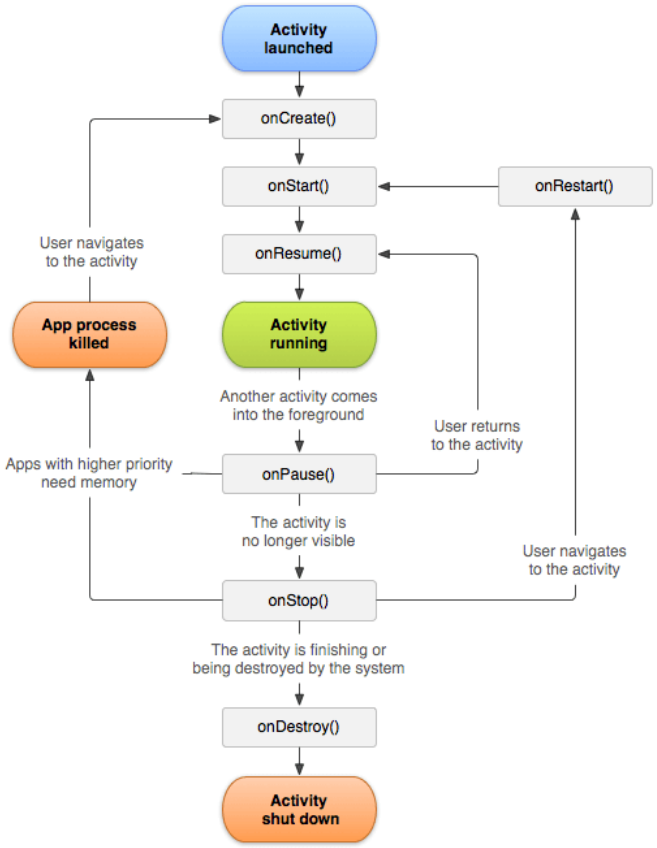

### *Android Activity Lifecycle*

*Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.*

*An activity is the single screen in android. It is like window or frame of Java.*

*By the help of activity, you can place all your UI components or widgets in a single screen.*

*The 7 lifecycle method of Activity describes how activity will behave at different states.*




***Android Activity Lifecycle methods***

1. ***onCreate*** *called when activity is first created.*
2. ***onStart*** *called when activity is becoming visible to the user.*
3. ***onResume*** *called when activity will start interacting with the user.*
4. ***onPause*** *called when activity is not visible to the user.*
5. ***onStop*** *called when activity is no longer visible to the user.*
6. ***onRestart*** *called after your activity is stopped, prior to start.*
7. ***onDestroy*** *called before the activity is destroyed.*

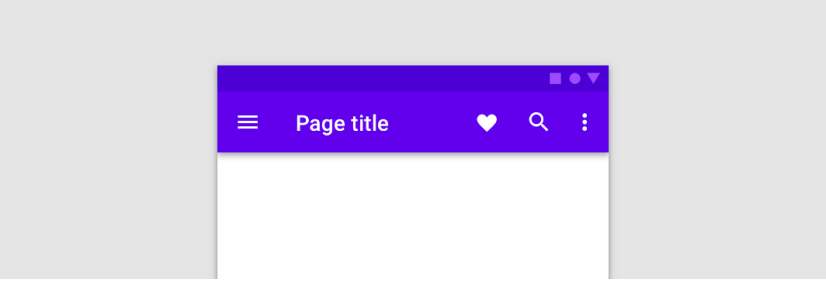
```
1. import android.app.Activity;
2. import android.os.Bundle;
3. import android.util.Log;
4.
5. public class MainActivity extends Activity {
6.
7.     @Override
8.     protected void onCreate(Bundle savedInstanceState) {
9.         super.onCreate(savedInstanceState);
```

```
10.         setContentView(R.layout.activity_main);
11.         Log.d("lifecycle","onCreate invoked");
12.     }
13.     @Override
14.     protected void onStart() {
15.         super.onStart();
16.         Log.d("lifecycle","onStart invoked");
17.     }
18.     @Override
19.     protected void onResume() {
20.         super.onResume();
21.         Log.d("lifecycle","onResume invoked");
22.     }
23.     @Override
24.     protected void onPause() {
25.         super.onPause();
26.         Log.d("lifecycle","onPause invoked");
27.     }
28.     @Override
29.     protected void onStop() {
30.         super.onStop();
31.         Log.d("lifecycle","onStop invoked");
32.     }
33.     @Override
34.     protected void onRestart() {
35.         super.onRestart();
36.         Log.d("lifecycle","onRestart invoked");
37.     }
38.     @Override
39.     protected void onDestroy() {
40.         super.onDestroy();
41.         Log.d("lifecycle","onDestroy invoked");
42.     }
43. }
```

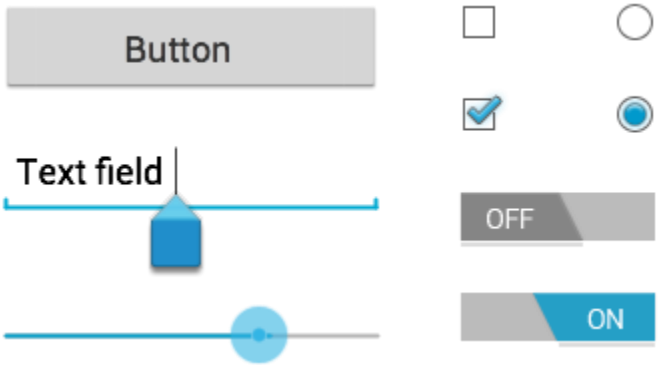
### 2.3.2 Layout

Android Layout Components

- Main Action Bar

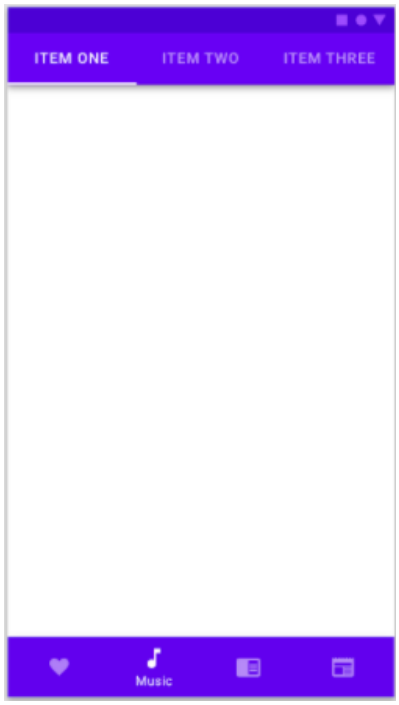



- View Control

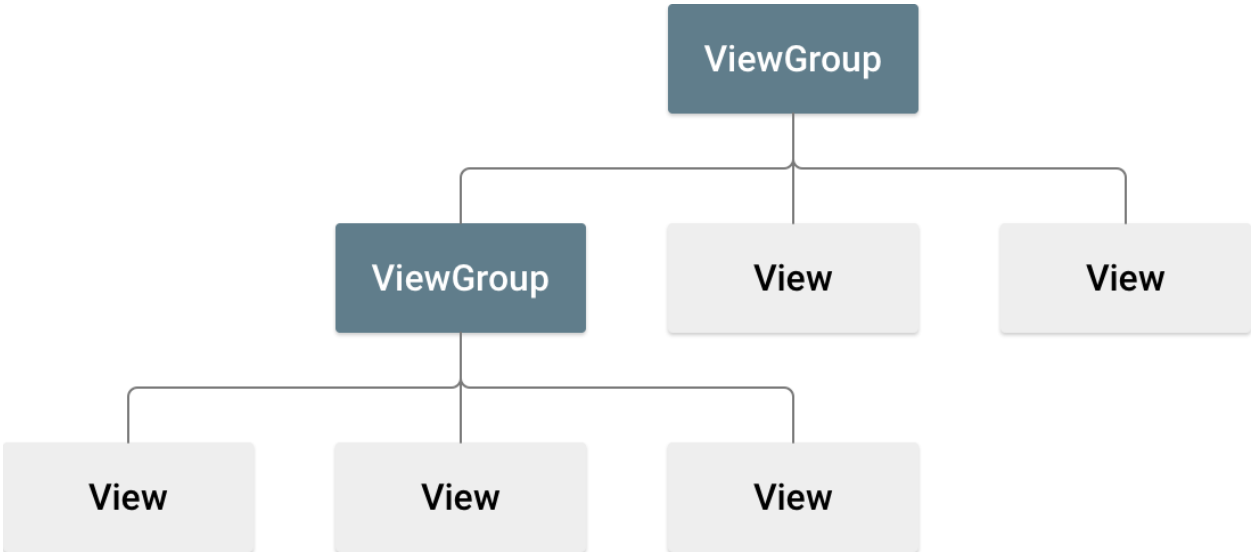



- Content Area

- Split Action Bar




*ViewGroup/View*



*Types of Layouts*

*- Liner Layout*

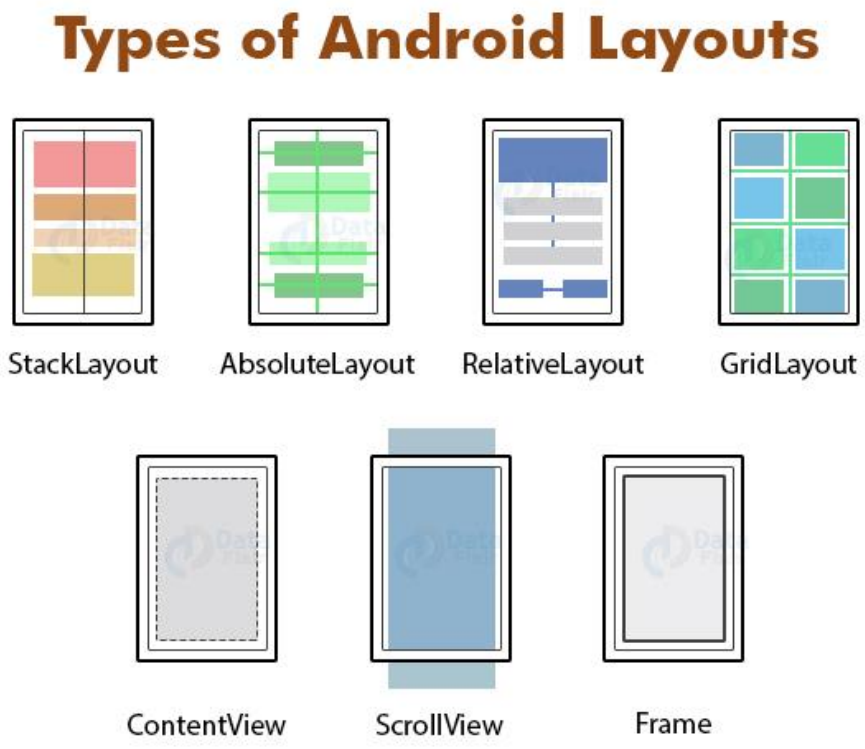
*- Absolute Layout*

*- Table Layout*

*- Frame Layout*

*- Relative Layout*

*- Constrained Layout*




*Unit Of Measurement*

*dp - Density Independent. 1dp ~ 1px on 160dpi screen*

*sp - Scale Independent Pixel*

*pt - point, 1/72 of an inch*

*px - pixel.*