MOBILE APPLICATION DEVELOPMENT

UNIT 1

What is a Mobile Operating System (Mobile OS)?

A Mobile OS also called a mobile OS, is an OS that is specifically designed to run on mobile devices such as mobile phones, smart phones, PDAs, tablet computers and other handheld devices.

Linux or Windows operating system controls your desktop or laptop computer, a mobile operating system is the software platform on top of which other programs can run on mobile devices. The operating system is responsible for determining the functions and features available on your device, such as thumb wheel, keyboards, WAP, synchronization with applications, email, text messaging and more. The mobile OS will also determine which third-party applications (mobile apps) can be used on your device.

Types of Mobile Operating Systems

When you purchase a mobile device the manufacturer will have chosen the operating system for that specific device. Often, you will want to learn about the mobile operating system before you purchase a device to ensure compatibility and support for the mobile applications you want to use.

9 Popular Mobile Operating Systems

1. Android OS (Google Inc.)

The Android mobile operating system is Google's open and free software stack that includes an operating system, middleware and also key applications for use on mobile devices, including smartphones. Updates for the open source Android mobile operating system have been developed under "dessert-inspired" version names (Cupcake, Donut, Eclair, Gingerbread, Honeycomb, Ice Cream Sandwich) with each new version arriving in alphabetical order with new enhancements and improvements.

2. Bada (Samsung Electronics)

Bada is a proprietary Samsung mobile OS that was first launched in 2010. The Samsung Wave was the first smartphone to use this mobile OS. Bada provides mobile features such as multipoint-touch, 3D graphics and of course, application downloads and installation.

3. BlackBerry OS (Research In Motion)

The BlackBerry OS is a proprietary mobile operating system developed by Research In Motion for use on the company’s popular BlackBerry handheld devices. The BlackBerry platform is popular with corporate users as it offers synchronization with Microsoft Exchange, Lotus

Domino, Novell GroupWise email and other business software, when used with the BlackBerry Enterprise Server.

4. iPhone OS / iOS (Apple)

Apple's iPhone OS was originally developed for use on its iPhone devices. Now, the mobile operating system is referred to as iOS and is supported on a number of Apple devices including the iPhone, iPad, iPad 2 and iPod Touch. The iOS mobile operating system is available only on Apple's own manufactured devices as the company does not license the OS for third-party hardware. Apple iOS is derived from Apple's Mac OS X operating system.

5. MeeGo OS (Nokia and Intel)

A joint open source mobile operating system which is the result of merging two products based on open source technologies: Maemo (Nokia) and Moblin (Intel). MeeGo is a mobile OS designed to work on a number of devices including smartphones, netbooks, tablets, in-vehicle information systems and various devices using Intel Atom and ARMv7 architectures.

6. Palm OS (Garnet OS)

The Palm OS is a proprietary mobile operating system (PDA operating system) that was originally released in 1996 on the Pilot 1000 handheld. Newer versions of the Palm OS have added support for expansion ports, new processors, external memory cards, improved security and support for ARM processors and smartphones. Palm OS 5 was extended to provide support for a broad range of screen resolutions, wireless connections and enhanced multimedia capabilities and is called Garnet OS.

7. Symbian OS (Nokia)

Symbian is a mobile operating system (OS) targeted at mobile phones that offers a high-level of integration with communication and personal information management (PIM) functionality. Symbian OS combines middleware with wireless communications through an integrated mailbox and the integration of Java and PIM functionality (agenda and contacts). Nokia has made the Symbian platform available under an alternative, open and direct model, to work with some OEMs and the small community of platform development collaborators. Nokia does not maintain Symbian as an open source development project.

8. webOS (Palm/HP)

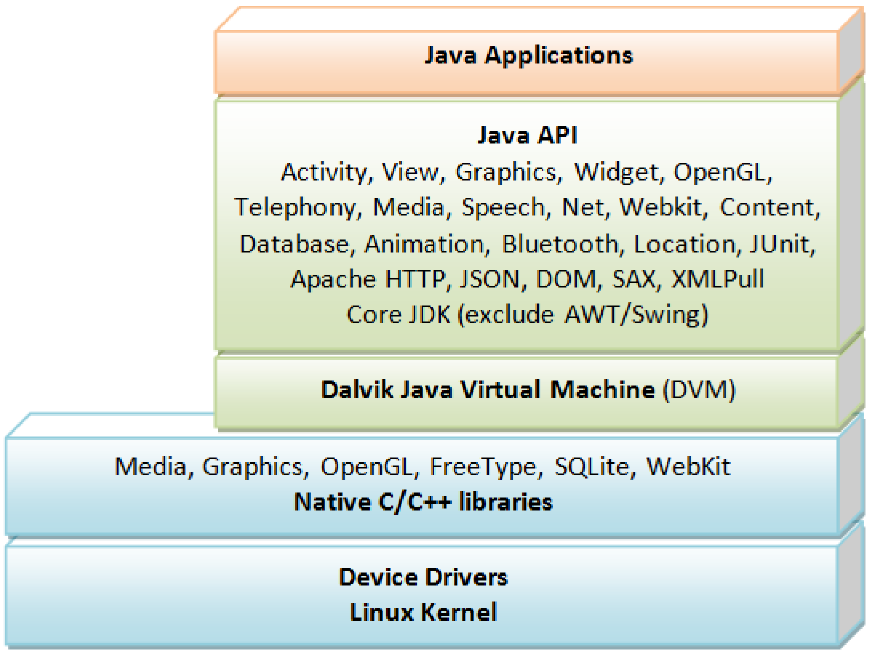
WebOS is a mobile operating system that runs on the Linux kernel. WebOS was initially developed by Palm as the successor to its Palm OS mobile operating system. It is a proprietary Mobile OS which was eventually acquired by HP and now referred to as webOS (lower-case w) in HP literature. HP uses webOS in a number of devices including several smartphones and HP TouchPads. HP has pushed its webOS into the enterprise mobile market by focusing on improving security features and management with the release of webOS 3.x. HP has also announced plans for a version of webOS to run within the Microsoft Windows operating system and to be installed on all HP desktop and notebook computers in 2012.

9. Windows Mobile (Windows Phone)

Windows Mobile is Microsoft's mobile operating system used in smartphones and mobile devices – with or without touchscreens. The Mobile OS is based on the Windows CE 5.2 kernel. In 2010 Microsoft announced a new smartphone platform called Windows Phone 7.

Android Platform

Android is based on Linux with a set of native core C/C++ libraries. Android applications are written in Java. However, they run on Android's own Java Virtual Machine, called Dalvik Virtual Machine (DVM) (instead of JDK's JVM) which is optimized to operate on the small and mobile devices.


Java Applications
Java API
Activity, View, Graphics, Widget, OpenGL, Telephony, Media, Speech, Net, Webkit, Content, Database, Animation, Bluetooth, Location, JUnit, Apache HTTP, JSON, DOM, SAX, XMLPull
Core JDK (exclude AWT/Swing)
Dalvik Java Virtual Machine (DVM)
Media, Graphics, OpenGL, FreeType, SQLite, WebKit
Native C/C++ libraries
Device Drivers
Linux Kernel

In May 2017, Google announced support for a new Kotlin programming language. As you are familiar with Java, you probably should start in Java (many of the examples out there are written in Java), and then move into Kotlin. Kotlin will not be discussed in this article.

ANDROID

What is android?

Android is a mobile operating system developed by Google. It is based on a modified version of the Linux Kernel and other open source software, and is designed primarily for touch screen mobile devices such as smart phones and tablets.

History of Android?

- Initially, Andy Rubin founded Android Incorporation in Palo Alto, California, United States in October, 2003.
- In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.
- The Key employees of Android Incorporation are Andy Rubin, Rich Miner, Chris White and Nick Sears.
- Originally intended for camera but shifted to smart phones later because of low market for camera only.
- Android is the nick name of Andy Rubin given by coworkers because of his love to robots.
- In 2007, Google announces the developed of android OS.
- In 2008, HTC launched the first android mobile.

Features of Android operating System

- Near Field Communication (NFC)
- Alternate Keyboards
- Infrared Transmission
- No-Touch Control
- Automation
- Wireless App Downloads
- Storage and Battery Swap
- Custom Home Screen
- Widgets
- Custom ROMs

ANDROID VERSIONS

ANDROID VERSIONS NAME	APL LEVEL	LINUX KERNEL IN (AOSP)	VERSION	RELEASE	FEATURES

No codename	1	1.0	-	-	-
Cupcake	3	2.6.27	1.5	2009	Support widgets, search browsers navigation apps,
Donut	4	2.6.29	1.6	2009	Screen capture, voice commands
Éclair	5	2.6.29	2.1	2010	Battery savor, enable in low resolution 320*240 Keyboard auto-correct
Froyo	8	2.6.32	2.2	2010	High security, cloud API, gesture detection.
Ginger-bread	9and10	2.6.35	2.3	2011	NFC range in 10 cm, audio video calls, 3g supported, gyroscope sensors
Honeycomb	11and13	2.6.36	3.1and3.3	2011	Symmetric multiprocessor, multitasking, Google talk, 3D effects
Ice cream sandwich	14	3.0.1	4.0	2011	Notification pop up on lock screens, fast image capture, NFG enable.
Jelly bean	16	3.4.39	4.1,4.2,4.3	2012	Speedy, audio & video calling, remote access
Kitkat	19	3.10	4.4	2013	Responsive, tri core cpu, cloud response
Lollipop	21	3.16.1	5.0	2015	3D views, 64 bit MIPS, Sensors, heart rate, swipe pinch. Screen capture.
Marsh-mallow	23	3.18.10	6.0	2015	Finger Print detection, gesture & voice sensors

					in camera, high resolution, backup in cloud, power saving mode
Nougat	25	4.4.1	7.0	2016	Finger print, gestures voice sensors, palm detection. Power saving modes, cloud
Oreo	26-27		8.0	2017	Auto enable wifi, smart text selection, split screen apps.
Pie	28		9	2018	Multi-camera support, display cutout support, animation.
Android 10	29		10.0	2019	Smart Reply in all messaging apps, live caption, new parental controls,

PREREQUISITES ANDROID DEVELOPMENTR

- Java
- Understanding of XML
- Android SDK
- Android Studio
- APLS
- Databases
- Material Design

SYSTEM REQUIREMENTS

Windows

- Microsoft® Windows® 7/8/10 (64-bit)
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Mac

- Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Linux

- GNOME or KDE desktop

  Tested on gLinux based on Debian.
- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.19 or later
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

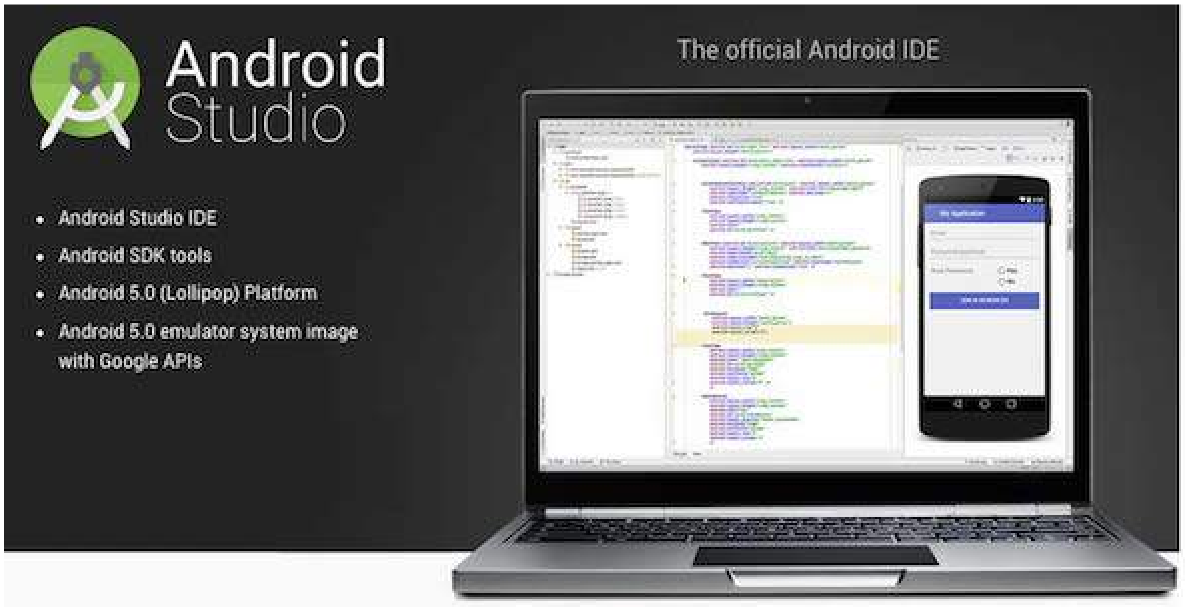
Chrome OS

- 8 GB RAM or more recommended
- 4 GB of available disk space minimum
- 1280 x 800 minimum screen resolution
- Intel i5 or higher (U series or higher) recommended

Recommended devices:

- **Acer:** Chromebook 13/Spin 13, Chromebox CXI3
- **Lenovo:** Yoga C630 Chromebook
- **HP:** Chromebook x360 14, Chromebox G2
- **Dell:** Inspiron Chromebook 14
- **ASUS:** Chromebox 3
- **ViewSonic:** NMP660 Chromebox
- **CTL:** Chromebox CBx1

ANDROID STUDIO INSTALLATION


Android Studio
The official Android IDE
Android Studio IDE
Android SDK tools
Android 5.0 (Lollipop) Platform
Android 5.0 emulator system image with Google APIs

Step 1 - System Requirements

You will be delighted, to know that you can start your Android application development on either of the following operating systems −

- Microsoft® Windows® 10/8/7/Vista/2003 (32 or 64-bit)
- Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks)
- GNOME or KDE desktop

Second point is that all the required tools to develop Android applications are open source and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.

- Java JDK5 or later version
- Java Runtime Environment (JRE) 6
- Android Studio

Step 2 - Setup Android Studio

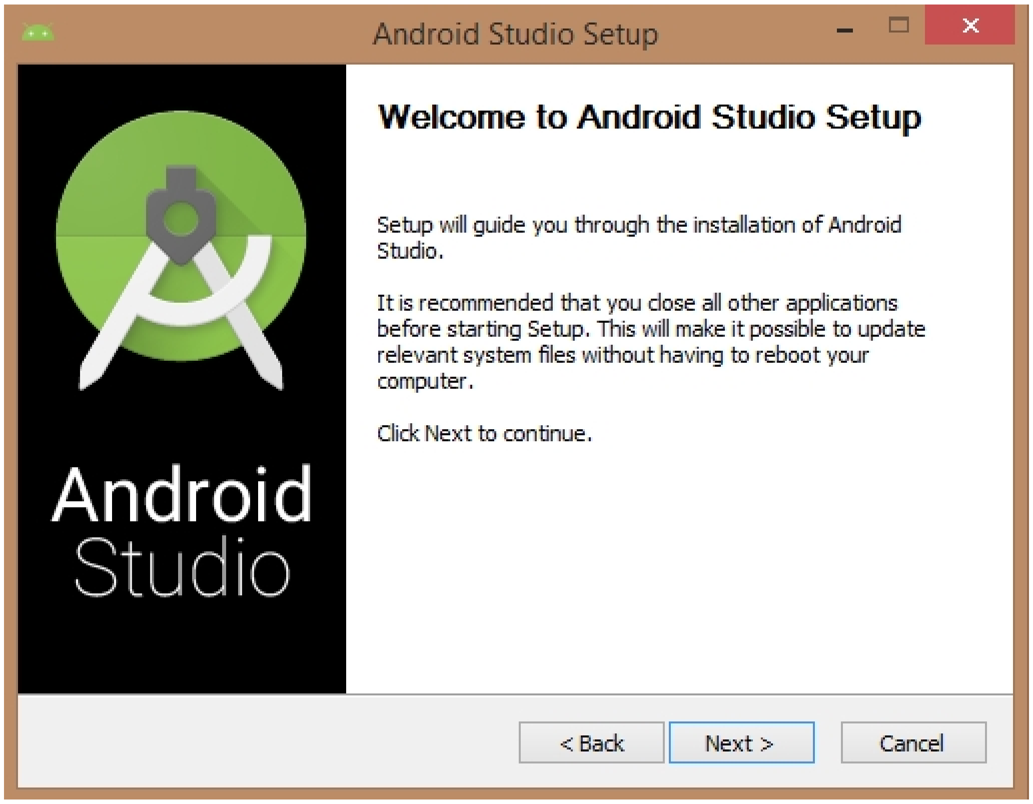
Overview

Android Studio is the official IDE for android application development.It works based on **IntelliJ IDEA**, You can download the latest version of android studio from Android Studio Download, If you are new to installing Android Studio on windows,you will find a file, which is named as *android-studio-bundle-143.3101438-windows.exe*.So just download and run on windows machine according to android studio wizard guideline.

If you are installing Android Studio on Mac or Linux, You can download the latest version from Android Studio Mac Download,or Android Studio Linux Download, check the instructions provided along with the downloaded file for Mac OS and Linux. This tutorial will consider that you are going to setup your environment on Windows machine having Windows operating system.

Installation

So let's launch *Android Studio.exe*,Make sure before launch Android Studio, Our Machine should required installed Java JDK. To install Java JDK,take a references of Android environment setup



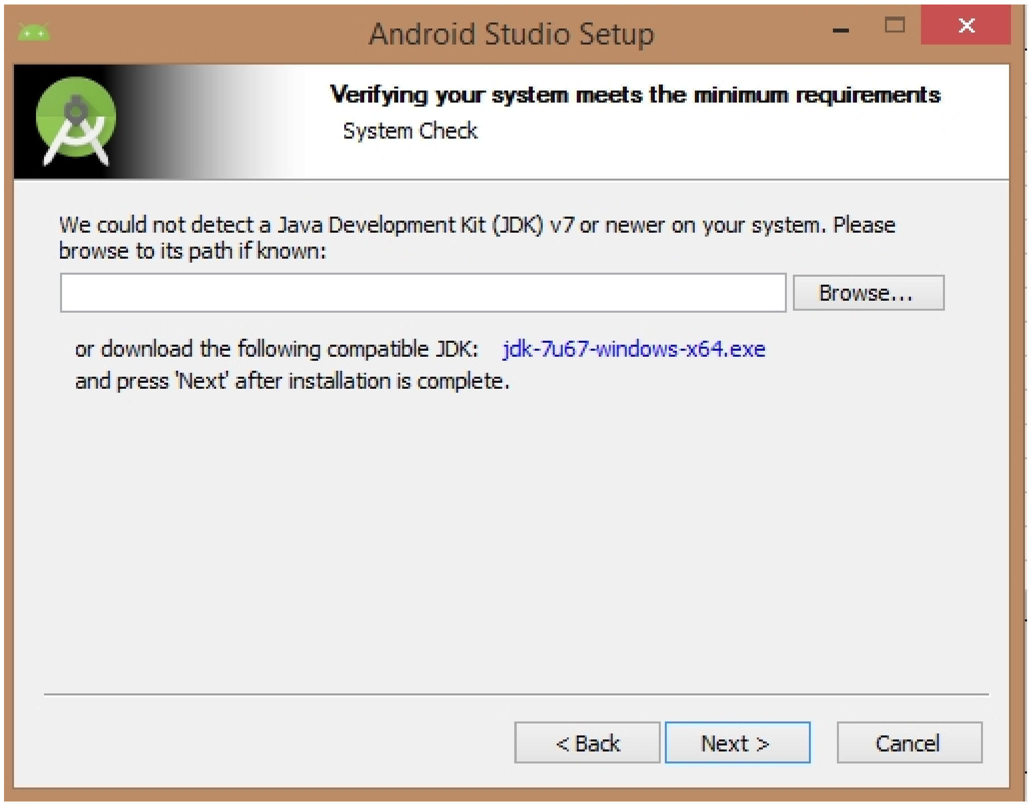
Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Once you launched Android Studio, its time to mention JDK7 path or later version in android studio installer.


Android Studio Setup

Verifying your system meets the minimum requirements

System Check

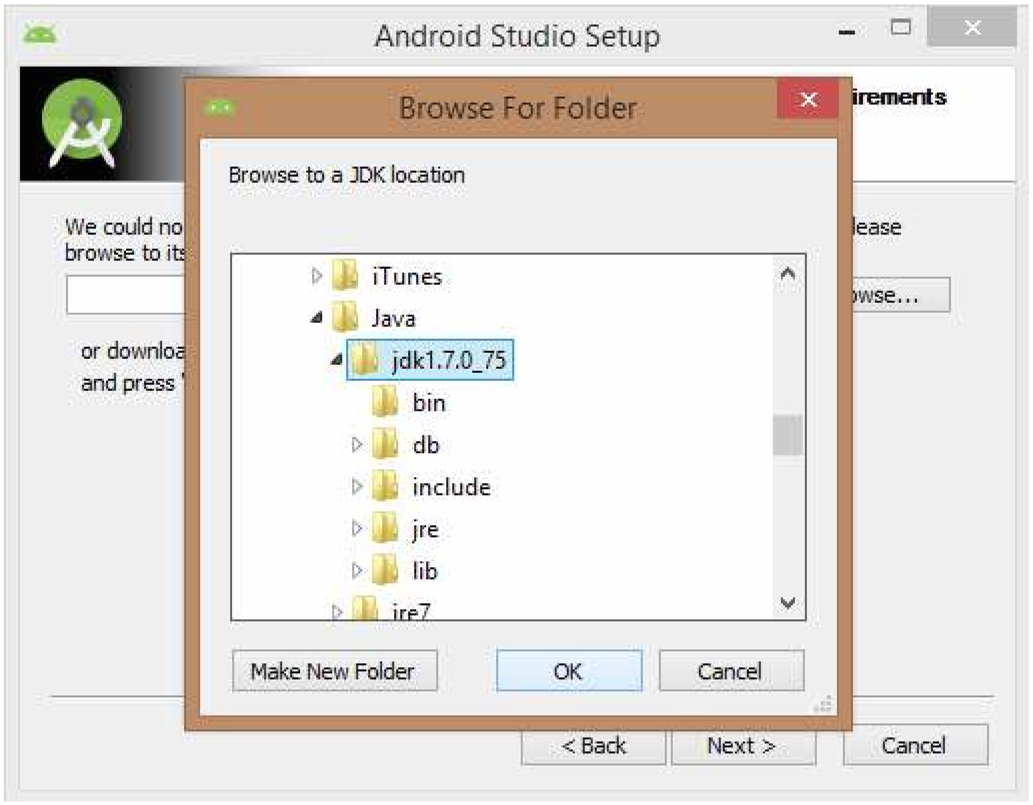
We could not detect a Java Development Kit (JDK) v7 or newer on your system. Please browse to its path if known:

Browse...

or download the following compatible JDK: jdk-7u67-windows-x64.exe
and press 'Next' after installation is complete.

< Back | Next > | Cancel

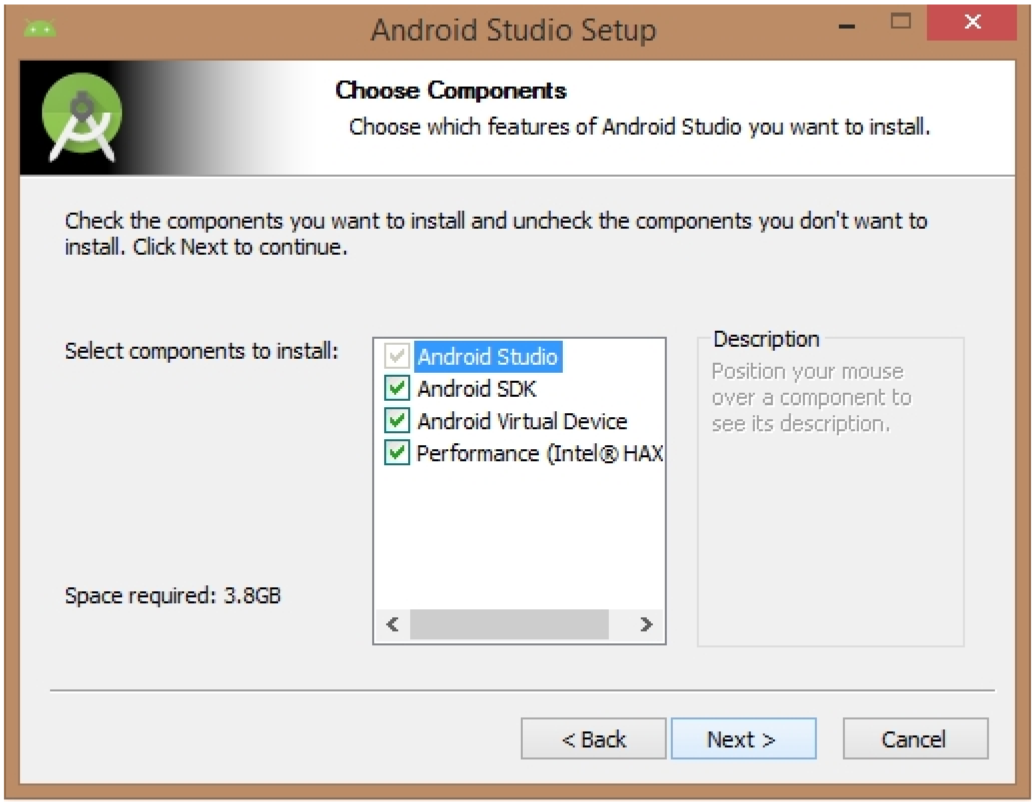
Below the image initiating JDK to android SDK


Android Studio Setup

Browse For Folder

Browse to a JDK location

Need to check the components, which are required to create applications, below the image has selected **Android Studio**, **Android SDK**, **Android Virtual Machine** and **performance(Intel chip).**


Android Studio Setup
Choose Components
Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

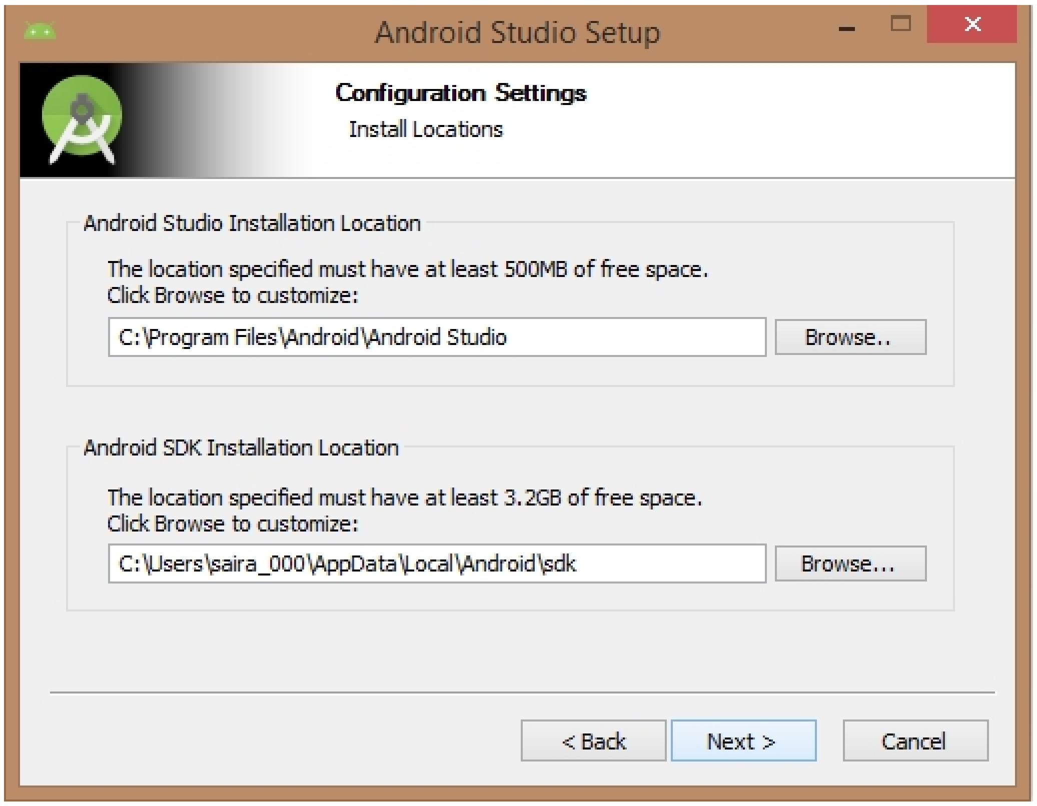
- [✓] Android Studio
- [✓] Android SDK
- [✓] Android Virtual Device
- [✓] Performance (Intel® HAX

Description

Position your mouse over a component to see its description.

Space required: 3.8GB

Need to specify the location of local machine path for Android studio and Android SDK, below the image has taken default location of windows 8.1 x64 bit architecture.



Configuration Settings

Install Locations

Android Studio Installation Location

The location specified must have at least 500MB of free space.
Click Browse to customize:

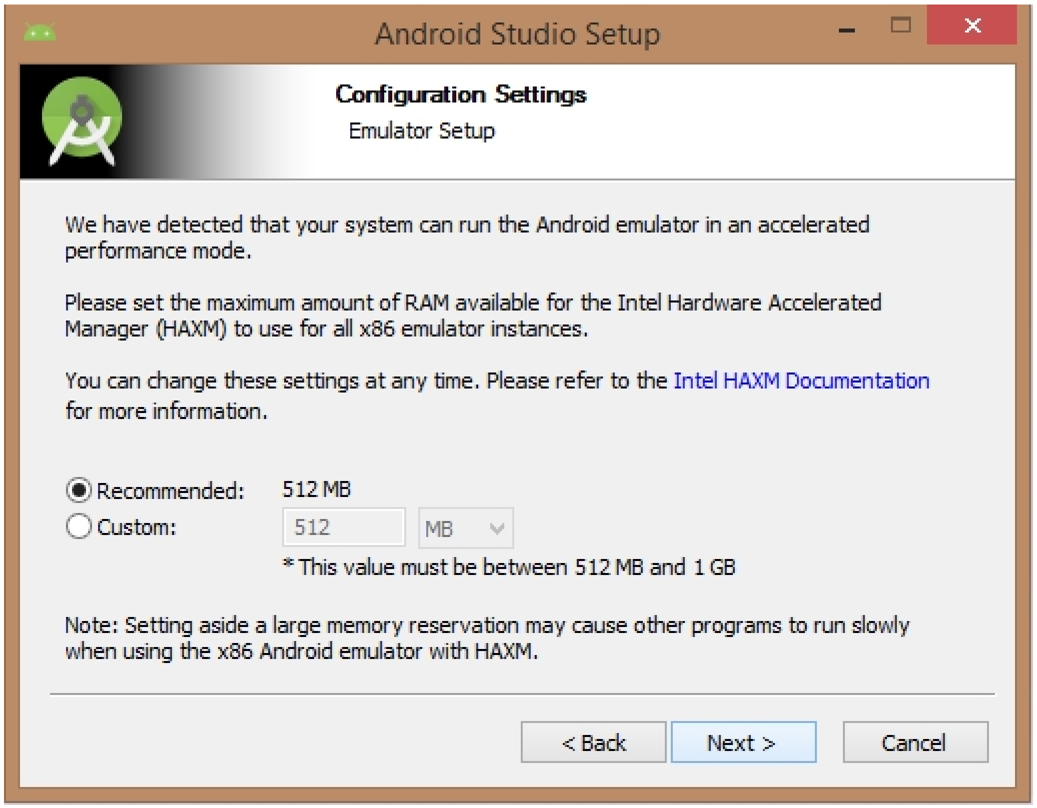
C:\Program Files\Android\Android Studio

Android SDK Installation Location

The location specified must have at least 3.2GB of free space.
Click Browse to customize:

C:\Users\saira_000\AppData\Local\Android\sdk

Need to specify the ram space for Android emulator by default it would take 512MB of local machine RAM.



Configuration Settings

Emulator Setup

We have detected that your system can run the Android emulator in an accelerated performance mode.

Please set the maximum amount of RAM available for the Intel Hardware Accelerated Manager (HAXM) to use for all x86 emulator instances.

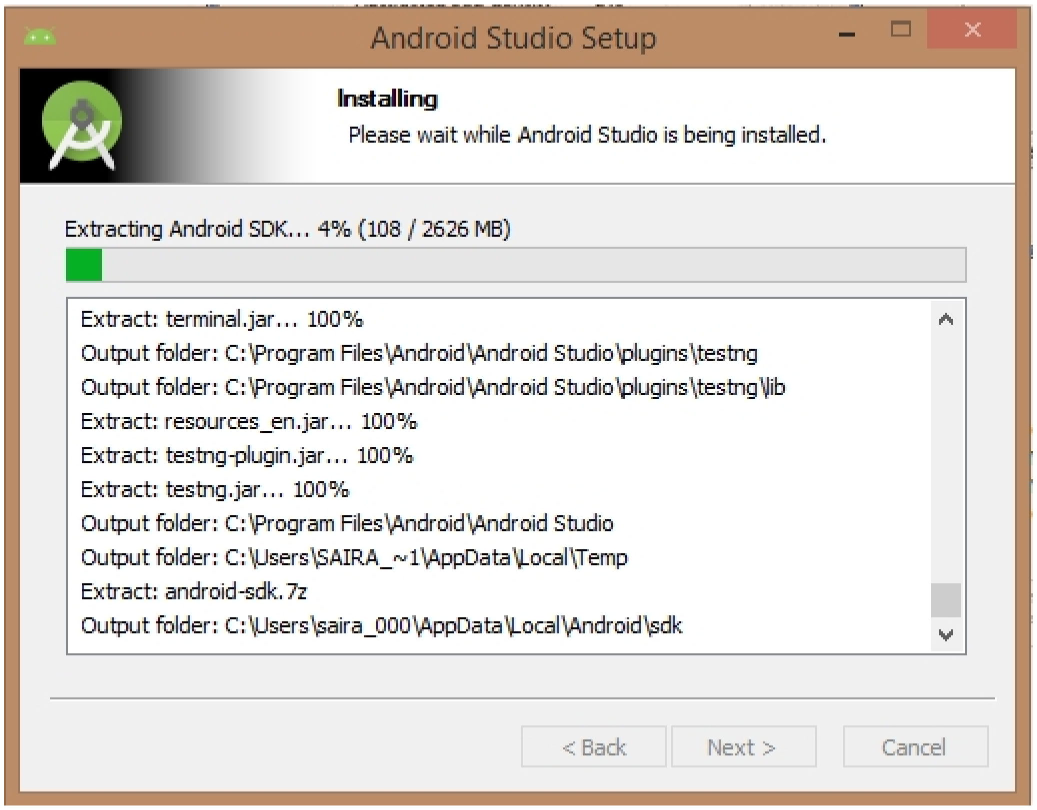
You can change these settings at any time. Please refer to the Intel HAXM Documentation for more information.

[✓ Recommended:] 512 MB
Custom: 512 MB

* This value must be between 512 MB and 1 GB

Note: Setting aside a large memory reservation may cause other programs to run slowly when using the x86 Android emulator with HAXM.

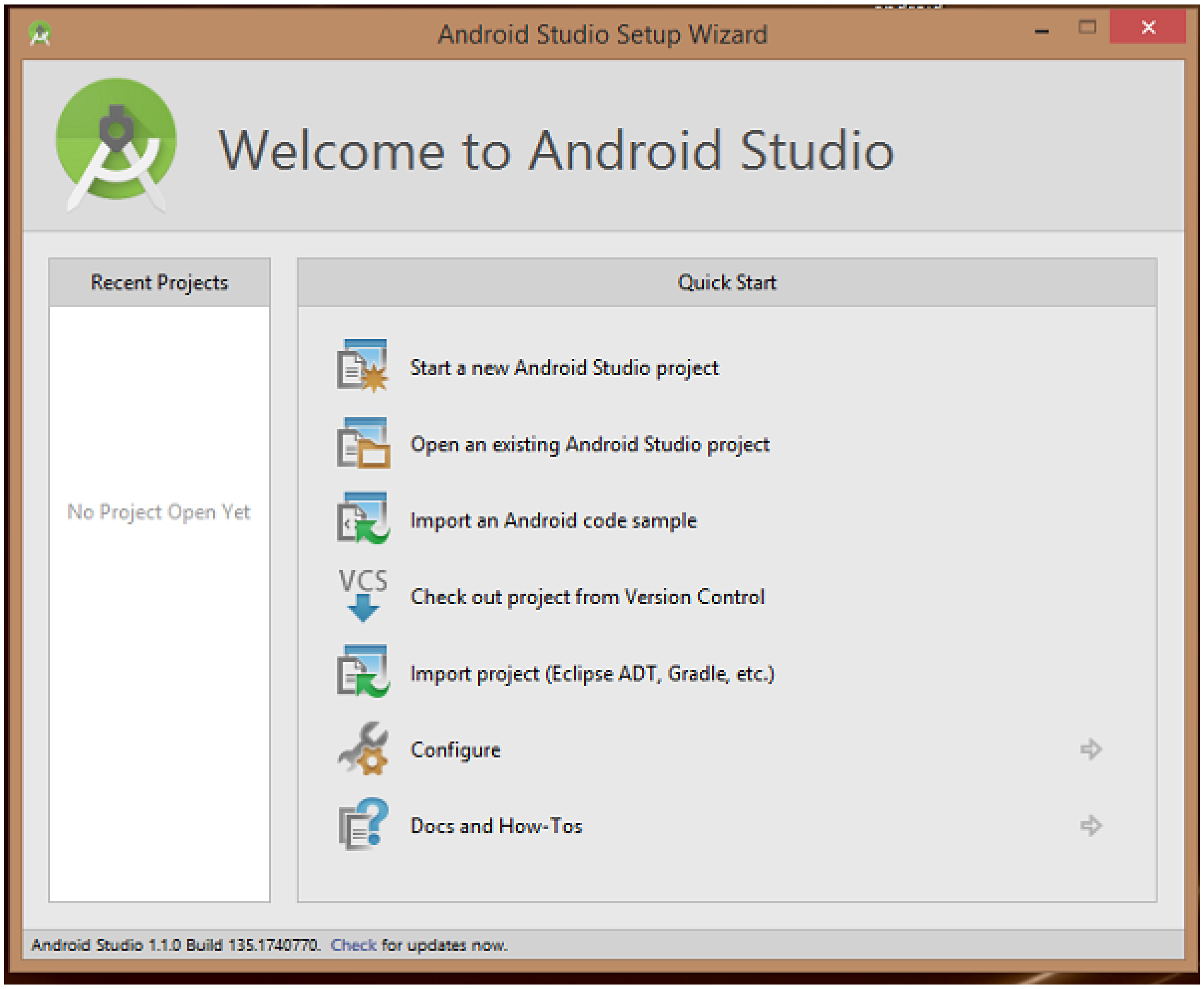
At final stage, it would extract SDK packages into our local machine, it would take a while time to finish the task and would take 2626MB of Hard disk space.



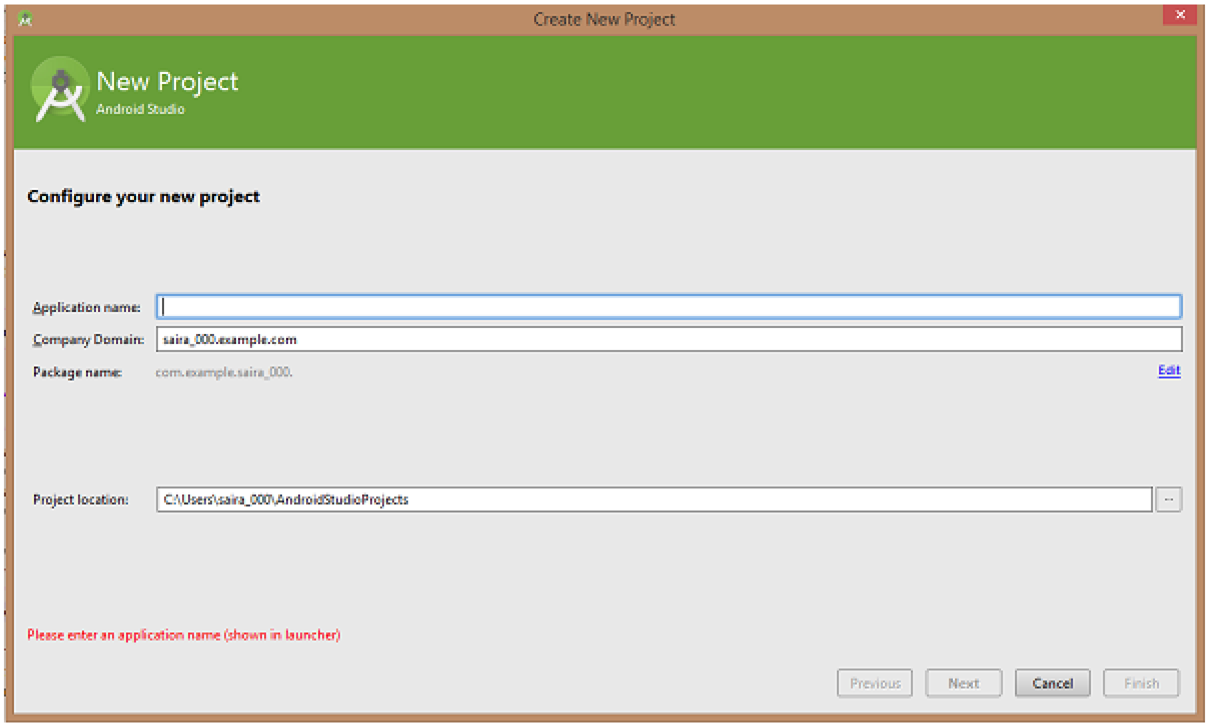
```
Extracting Android SDK... 4% (108 / 2626 MB)
```

```
Extract: terminal.jar... 100%
Output folder: C:\Program Files\Android\Android Studio\plugins\testng
Output folder: C:\Program Files\Android\Android Studio\plugins\testng\lib
Extract: resources_en.jar... 100%
Extract: testng-plugin.jar... 100%
Extract: testng.jar... 100%
Output folder: C:\Program Files\Android\Android Studio
Output folder: C:\Users\SAIRA_~1\AppData\Local\Temp
Extract: android-sdk.7z
Output folder: C:\Users\saira_000\AppData\Local\Android\sdk
```

After done all above steps perfectly, you must get finish button and it gonna be open android studio project with Welcome to android studio message as shown below


Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
VCS
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos
Android Studio 1.1.0 Build 135.1740770. Check for updates now.

You can start your application development by calling start a new android studio project. in a new installation frame should ask Application name, package information and location of the project.



Configure your new project

Application name:

Company Domain: saira_000.example.com

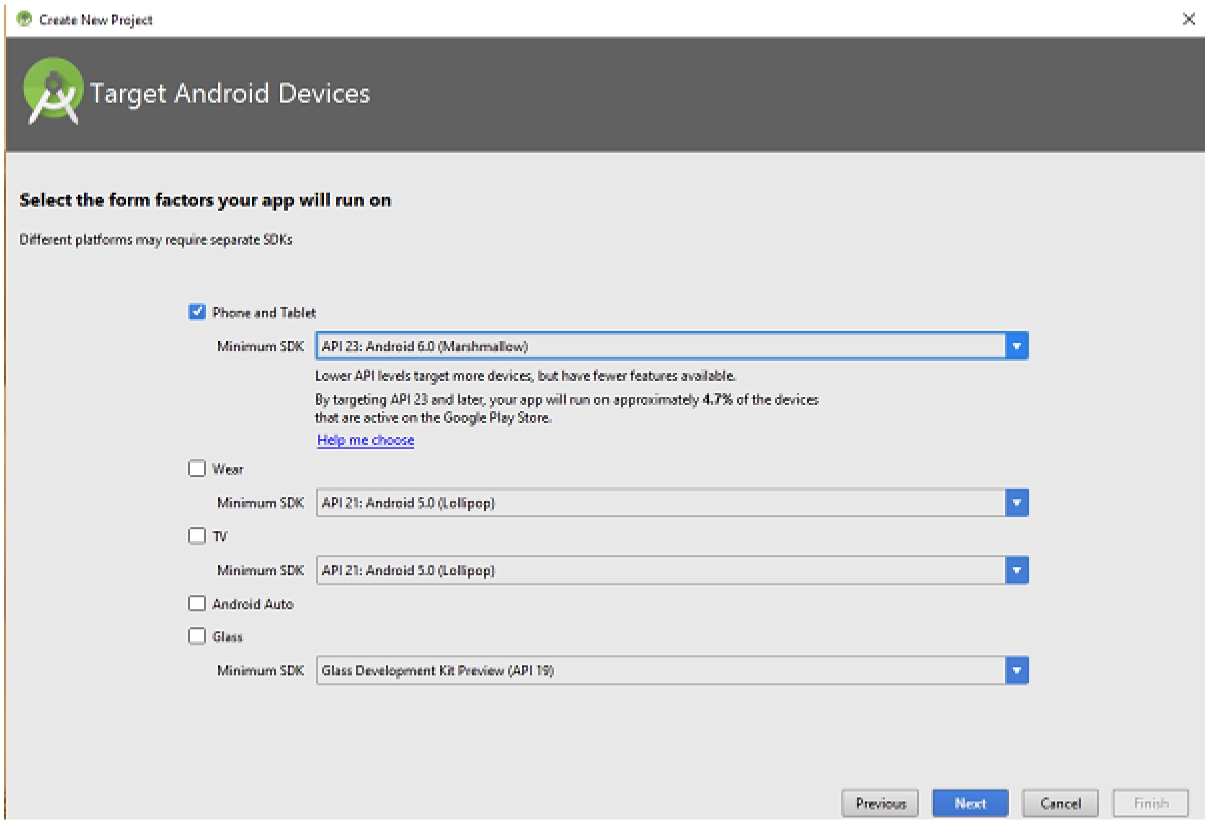
Package name: com.example.saira_000.

Edit

Project location: C:\Users\saira_000\AndroidStudioProjects

Please enter an application name (shown in launcher)

After entered application name, it going to be called select the form factors your application runs on, here need to specify Minimum SDK, in our tutorial, I have declared as API23: Android 6.0(Mashmallow)



Select the form factors your app will run on

Different platforms may require separate SDKs

[✓] Phone and Tablet

Minimum SDK: API 23: Android 6.0 (Marshmallow)

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

By targeting API 23 and later, your app will run on approximately 4.7% of the devices that are active on the Google Play Store.

Help me choose

[ ] Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] TV

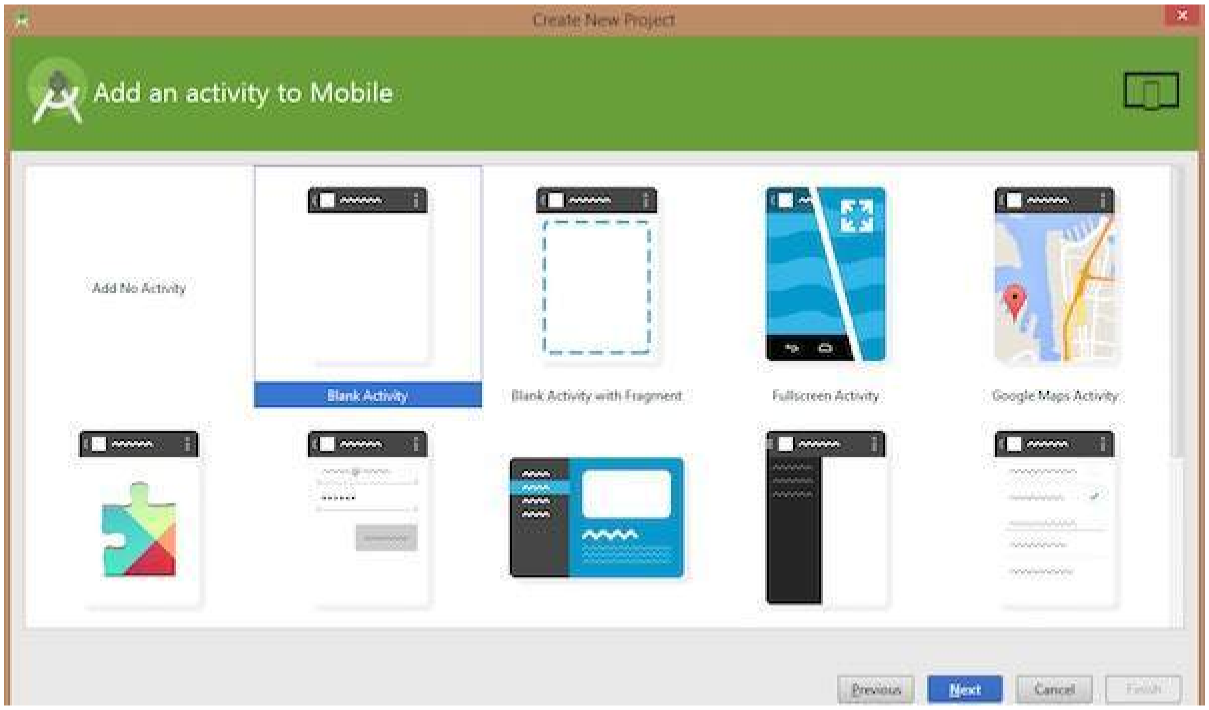
Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] Android Auto

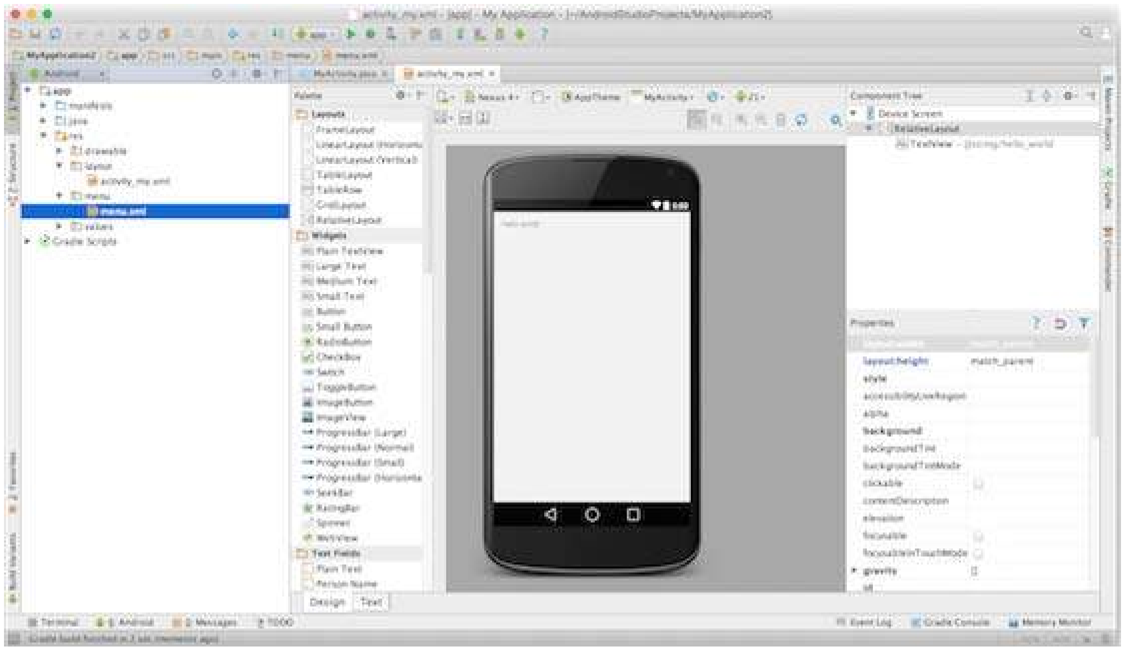
[ ] Glass

Minimum SDK: Glass Development Kit Preview (API 19)

The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications

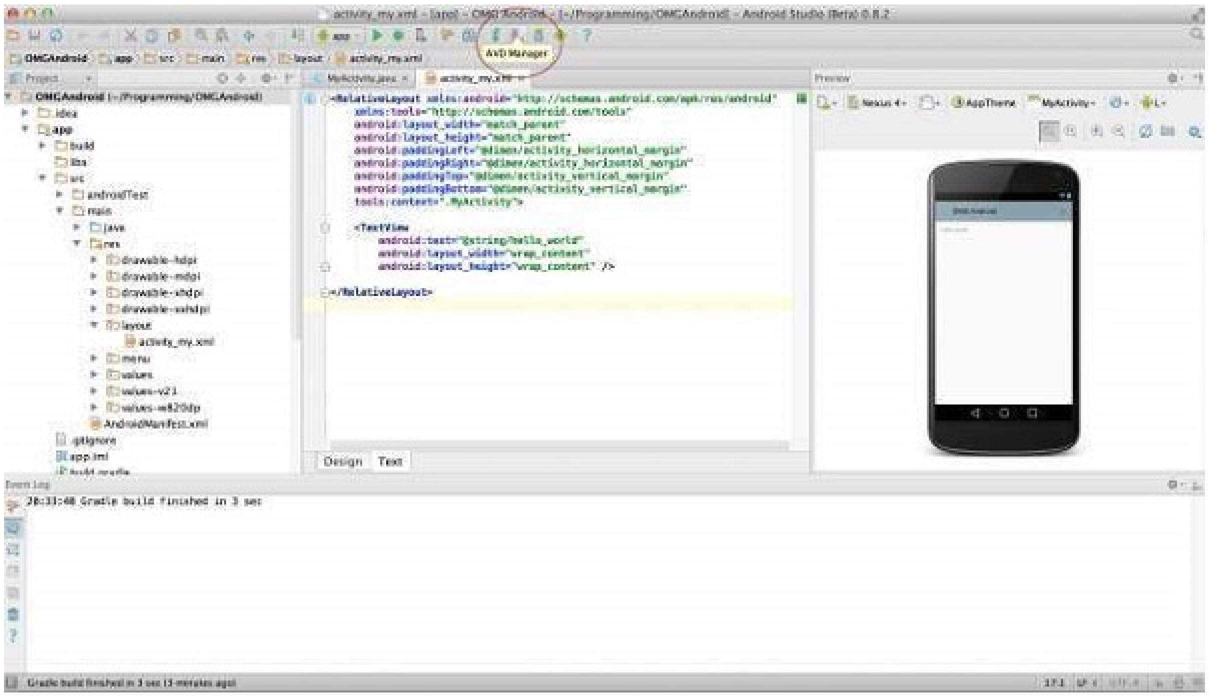


At the final stage it going to be open development tool to write the application code.



Step 3 - Create Android Virtual Device

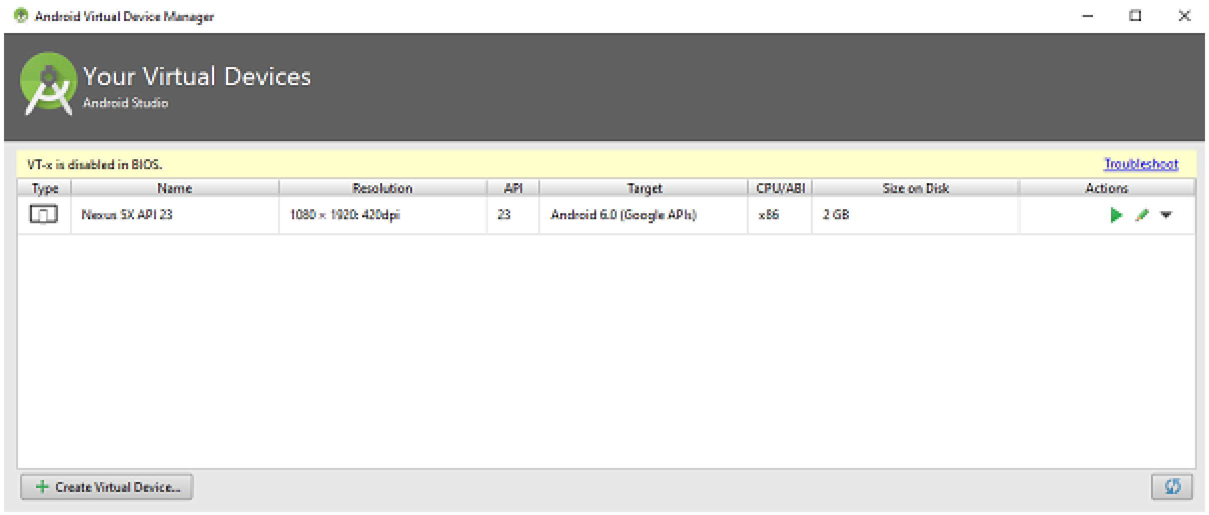
To test your Android applications, you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device. Launch Android AVD Manager Clicking AVD_Manager icon as shown below



```
<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=".MyActivity">

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

</RelativeLayout>
```

```
20:33:48 Gradle build finished in 3 sec
```

After Click on a virtual device icon, it going to be shown by default virtual devices which are present on your SDK, or else need to create a virtual device by clicking **Create new Virtual device** button



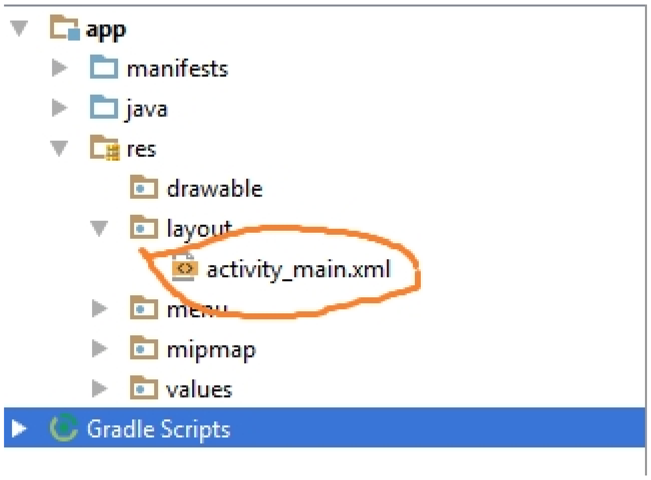
VT-x is disabled in BIOS.

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 23	1080 × 1920: 420dpi	23	Android 6.0 (Google APIs)	x86	2 GB	

If your AVD is created successfully it means your environment is ready for Android application development. If you like, you can close this window using top-right cross button. Better you re-start your machine and once you are done with this last step, you are ready to proceed for your first Android example but before that we will see few more important concepts related to Android Application Development.

Hello Word Example

Before Writing a Hello word code, you must know about XML tags.To write hello word code, you should redirect to **App>res>layout>Activity_main.xml**



To show hello word, we need to call text view with layout (about text view and layout, you must take references at Relative Layout and Text 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="@string/hello_world"
    android:layout_width="550dp"
    android:layout_height="wrap_content" />
</RelativeLayout>
```

Need to run the program by clicking **Run>Run App** or else need to call **shift+f10**key. Finally, result should be placed at Virtual devices as shown below

Integrated Development Environment (IDE)

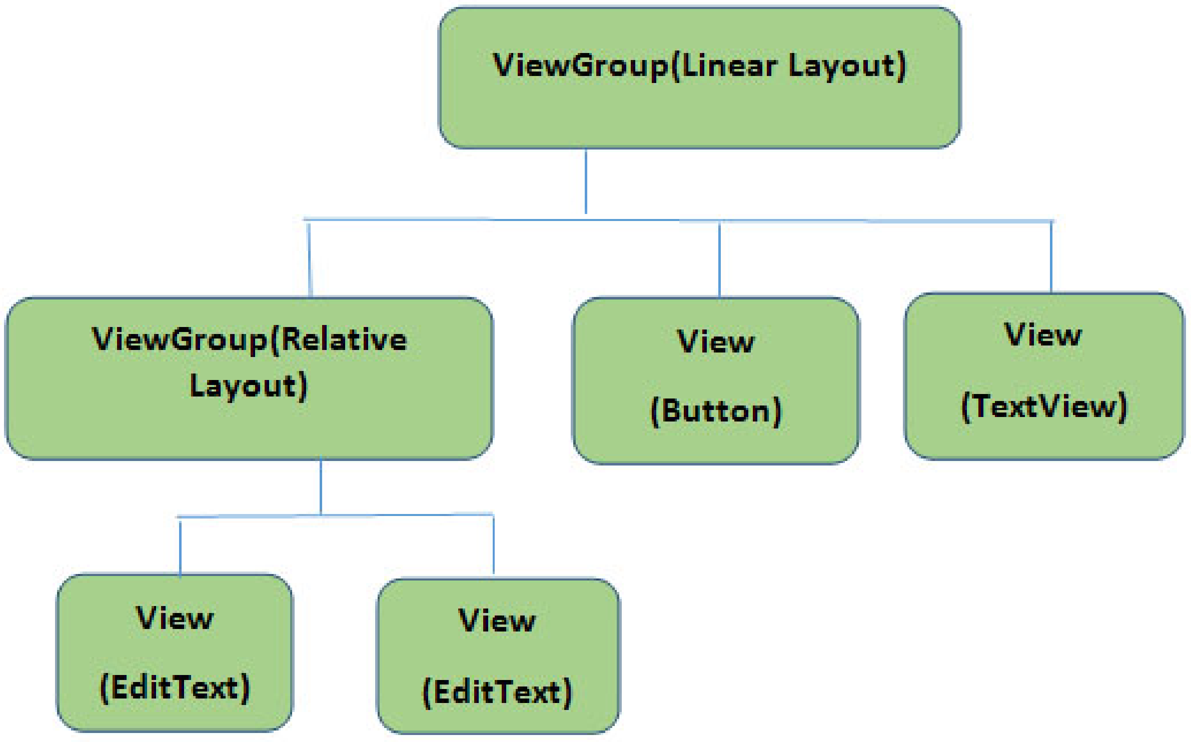
An integrated development environment (IDE) is a software suite that consolidates basic tools required to write and test software.

Developers use numerous tools throughout software code creation, building and testing. Development tools often include text editors, code libraries, compilers and test platforms. Without an IDE, a developer must select, deploy, integrate and manage all of these tools separately. An IDE brings many of those development-related tools together as a single framework, application or service.

XML in Android: Basics And Different XML Files Used In Android

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. XML tags are not predefined in XML. We must define our own Tags. Xml as itself is well readable both by human and machine. Also, it is scalable and simple to develop. In Android we use xml for designing our layouts because xml is lightweight language so it doesn’t make our layout heavy.

In this article we will go through the basic concepts of xml in Android and different XML files used for different purpose in Android. This will help you in writing a UI code to design your desired user interface

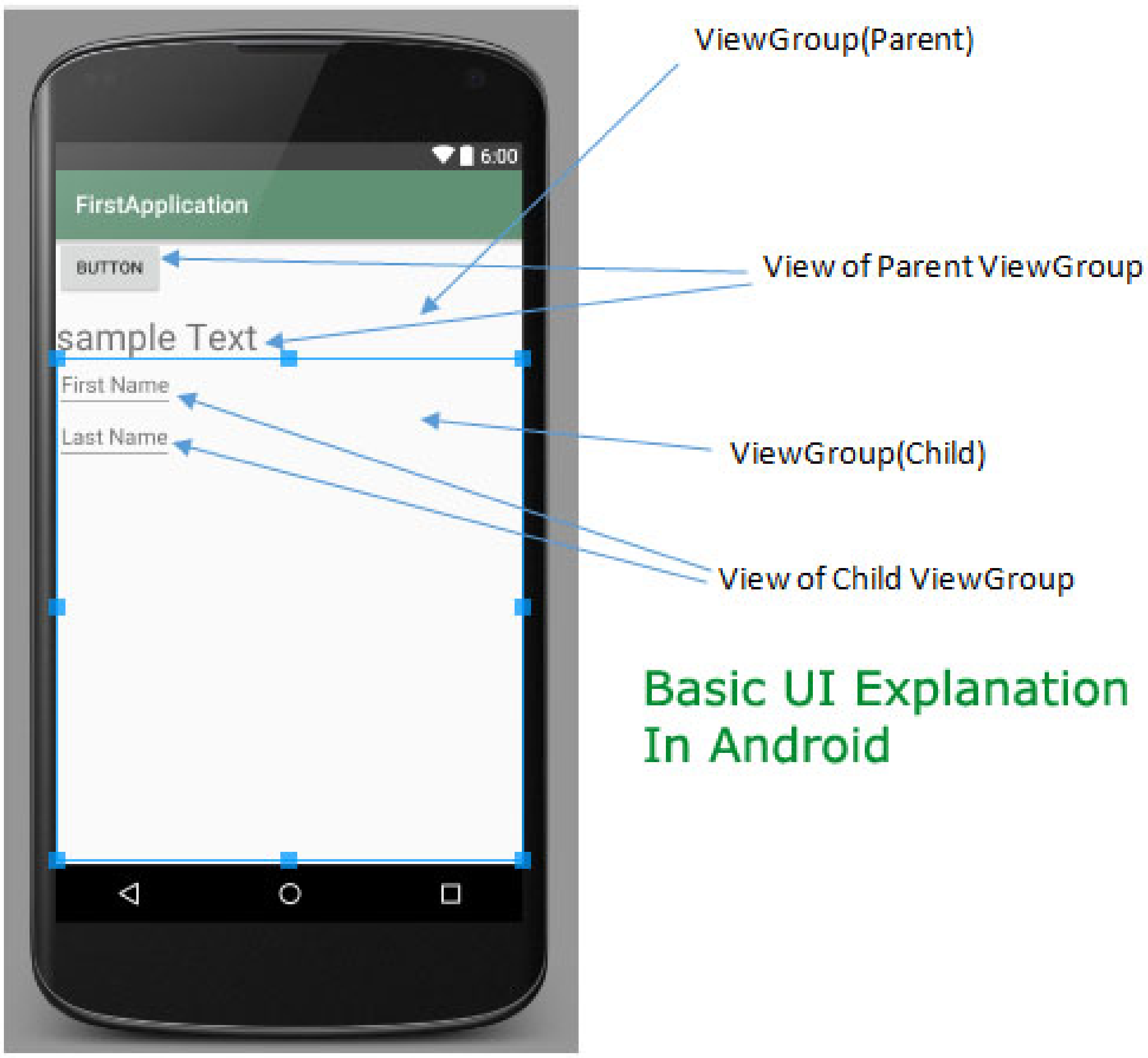


Here in above Diagram ViewGroup (Linear Layout) contains one ViewGroup (i.e. Relative Layout)and two View(Button and TextView). Further two more View (i.e. 2 EditText) are nested inside Relative Layout ViewGroup. It is important to note that one layout can be nested in another layout.

The below code snippet will explain the above image in better way. Paste it in **activity_main.xml**:

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

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



 <Button
 android:id="@+id/buton1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Button"/>

 <TextView
 android:id="@+id/textView1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="sample Text"
 android:layout_marginTop="15dp"
 android:textSize="30dp"/>

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

 <EditText
 android:id="@+id/editTextName"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:hint="First Name"
 />
 <EditText
 android:id="@+id/editTextLastName"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:hint="Last Name"/>


 </RelativeLayout>
</LinearLayout>
```



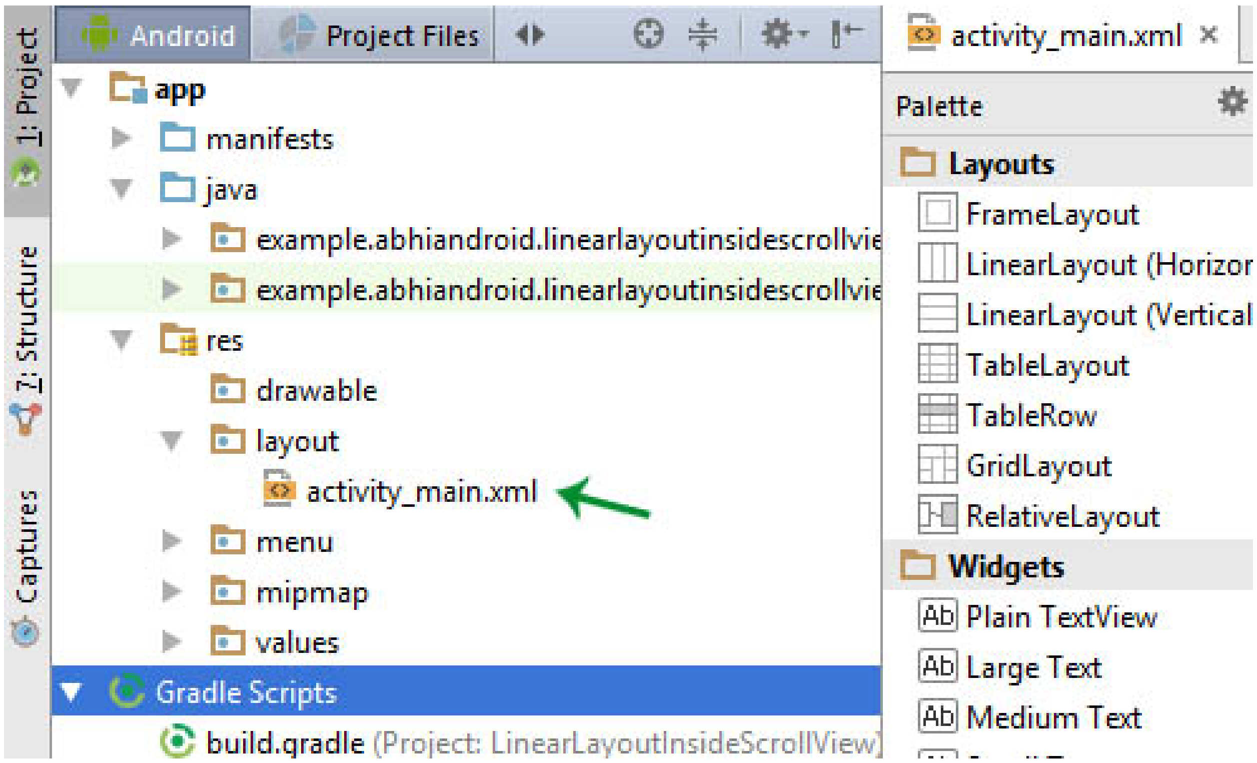
Different XML Files Used in Android:

In Android there are several xml files used for several different purposes. Below we define each and every one.

1. Layout XML Files: Layout xml files are used to define the actual UI(User interface) of our application. It holds all the elements(views) or the tools that we want to use in our application. Like the TextView's, Button's and other UI elements.

Location in Android Studio:

You will find out this file inside the **res** folder and inside it there is another folder named **layout** where you will get all the layout files for their respective activities or fragments.



Basic Layout XML Code:

Below we show **activity_main.xml** file in which we have two TextView's.

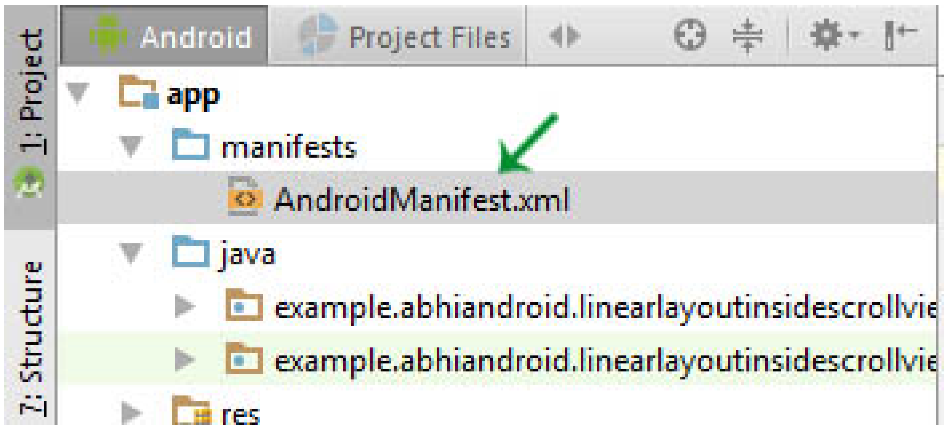
```
<!--  RelativeLayout in which we set green color for the background -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@color/greenColor"
tools:context=".MainActivity">


<TextView
android:id="@+id/firstTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_margin="20dp"
```

```
android:padding="10dp"
android:text="First Text View"
android:textColor="@color/white"
android:textSize="20sp"
android:textStyle="bold" />
<!-- second TextView -->
<TextView
android:id="@+id/secondTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@+id/firstTextView"
android:layout_centerHorizontal="true"
android:layout_margin="20dp"
android:padding="10dp"
android:text="Second Text View"
android:textColor="@color/white"
android:textSize="20sp"
android:textStyle="bold" />


</RelativeLayout>
```

2. Manifest xml File(Mainfest.xml): This xml is used to define all the components of our application. It includes the names of our application packages, our Activities, receivers, services and the permissions that our application needs. For Example – Suppose we need to use internet in our app then we need to define Internet permission in this file.

Location in Android Studio:

It is located inside app > manifests folder


1: Project
7: Structure
Android
Project Files
app
manifests
AndroidManifest.xml
java
example.abhiandroid.linearlayoutinsidescrollvie
example.abhiandroid.linearlayoutinsidescrollvie
res

Defining Internet Permission in AndroidManifest.xml

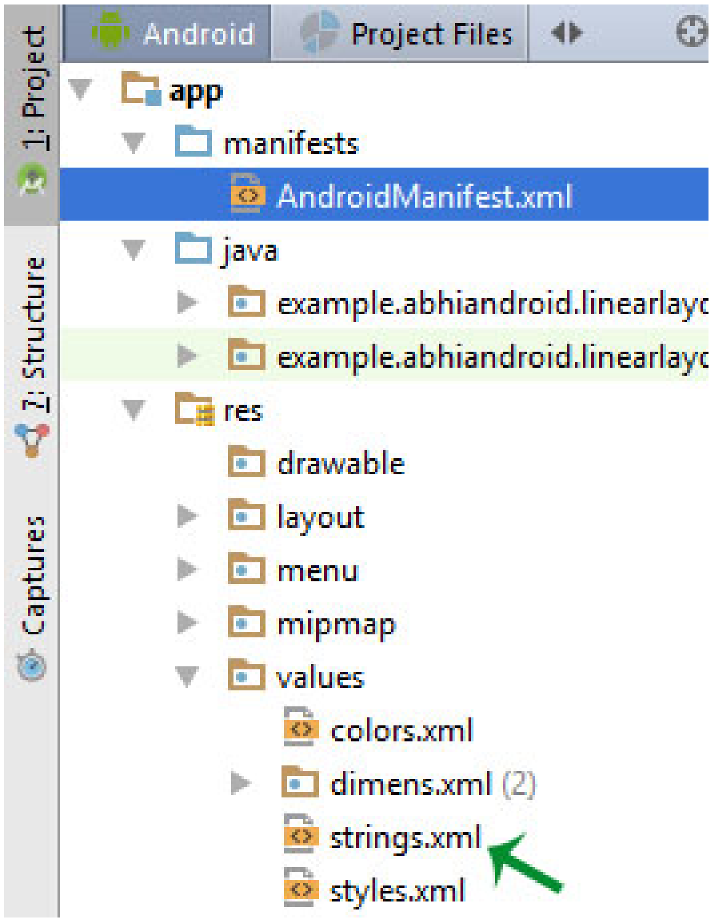
Below we show the **AndroidManifest.xml** file and define the Internet Permission in that file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="example.abhiandroid.MyApplication">    <!-- application package name -->

<uses-permission android:name="ANDROID.PERMISSION.INTERNET" />
<!-- define Internet Permission -->
<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">

<!-- add your Activities, Receivers, Services Names Here -->
<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>
```

3. Strings xml File(strings.xml): This xml file is used to replace the Hard-coded strings with a single string. We define all the strings in this xml file and then access them in our app(Activity or in Layout XML files) from this file. This file enhance the reusability of the code.

Location in Android Studio:



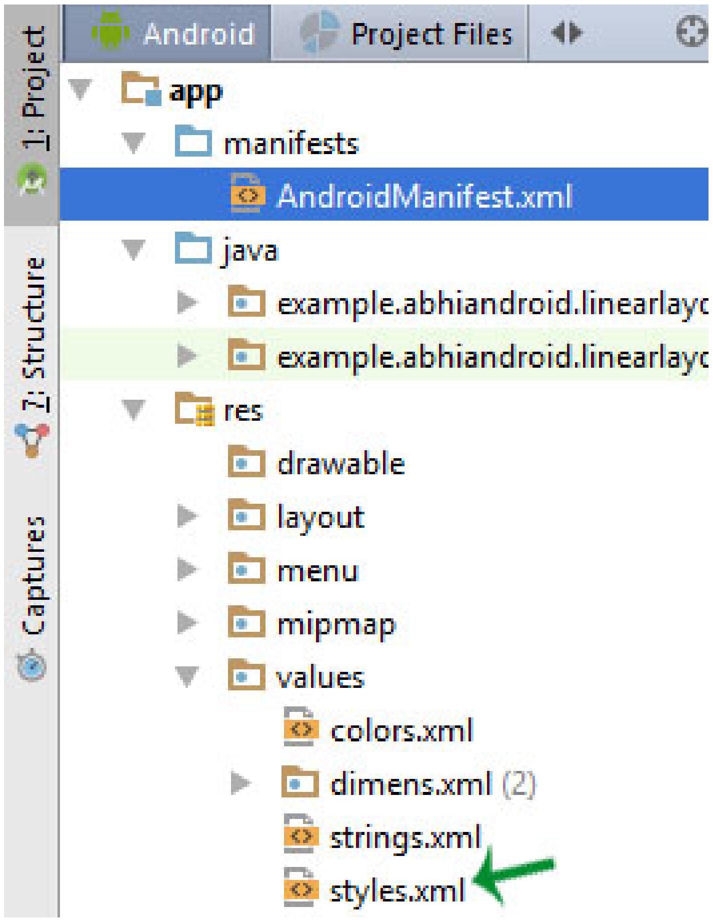
Below we show **strings.xml** file and define a string in the file.

```
<resources>
```

```
<string name="app_name">My Application</string>


<string name="hello_world">Hello world!</string>
<string name="action_settings">Settings</string>
<string name="login">User Login</string>
<!-- define your strings here -->
</resources>
```

4. Styles xml File(styles.xml): This xml is used to define different styles and looks for the UI(User Interface) of application. We define our custom themes and styles in this file.

Location in Android Studio:



Below we show the style.xml file.

```
<resources>
```

```
<!-- Base application theme. -->

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

</style>


</resources>
```

5. Drawable xml Files: These are those xml files that are used to provide various graphics to the elements or views of application. When we need to create a custom UI we use drawable xml files. Suppose if we need to define a gradient color in the background of Button or any custom shape for a view then we create a Drawable xml file and set it in the background of View.

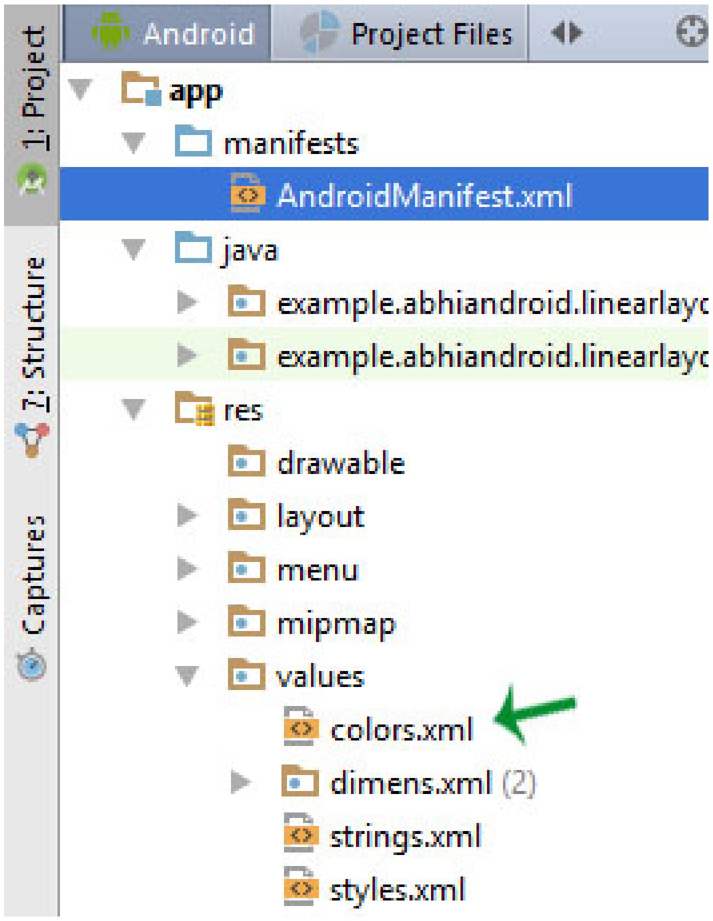
Do Read: How To Create Drawable Resource XML File in Android Studio

Below we show **custom_drawable.xml** file and create a gradient background color using style attribute.

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

<shape xmlns:android="http://schemas.android.com/apk/res/android">

<!-- define start, center and end color for gradient -->

<gradient

android:centerColor="#0f0"

android:endColor="#00f"

android:startColor="#f00" />

</shape>
```

6. Color xml File (colors.xml): This file is used to define the color codes that we used in our app. We simply define the color’s in this file and used them in our app from this file.

Location in Android Studio



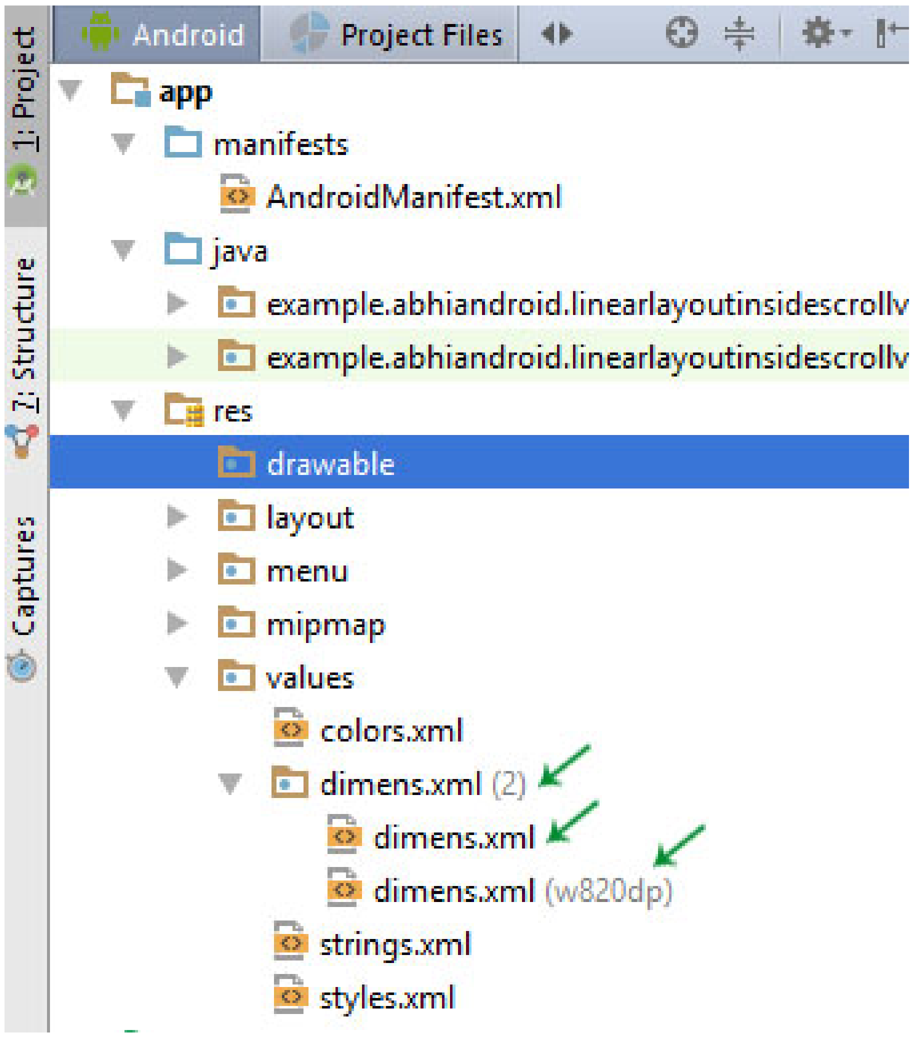
Below we show the **colors.xml** file in which we define green and white color.

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

<resources>

<!-- define your colors Here -->

<color name="greenColor">#0f0</color>

<color name="white">#fff</color>

</resources>
```

7. Dimension xml File(dimens.xml): This xml file is used to define the dimensions of the View's. Suppose we need a Button with 50dp(density pixel) height then we define the value 50dp in dimens.xml file and then use it in our app from this file.

stom themes and styles in this file.

Location in Android Studio:



Below we show the dimens.xml file in which we define 50dp dimension for Button height.

```
<resources>
<!-- Default screen margins, per the Android Design guidelines. -->
<dimen name="activity_horizontal_margin">16dp</dimen>
<dimen                    name="activity_vertical_margin">16dp</dimen><dimen
name="btnheight">50dp</dimen>
</resources>
```

.

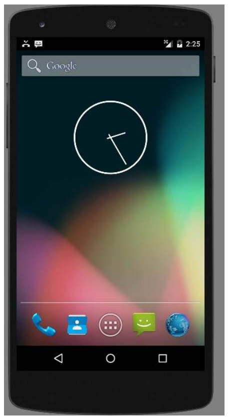
CREATING AVD

If you want to emulate a real device, first crate an AVD with the same device configurations as real device, then launch this AVD from AVD manager.

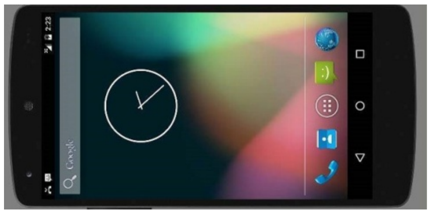
Changing Orientation

Usually by default when you launch the emulator, its orientation is vertical, but you can change it orientation by pressing Ctrl+F11 key from keyboard.

First launch the emulator. It is shown in the picture below −



Once it is launched, press **Ctrl+F11** key to change its orientation. It is shown below −



3.4.4 Emulator Commands.

Apart from just orientation commands, there are other very useful commands of emulator that you should keep in mind while using emulator. They are listed below −

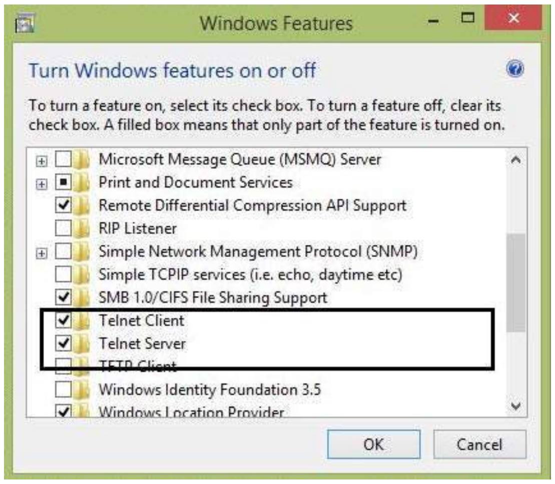
Sr.No	Command & description
1	**Home** Shifts to main screen
2	**F2** Toggles context sensitive menu
3	**F3** Bring out call log
4	**F4** End call
5	**F5**

	Search
6	**F6** Toggle trackball mode
7	**F7** Power button
8	**F8** Toggle data network
9	**Ctrl+F5** Ring Volume up
10	**Ctrl+F6** Ring Volume down

3.4.5 Emulator - Sending SMS

You can emulate sending SMS to your emulator. There are two ways to do that. You can do that from DDMS which can be found in Android studio, or from Telnet.(Network utility found in windows).

Sending SMS through Telnet.



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

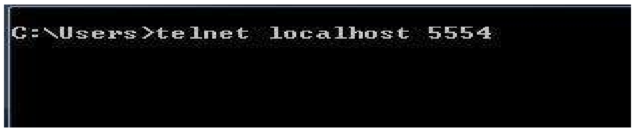
- ☐ Microsoft Message Queue (MSMQ) Server
- ■ Print and Document Services
- ✓ Remote Differential Compression API Support
- ☐ RIP Listener
- ☐ Simple Network Management Protocol (SNMP)
- ☐ Simple TCPIP services (i.e. echo, daytime etc)
- ✓ SMB 1.0/CIFS File Sharing Support
- ✓ Telnet Client
- ✓ Telnet Server
- ☐ TFTP Client
- ☐ Windows Identity Foundation 3.5
- ✓ Windows Location Provider

Telnet is not enabled by default in windows. You have to enable it to use it. Once enabled you can go to command prompt and start telnet by typing telnet.

In order to send SMS , note down the AVD number which can be found on the title bar of the emulator. It could be like this 5554 e.t.c. Once noted , type this command in command prompt.

```
telnet localhost 5554
```

Press enter when you type the command. It is shown below in the figure.

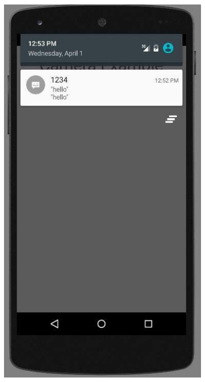


```
C:\Users>telnet localhost 5554
```

You will see that you are now connected to your emulator. Now type this command to send message.

```
sms send 1234 "hello"
```

Once you type this command , hit enter. Now look at the AVD. You will receive a notification displaying that you got a new text message. It is shown below −



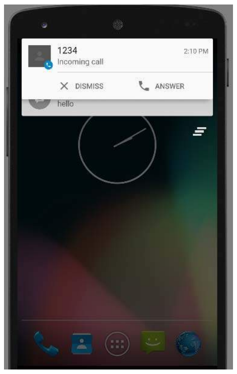
Emulator - Making Call

You can easily make phone calls to your emulator using telent client. You need to connect to your emulator from telnet. It is discussed in the sending sms topic above.

After that you will type this command in the telent window to make a call. Its syntax is given below −

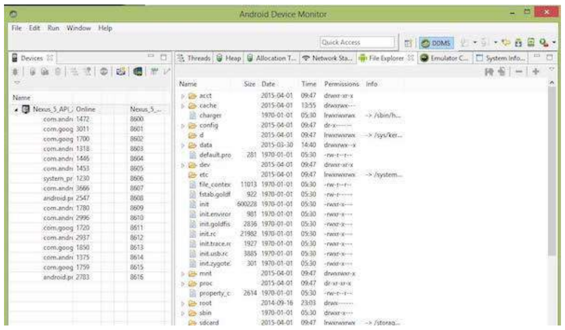
```
gsm call 1234
```

Once you type this command , hit enter. Now look at the AVD. You will receive a call from the number your put in the command. It is shown below −



Emulator - Transferring files

You can easily transfer files into the emulator and vice versa. In order to do that, you need to select the DDMS utility in Android studio. After that select the file explorer tab. It is shown below −



Name		
Nexus_5_API_	Online	Nexus_5_...
com.andr	1472	8600
com.goog	3011	8601
com.goog	1700	8602
com.andr	1318	8603
com.andr	1446	8604
com.andr	1453	8605
system_pr	1230	8606
com.andr	3666	8607
android.pr	2547	8608
com.andr	1780	8609
com.andr	2996	8610
com.goog	1720	8611
com.andr	2937	8612
com.goog	1850	8613
com.andr	1375	8614
com.goog	1759	8615
android.pr	2783	8616

Name	Size	Date	Time	Permissions	Info
acct		2015-04-01	09:47	drwxr-xr-x	
cache		2015-04-01	13:55	drwxrwx---	
charger		1970-01-01	05:30	lrwxrwxrwx	-> /sbin/h...
config		2015-04-01	09:47	dr-x------	
d		2015-04-01	09:47	lrwxrwxrwx	-> /sys/ker...
data		2015-03-30	14:40	drwxrwx--x	
default.pro	281	1970-01-01	05:30	-rw-r--r--	
dev		2015-04-01	09:47	drwxr-xr-x	
etc		2015-04-01	09:47	lrwxrwxrwx	-> /system...
file_contex	11013	1970-01-01	05:30	-rw-r--r--	
fstab.goldf	922	1970-01-01	05:30	-rw-r-----	
init	600228	1970-01-01	05:30	-rwxr-x---	
init.enviror	981	1970-01-01	05:30	-rwxr-x---	
init.goldfis	2836	1970-01-01	05:30	-rwxr-x---	
init.rc	21982	1970-01-01	05:30	-rwxr-x---	
init.trace.rc	1927	1970-01-01	05:30	-rwxr-x---	
init.usb.rc	3885	1970-01-01	05:30	-rwxr-x---	
init.zygote	301	1970-01-01	05:30	-rwxr-x---	
mnt		2015-04-01	09:47	drwxrwxr-x	
proc		2015-04-01	09:47	dr-xr-xr-x	
property_c	2614	1970-01-01	05:30	-rw-r--r--	
root		2014-09-16	23:03	drwx------	
sbin		1970-01-01	05:30	drwxr-x---	
sdcard		2015-04-01	09:47	lrwxrwxrwx	-> /storag...

Browse through the explorer and make new folder, view existing contents.

Android Architecture

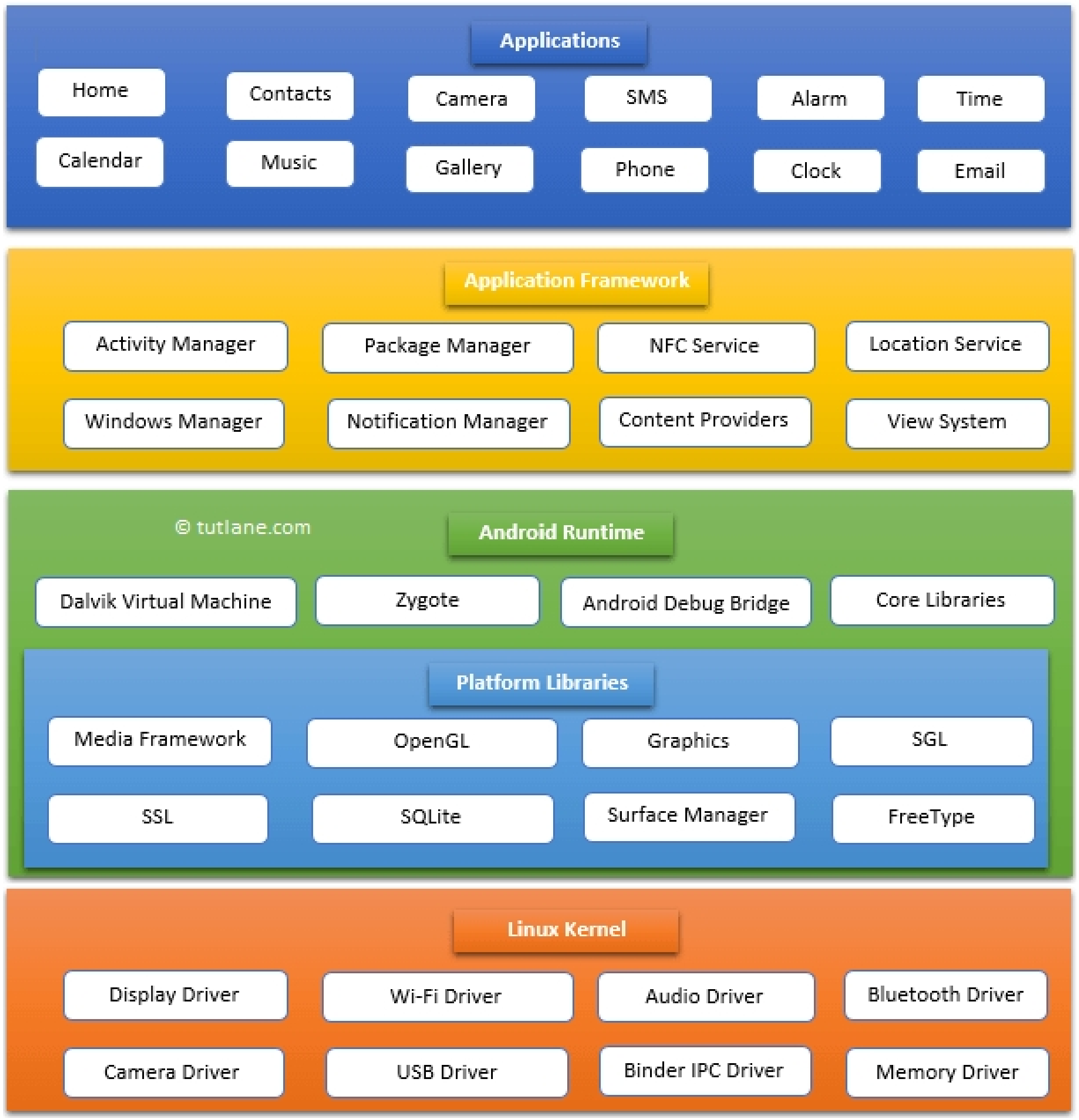
Android architecture is a software stack of components to support mobile device needs. Android software stack contains a Linux Kernel, collection of c/c++ libraries which are exposed through an application framework services, runtime, and application.

Following are main components of android architecture those are

1. Applications
2. Android Framework
3. Android Runtime
4. Platform Libraries
5. Linux Kernel

In these components, the **Linux Kernel** is the main component in android to provide its operating system functions to mobile and **Dalvik Virtual Machine** (**DVM**) which is responsible for running a mobile application.

Following is the pictorial representation of android architecture with different components



Applications

The top layer of the android architecture is **Applications**. The native and third-party applications like contacts, email, music, gallery, clock, games, etc. whatever we will build those will be installed on this layer only.
The application layer runs within the Android run time using the classes and services made available from the application framework.

Application Framework

The **Application Framework** provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources. It basically provides the services through which we can create a particular class and make that class helpful for the Application creation.

The application framework includes services like telephony service, location services, notification manager, NFC service, view system, etc. which we can use for application development as per our requirements.

Android Runtime

Android Runtime environment is an important part of Android rather than an internal part and it contains components like **core libraries** and the **Dalvik virtual machine**. The Android run time is the engine that powers our applications along with the libraries and it forms the basis for the application framework.

Dalvik Virtual Machine (**DVM**) is a register-based virtual machine like Java Virtual Machine (JVM). It is specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory management.

The **core libraries** in android runtime will enable us to implement android applications using standard JAVA programming language.

Platform Libraries

The **Platform Libraries** includes various C/C++ core libraries and Java-based libraries such as SSL, libc, Graphics, SQLite, Webkit, Media, Surface Manger, OpenGL, etc. to provide support for Android development.

The following are the summary details of some core android libraries available for android development.

- Media library for playing and recording audio and video formats
- The Surface manager library to provide a display management
- SGL and OpenGL Graphics libraries for 2D and 3D graphics
- SQLite is for database support and FreeType for font support
- Web-Kit for web browser support and SSL for Internet security.

Linux Kernel

Linux Kernel is a bottom layer and heart of the android architecture. It manages all the drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are mainly required for the android device during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the remainder of the stack. It is responsible for memory management, power management, device management, resource access, etc.