

Introduction:

Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a partnership of developers known as the Open Handset Alliance and commercially sponsored by Google. It was disclosed in November 2007, with the first commercial Android device, the HTC Dream, launched in September 2008.

It is free and open-source software. Its source code is Android Open Source Project (AOSP), primarily licensed under the Apache License. However, most Android devices dispatch with additional proprietary software pre-installed, mainly Google Mobile Services (GMS), including core apps such as Google Chrome, the digital distribution platform Google Play and the associated Google Play Services development platform.

- About 70% of Android Smartphone runs Google's ecosystem, some with vendor-customized user interface and some with software suite, such as *TouchWiz* and later *One UI* by Samsung, and *HTC Sense*.
- Competing Android ecosystems and forks include Fire OS (developed by Amazon) or LineageOS. However, the "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use android branding.

Definition: Android is a complete set of software for mobile devices such as tablet computers, notebooks, smart phones, electronic book readers, set-top boxes etc.

History of Android

The history and versions of android are interesting to know. The code names of android ranges from A to J, as **Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat** and **Lollipop**. Let's understand the android history in a sequence.

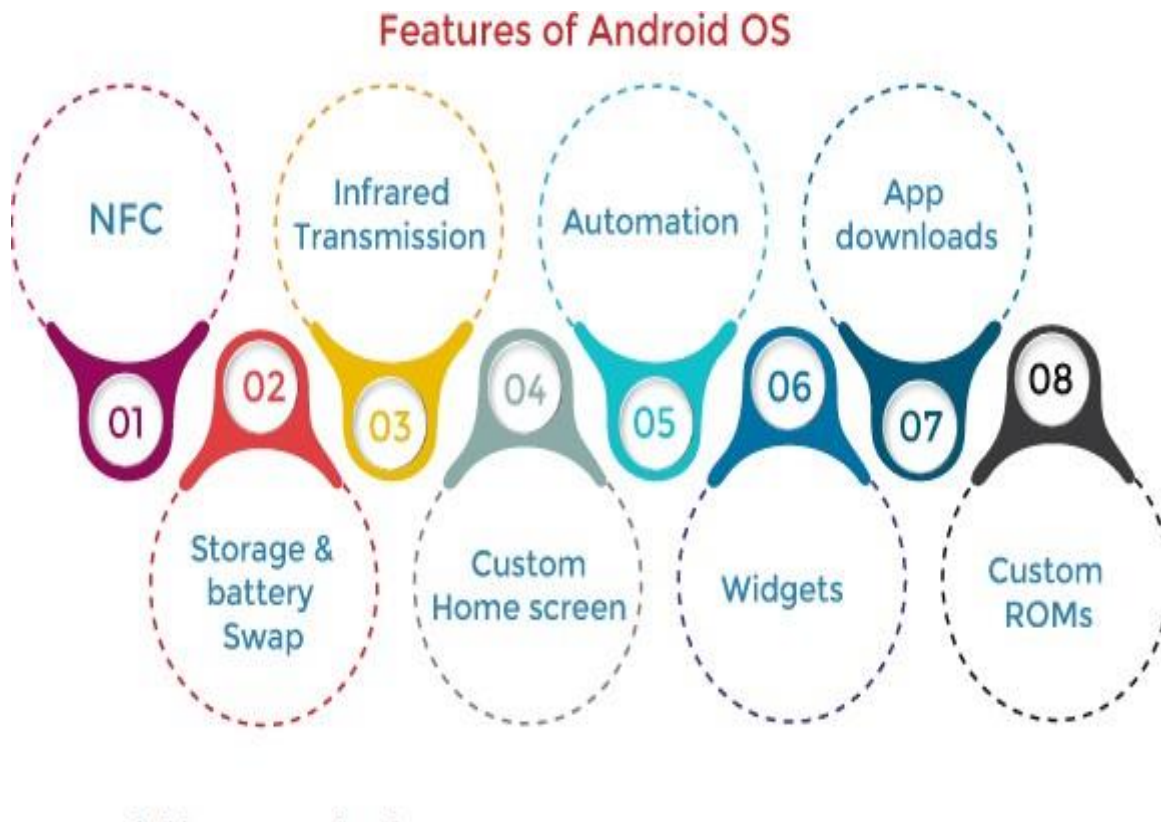
- 1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.
- 2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.
- 3) The key employees of Android Incorporation are **Andy Rubin, Rich Miner, Chris White** and **Nick Sears**.
- 4) Originally intended for camera but shifted to smart phones later because of low market for camera only.
- 5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS.

7) In 2008, HTC launched the first android mobile.

Features of Android Operating System

Below are the following unique features and characteristics of the android operating system, such as:



1. Near Field Communication (NFC)

Most Android devices support NFC, which allows electronic devices to interact across short distances easily. The main goal here is to create a payment option that is simpler than carrying cash or credit cards, and while the market hasn't exploded as many experts had predicted, there may be an alternative in the works, in the form of Bluetooth Low Energy (BLE).

2. Infrared Transmission

The Android operating system supports a built-in infrared transmitter that allows you to use your phone or tablet as a remote control.

3. Automation

The *Tasker* app allows control of app permissions and also automates them.

4. Wireless App Downloads

You can download apps on your PC by using the Android Market or third-party options like *AppBrain*. Then it automatically syncs them to your Droid, and no plugging is required.

5. Storage and Battery Swap

Android phones also have unique hardware capabilities. Google's OS makes it possible to upgrade, replace, and remove your battery that no longer holds a charge. In addition, Android phones come with SD card slots for expandable storage.

6. Custom Home Screens

While it's possible to hack certain phones to customize the home screen, Android comes with this capability from the get-go. Download a third-party launcher like *Apex*, *Nova*, and you can add gestures, new shortcuts, or even performance enhancements for older-model devices.

7. Widgets

Apps are versatile, but sometimes you want information at a glance instead of having to open an app and wait for it to load. Android widgets let you display just about any feature you choose on the home screen, including weather apps, music widgets, or productivity tools that helpfully remind you of upcoming meetings or approaching deadlines.

8. Custom ROMs

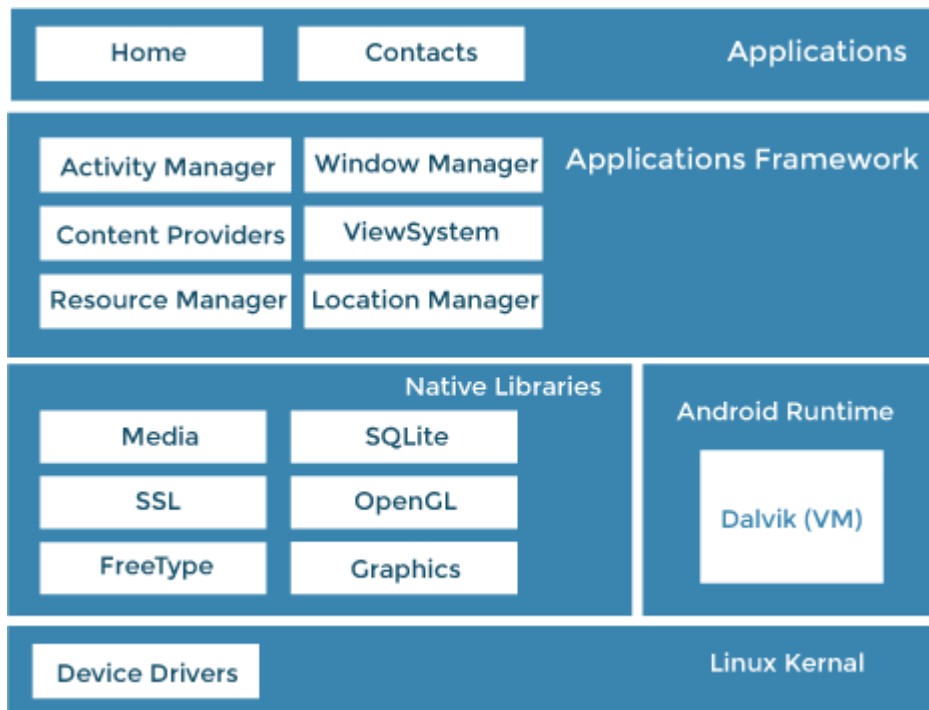
Because the Android operating system is open-source, developers can twist the current OS and build their versions, which users can download and install in place of the stock OS. Some are filled with features, while others change the look and feel of a device. Chances are, if there's a feature you want, someone has already built a custom ROM for it.

Architecture of Android OS

The android architecture contains a different number of components to support any android device needs. Android software contains an open-source Linux Kernel with many C/C++ libraries exposed through application framework services.

Among all the components, Linux Kernel provides the main operating system functions to Smartphone and Dalvik Virtual Machine (DVM) to provide a platform for running an android application. An android operating system is a stack of software components roughly divided into five sections and four main layers, as shown in the below architecture diagram.

- Applications
- Application Framework
- Android Runtime
- Platform Libraries
- Linux Kernel



1. Applications

An application is the top layer of the android architecture. The pre-installed applications like camera, gallery, home, contacts, etc., and third-party applications downloaded from the play store like games, chat applications, etc., will be installed on this layer.

It runs within the Android run time with the help of the classes and services provided by the application framework.

2. Application framework

Application Framework provides several important classes used to create an Android application. It provides a generic abstraction for hardware access and helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.

It includes different types of services, such as activity manager, notification manager, view system, package manager etc., which are helpful for the development of our application according to the prerequisite.

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications. The Android framework includes the following key services:

- **Activity Manager:** Controls all aspects of the application lifecycle and activity stack.
- **Content Providers:** Allows applications to publish and share data with other applications.
- **Resource Manager:** Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager:** Allows applications to display alerts and notifications to the user.
- **View System:** An extensible set of views used to create application user interfaces.

3. Application runtime

Android Runtime environment contains components like core libraries and the Dalvik virtual machine (DVM). It provides the base for the application framework and powers our application with the help of the core libraries.

Like *Java Virtual Machine* (JVM), *Dalvik Virtual Machine* (DVM) is a register-based virtual machine designed and optimized for Android to ensure that a device can run multiple instances efficiently.

It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard **JAVA** or **Kotlin** programming languages.

The Dalvik VM makes use of Linux core features like memory management and multithreading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Dalvik Virtual Machine | DVM

- Modern JVM provides high performance and excellent memory management.
- The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices.
- It optimizes the virtual machine for *memory, battery life and performance*.
- Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein

Difference between JVM and DVM

Java Virtual Machine	Dalvik Virtual machine
Stack-based VM that performs arithmetic and logic operations through push and pop operands. The result of operations is stored in stack memory	Register-based VM that uses registers located in the CPU to perform arithmetic and logic operations
Java source code is compiled into Java byte code format(.class file) that further translates into machine code	Source code files are first of all compiled into Java byte code format like JVM. Further, the DEX compiler(dx tool) converts the Java bytecode into Dalvik bytecode(classes.dex) file that will be used to create the .apk file
More information is required to the VM for data loading and manipulation as well as method loading in the stack data structure.	Instruction size is larger as it needs to encode the source and destination register of the VM.
Compiled byte code size is compact because the location of the operand is implicitly on the operand stack	Compiled byte code size is larger as each instruction needs all implicit operands
The executable file for the device is .jar file	The executable file for the device is .apk file .
A single instance of JVM is configured with shared processes and memory space in order to run all deployed applications.	The device runs multiple DVM instances with a separate process in shared memory space to deploy the code of each application
Supports multiple operating systems like Linux, Windows, and Mac OS	Support only the Android operation system

4. Platform libraries

The Platform Libraries include various C/C++ core libraries and Java-based libraries such as Media, Graphics, Surface Manager, OpenGL, etc., to support Android development.

- **Android.app:** Provides access to the application model and is the cornerstone of all Android applications.
- **Android.content:** Facilitates content access, publishing and messaging between applications and application components.

- **Android.database:** Used to access data published by content providers and includes SQLite database, management classes.
- **Android.OpenGL:** A Java interface to the OpenGL ES 3D graphics rendering API.
- **Android.os:** Provides applications with access to standard operating system services, including messages, system services and inter-process communication.
- **Android.text:** Used to render and manipulate text on a device display.
- **Android.view:** The fundamental building blocks of application user interfaces.
- **Android.widget:** A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **WebKit:** A set of classes intended to allow web-browsing capabilities to be built into applications.
- **media:** Media library provides support to play and record an audio and video format.
- **surface manager:** It is responsible for managing access to the display subsystem.
- **SQLite:** It provides database support, and FreeType provides font support.
- **SSL:** Secure Sockets Layer is a security technology to establish an encrypted link between a web server and a web browser.

5. Linux Kernel

Linux Kernel is the heart of the android architecture. It manages all the available drivers such as display, camera, Bluetooth, audio, memory, etc., required during the runtime.

The Linux Kernel will provide an abstraction layer between the device hardware and the other android architecture components. It is responsible for the management of memory, power, devices etc. The features of the Linux kernel are:

- **Security:** The Linux kernel handles the security between the application and the system.
- **Memory Management:** It efficiently handles memory management, thereby providing the freedom to develop our apps.
- **Process Management:** It manages the process well, allocates resources to processes whenever they need them.
- **Network Stack:** It effectively handles network communication.
- **Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

Advantages of Android Operating System

We considered every one of the elements on which Android is better as thought about than different platforms. Below are some important advantages of Android OS, such as:

- **Android Google Developer:** The greatest favorable position of Android is Google. Google claims an android operating system. Google is a standout amongst the most trusted and rumored item on the web. The name Google gives trust to the clients to purchase Android gadgets.
- **Android Users:** Android is the most utilized versatile operating system. More than a billion individuals' clients utilize it. Android is likewise the quickest developing operating system in the world. Various clients increment the number of applications and programming under the name of Android.
- **Android Multitasking:** The vast majority of us admire this component of Android. Clients can do heaps of undertakings on the double. Clients can open a few applications on the double and oversee them very. Android has incredible UI, which makes it simple for clients to do multitasking.
- **Google Play Store App:** The best part of Android is the accessibility of many applications. Google Play store is accounted for as the world's largest mobile store. It has practically everything from motion pictures to amusements and significantly more. These things can be effortlessly downloaded and gotten to through an Android phone.
- **Android Notification and Easy Access:** Without much of a stretch, one can access their notice of any SMS, messages, or approaches their home screen or the notice board of the android phone. The client can view all the notifications on the top bar. Its UI makes it simple for the client to view more than 5 Android notices immediately.
- **Android Widget:** Android operating system has a lot of widgets. This gadget improves the client encounter much and helps in doing multitasking. You can include any gadget relying on the component you need on your home screen. You can see warnings, messages, and a great deal more use without opening applications.

Disadvantages of Android Operating System

We know that the Android operating system has a considerable measure of interest for users nowadays. But at the same time, it most likely has a few weaknesses. Below are the following disadvantages of the android operating system, such as:

- **Android Advertisement pop-ups:** Applications are openly accessible in the Google play store. Yet, these applications begin demonstrating tons of advertisements on the notification bar and

over the application. This promotion is extremely difficult and makes a massive issue in dealing with your Android phone.

- **Android require Gmail ID:** You can't get to an Android gadget without your email ID or password. Google ID is exceptionally valuable in opening Android phone bolts as well.
- **Android Battery Drain:** Android handset is considered a standout amongst the most battery devouring operating systems. In the android operating system, many processes are running out of sight, which brings about the draining of the battery. It is difficult to stop these applications as the lion's share of them is system applications.
- **Android Malware/Virus/Security:** Android gadget is not viewed as protected when contrasted with different applications. Hackers continue attempting to take your data. It is anything but difficult to target any Android phone, and each day millions of attempts are done on Android phones

Android Applications and Their Categories

Android is an open-source operating system, based on the Linux kernel and used in mobile devices like smart phones, tablets, etc. Further, it was developed for smart watches and Android TV. Each of them has a specialized interface. Android has been one of the best-selling OS for smart phones. Android OS was developed by Android Inc. which Google bought in 2005. In this article, we will discuss android application types and categories as well as their advantages and disadvantages. Firstly let's see the types of applications, there are mainly 3 types of Android Applications.

Types of Android Applications

1. Native Apps

Native apps are built for particular operating systems, which are mostly Android and IOS. Also, there are more OS for mobile applications: Blackberry and Windows. This is available for download on Google Play Store and for IOS Apple App Store. Native apps are generally built to make the most of all the features and tools of the phones such as contacts, cameras, sensors, etc. Native apps ensure high performance and stylish user experience as the developers use the native device UI to build apps. WhatsApp, Spotify, Pokemon GO, etc. are examples of Natives apps. Android apps are built using **Java**, **Kotlin**, and **Flutter**, for the frontend, it uses the **XML** scripting language. And IOS apps built using **Swift**, **Flutter/ Dart**, and **C#**.

Advantages:

- Native apps are designed for the particular operating system and it gives the best user experience.
- Native apps are built with separate gestures it gives a good experience to users and it is very useful for all users.

Disadvantages:

- Native apps are costly in comparison to others because they want separate maintenance.
- Requires a separate codebase to add new features.

2. Web Apps

Web applications are built only the run on browsers. They are mainly the integrations of **HTML**, **CSS**, and **Javascript**. It runs on Chrome, Firefox, and other browsers. The responsiveness and functionality of the web apps could easily be confused with a native app since both the Native and web apps have almost the same features and responsive nature. And one of the major differences between the two is that native mobile apps can function both in the offline mode without an active internet connection and the online mode, whereas the web apps require an active internet connection for them to work. Gmail, Canva, and Google Docs are the best examples of web apps.

Advantages:

- Easy to build
- Web apps are used less storage than other applications.
- Web Apps are preinstalled on all devices.
- Web applications are easily accessible in any type of application.

Disadvantages:

- Local resources are not available in web apps.
- Depends on internet networks/ connections.

3. Hybrid Apps

Hybrid applications are also called Cross Platform Applications. Hybrid applications are runs on multiple platforms like Android and IOS. Also, these are made from the integration of web and native applications. Because hybrid apps use a single codebase, they can be deployed across devices. For example, when we build the android application, we can also launch it on IOS. As a cross-platform development option, developers have more freedom when designing their applications as they do not need to stick to specific design guidelines from either apple or google. Instagram, Uber, and Crypto change are examples of Hybrid apps. For Hybrid application development, we use **Flutter/Dart**, **React Native**, etc.

Advantages:

- Users can use it on more than one platform.
- It is integrated with browsers.
- Maintained by many versions.
- Shareable code makes it cheaper than a native app.

Disadvantages:

- Slower compared to native apps.
- There might be some user interface issues.
- In hybrid apps have limitations in using all the Hardware and Operating Systems features.

Now let's see about some categories of Android applications.

Categories of Android Applications

Some Categories of the Android Applications:

1. **E-Commerce Apps:** E-commerce apps are an example of a B2B model. It helps to people to sell and borrow different items and it saves time and money. In e-commerce applications, we can do trading of commercial goods on online marketplaces. To buy specific items and goods, you simply need to make electronic transactions like UPI, Phonepe, etc. through your smartphone or computer. Flipkart, Amazon, OLX, and, Quiker are examples of e-commerce applications.
2. **Educational Apps:** Educational apps are too much used to improve knowledge and peoples get productivity. Apps for education can make people more interactive, more engaged, and perform better. Keeping teaching methods good is integral to getting students engaged in their studies and learning apps are a fantastic way of achieving this. For example, Google Classroom, SoloLearn, edX, Duolingo, etc.
3. **Social Media Apps:** Social media apps give the opportunity to the peoples connect and communicate together. These apps are mainly used for sharing purposes and making fun. Many people's use social media applications for influence, marketing/ business, entrepreneurship, etc. Instagram, Facebook, WhatsApp, YouTube, LinkedIn, etc. are examples of social media applications.
4. **Productivity Apps:** Productivity apps typically organize and complete complex tasks for you, anything from sending an email to figuring out a tip. The easy-to-use Google Drive app gives users access to all of the files saved to the cloud-based storage service across multiple devices. Productivity applications arise in many different forms and they often

take a different approach to improving your workflow. For example, Hive, Todoist, Google Docs, etc.

5. **Entertainment Apps:** Entertainment apps are widely used apps worldwide. It contains OTT platforms and novels and other content. These platforms entertain people and give them much more knowledge about different things. Everyone is watching OTT platforms and those are trending these days, and their development is also in demand all over the world. Hotstar, Netflix, and Amazon prime video are the best examples of these entertainments applications.

ANDROID - Environment Setup:

Android application development on either of the following operating systems –

- Microsoft Windows XP or later version.
- Mac OS X 10.5.8 or later version with Intel chip.
- Linux including GNU C Library 2.7 or later.

Second point is that all the required tools to develop Android applications are freely available 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
- Android Studio

Here last two components are optional and if you are working on Windows machine then these components make your life easy while doing Java based application development. So let us have a look how to proceed to set required environment.

Step 1: Set-up Java Development Kit (JDK)

You can download the latest version of Java JDK from Oracle's Java site – [Java SE Downloads](#). You will find instructions for installing JDK in downloaded files, follow the given instructions to install and configure the setup. Finally set PATH and JAVA_HOME environment variables to refer to the directory that contains **java** and **javac**, typically `java_install_dir/bin` and `java_install_dir` respectively.

If you are running Windows and installed the JDK in `C:\jdk1.8.0_102`, you would have to put the following line in your `C:\autoexec.bat` file.

```
set PATH=C:\jdk1.8.0_102\bin;%PATH%
set JAVA_HOME=C:\jdk1.8.0_102
```

Alternatively, you could also right-click on *My Computer*, select *Properties*, then *Advanced*, then *Environment Variables*. Then, you would update the PATH value and press the OK button.

On Linux, if the SDK is installed in /usr/local/jdk1.8.0_102 and you use the C shell, you would put the following code into your **.cshrc** file.

```
setenv PATH /usr/local/jdk1.8.0_102/bin:$PATH
setenv JAVA_HOME /usr/local/jdk1.8.0_102
```

Alternatively, if you use Android studio, then it will know automatically where you have installed your Java.

Android IDEs

There are so many sophisticated Technologies are available to develop android applications, the familiar technologies, which are predominantly using tools as follows

- Android Studio
- Eclipse IDE(Deprecated)

Step 2 : Setup Android SDK

You can download the latest version of Android SDK from Android's official website: <http://developer.android.com/sdk/index.html>. If you are installing SDK on Windows machine, then you will find ainstaller_rXX-windows.exe, so just download and run this exe which will launch Android SDK Tool Setup wizard to guide you throughout the installation, so just follow the instructions carefully. Finally, you will have Android SDK Tools installed on your machine. If you are installing SDK either on Mac OS or Linux, check the instructions provided along with the downloaded android-sdk_rXX-macosx.zip file for Mac OS and android-sdk_rXXlinux.tgz file for Linux. This tutorial will consider that you are going to setup your environment on Windows machine having Windows 7 operating system. So let's launch Android SDK Manager using the option All Programs > Android SDK Tools > SDK Manager.

Once you launched SDK manager, it is time to install other required packages. By default it will list down total 7 packages to be installed, but we will suggest to de-select Documentation for Android SDK and Samples for SDK packages to reduce installation time. Next click the Install 7 Packages button to proceed.

If you agree to install all the packages, select Accept All radio button and proceed by clicking Install button. Now let SDK manager do its work and you go, pick up a cup of coffee and wait until all the packages are installed. It may take some time depending on your internet connection. Once all the packages are installed, you can close SDK manager using top-right cross button.

Step 3 -Setup Eclipse IDE

All the examples in this tutorial have been written using Eclipse IDE. So we would suggest you should have latest version of Eclipse installed on your machine. To install Eclipse IDE, download the latest Eclipse binaries from <http://www.eclipse.org/downloads/>. Once you have downloaded the installation, unpack the binary distribution into a convenient location.

For example in C:\eclipse on windows, or /usr/local/eclipse on Linux and finally set PATH variable appropriately. Eclipse can be started by executing the following commands on windows machine, or you can simply double click on eclipse.exe

```
%C:\eclipse\eclipse.exe
```

Eclipse can be started by executing the following command on Linux machine:

```
$/usr/local/eclipse/eclipse
```

Step 4 -Setup Android Development Tools (ADT) Plugin

This step will help you in setting Android Development Tool plugin for Eclipse. Let's start with launching Eclipse and then, choose Help > Software Updates > Install New Software.

Now use Add button to add ADT Plugin as name and <https://dlssl.google.com/android/eclipse/> as the location. Then click OK to add this location. As soon as you will click OK button to add this location, Eclipse starts searching for the plug-in available in the given location and finally lists down the found plugins.

Now select all the listed plug-ins using Select All button and click Next button which will guide you ahead to install Android Development Tools and other required plugins.

Step 5 - 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 using Eclipse menu options Window > AVD Manager> which will launch Android AVD Manager. Use New button to create a new Android Virtual Device and enter the following information, before clicking Create AVD button.

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 restart 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.

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**.

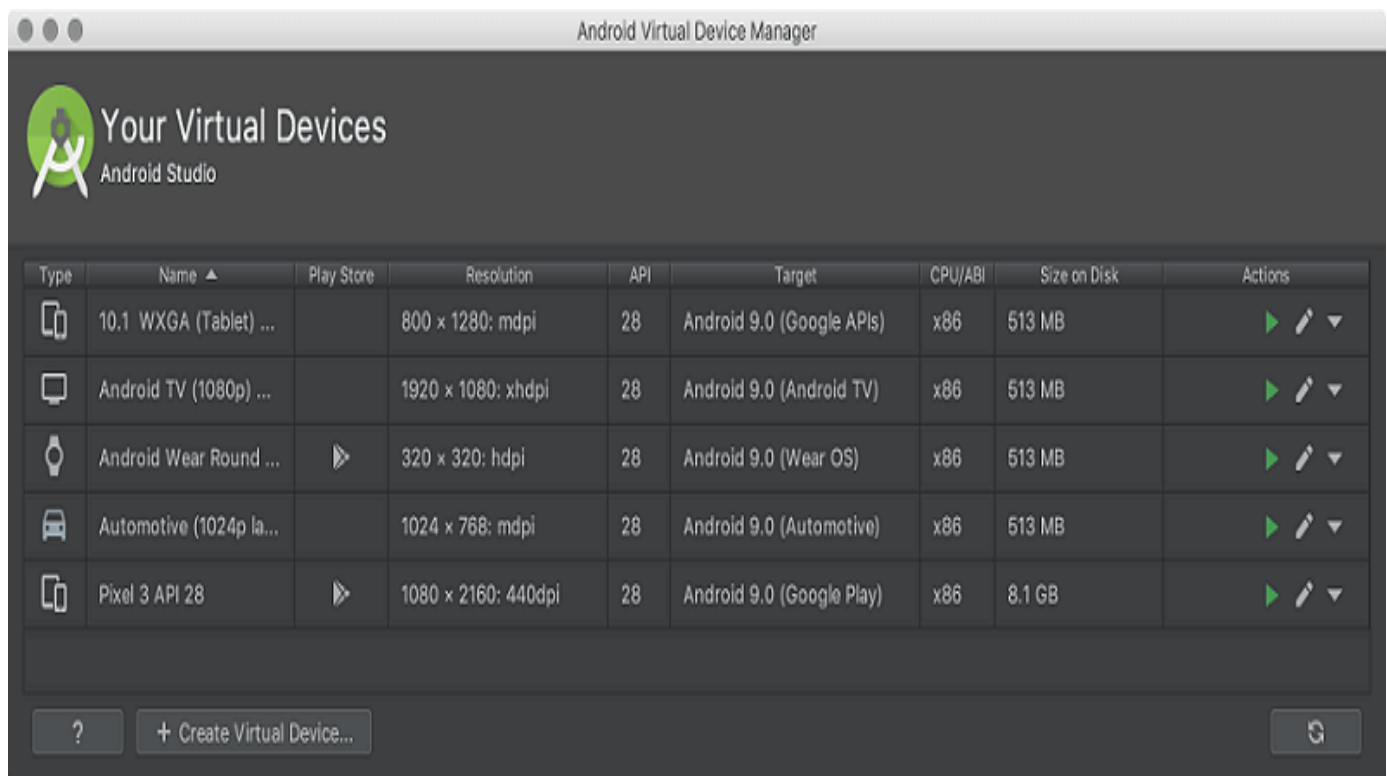
Run an Android app on the Emulator

We can run an Android app from 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.

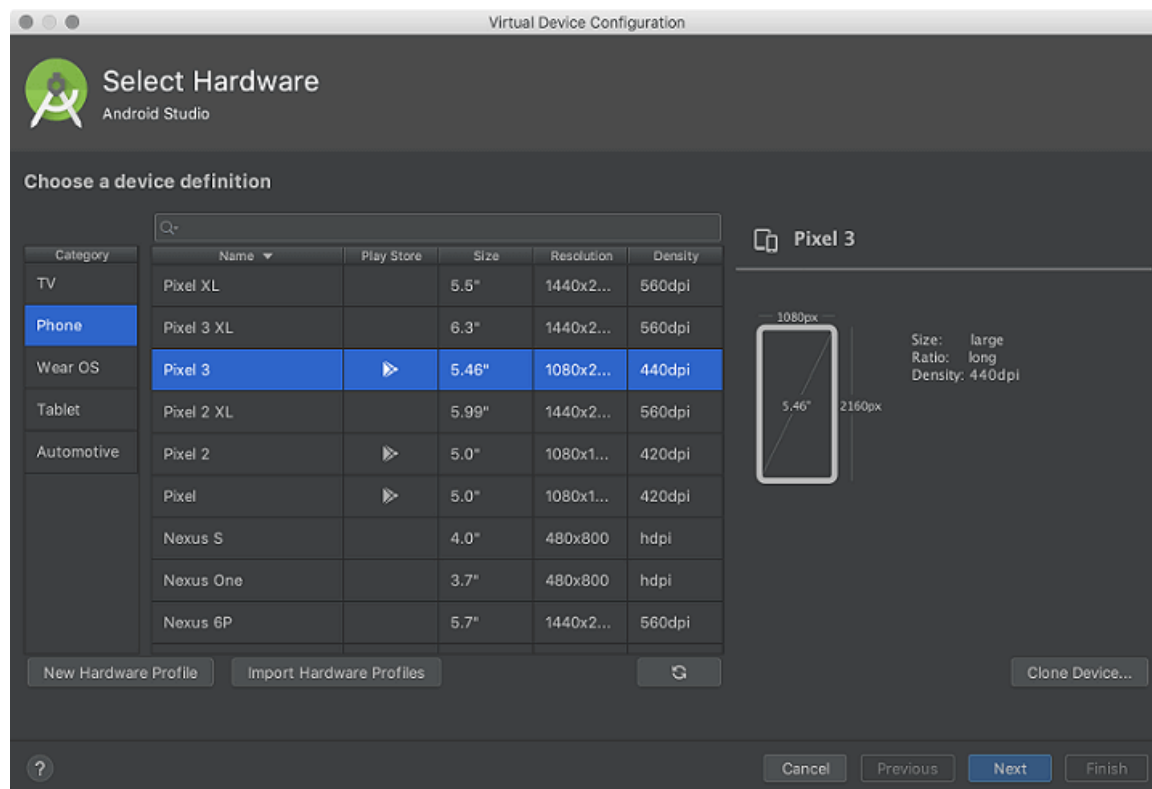
To start the Android Emulator and run an application in our project:

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:-

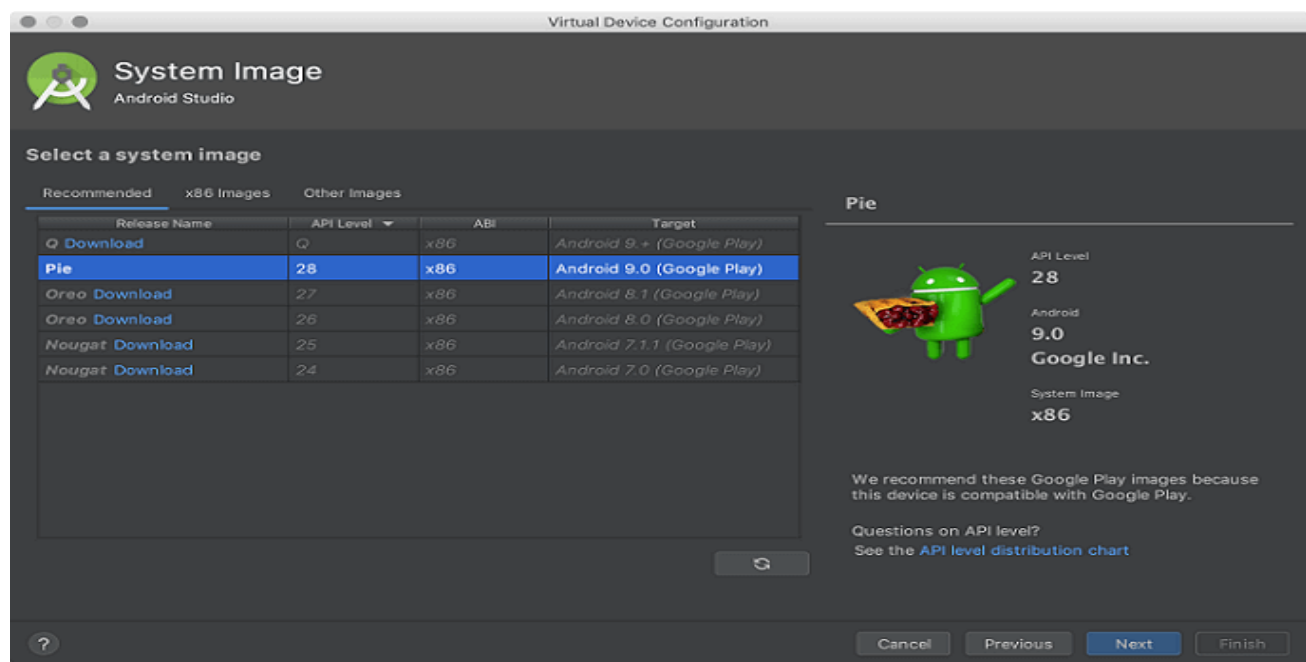
Open the AVD Manager by clicking **Tools > AVD Manager**.



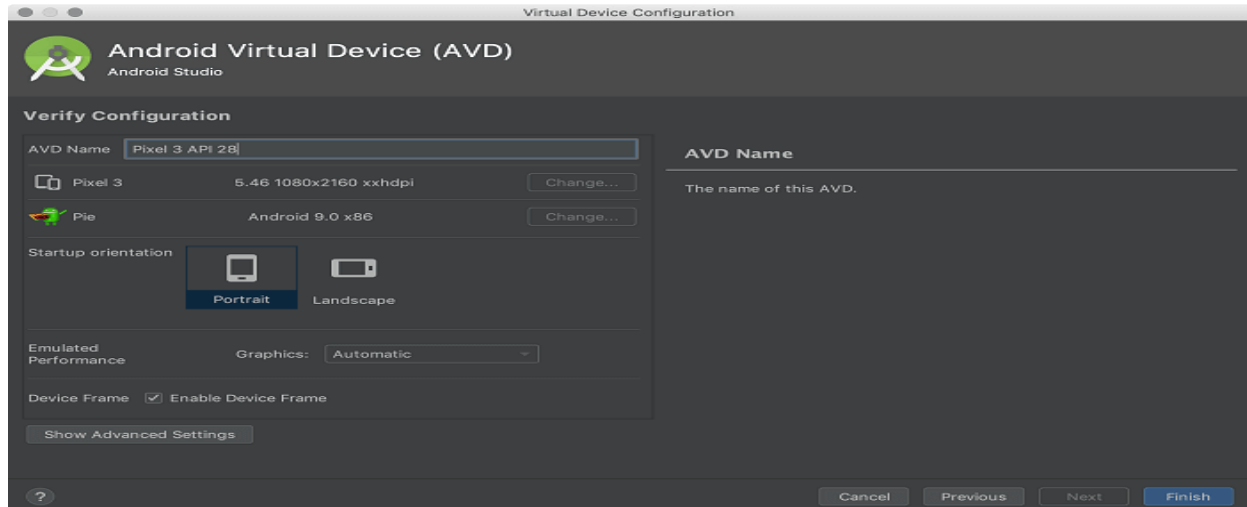
Click on **Create Virtual Device**, at the bottom of the AVD Manager dialog. Then **Select Hardware** page appears.



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.

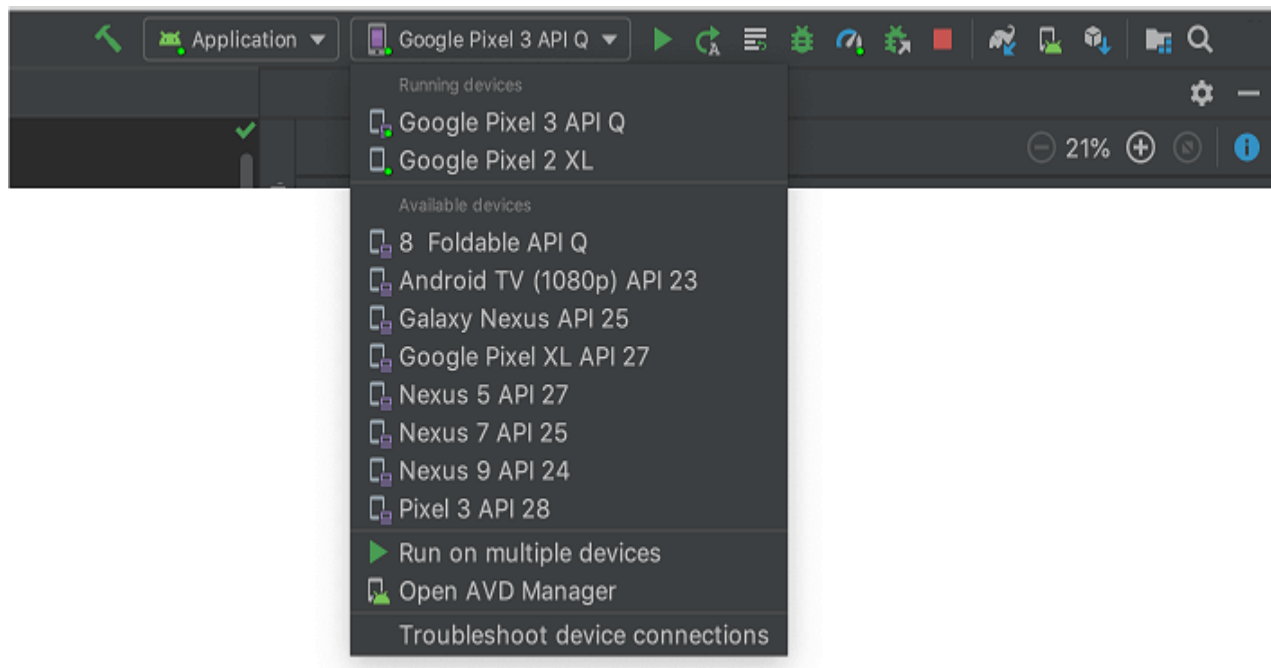


Select the system image for the particular API level and click **Next**. This leads to open a **Verify Configuration** page.



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

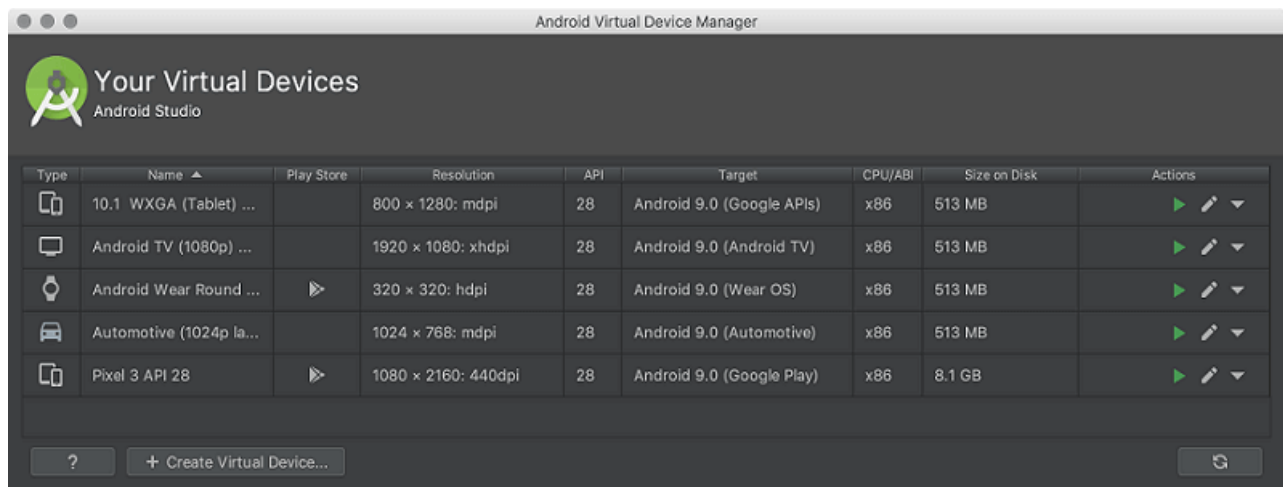
To start the emulator:

1. Open the AVD Manager.
2. 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**.

Android Core Building Blocks

An android **component** is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Activity

An activity is a class that represents a single screen. It is like a Frame in AWT.

View

A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent

Intent is used to invoke components. It is 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.

For example, you may write the following code to view the webpage.

```
Intent intent=new Intent(Intent.ACTION_VIEW);  
intent.setData(Uri.parse("http://www.javatpoint.com"));  
startActivity(intent);
```

It is a powerful inter-application message-passing framework. They are extensively used throughout Android.

Intents can be used to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service or Broadcast Receiver or to request action be performed on a particular piece of data.

Service

Service is a background process that can run for a long time.

There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

Content Provider

Content Providers are used to share data between the applications.

Fragment

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

AndroidManifest.xml

It contains information's about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

Android Virtual Device (AVD)

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

Components of an Android Application

There are some necessary building blocks that an Android application consists of. These loosely coupled components are bound by the application manifest file which contains the description of each component and how they interact.

The manifest file also contains the app's metadata, its hardware configuration, and platform requirements, external libraries, and required permissions. There are the following main components of an android app:

There are following five main components that can be used within an Android application –

- ✓ Activities
- ✓ Services
- ✓ Broadcast Receivers
- ✓ Content Providers

Sl.No	Components & Description
1	Activities They dictate the UI and handle the user interaction to the smart phone screen.
2	Services They handle background processing associated with an application.
3	Broadcast Receivers They handle communication between Android OS and applications.
4	Content Providers They handle data and database management issues.

1. Activities

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of **Activity** class as follows –

```
public class MainActivity extends Activity
{
}
```

2. Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity

A service is implemented as a subclass of **Service** class as follows –

```
public class MyService extends Service
{
}
```

3. Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. 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.

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and each message is broadcaster as an **Intent** object.

```
public class MyReceiver extends BroadcastReceiver
{
    public void onReceive(context,intent){ }
}
```

Widgets

These are the small visual application components that you can find on the home screen of the devices. They are a special variation of Broadcast Receivers that allow us to create dynamic, interactive application components for users to embed on their Home Screen.

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. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider
{
    public void onCreate(){ }
}
```

Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are –

Sl. No	Components & Description
1	Fragments Represents a portion of user interface in an Activity.

2	Views UI elements that are drawn on-screen including buttons, lists forms etc.
3	Layouts View hierarchies that control screen format and appearance of the views.
4	Intents Messages wiring components together.
5	Resources External elements, such as strings, constants and drawable pictures.
6	Manifest Configuration file for the application.

Notifications

Notifications are the application alerts that are used to draw the user's attention to some particular app event without stealing focus or interrupting the current activity of the user. They are generally used to grab user's attention when the application is not visible or active, particularly from within a Service or Broadcast Receiver. Examples: E-mail popups, Messenger popups, etc.

ANDROID – HELLO WORLD EXAMPLE

Let us start actual programming with Android Framework. Before you start writing your first example using Android SDK, you have to make sure that you have set-up your Android development environment.

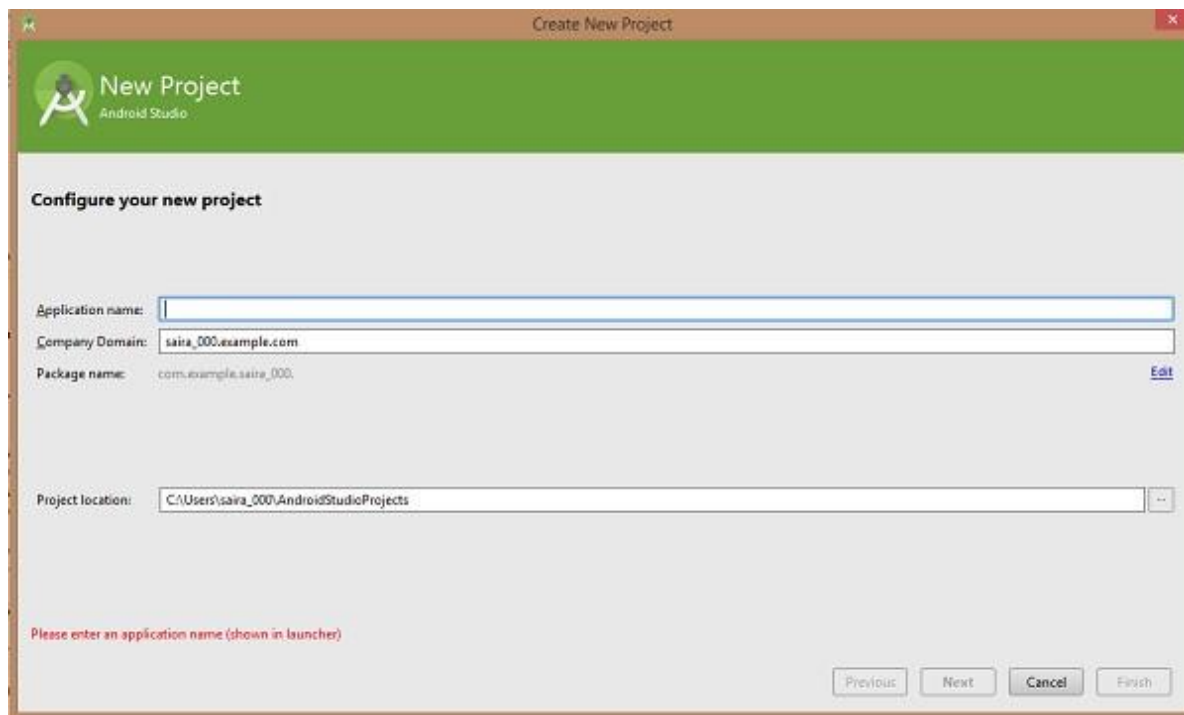
So let us proceed to write a simple Android Application which will print "Hello World!".

Create Android Application

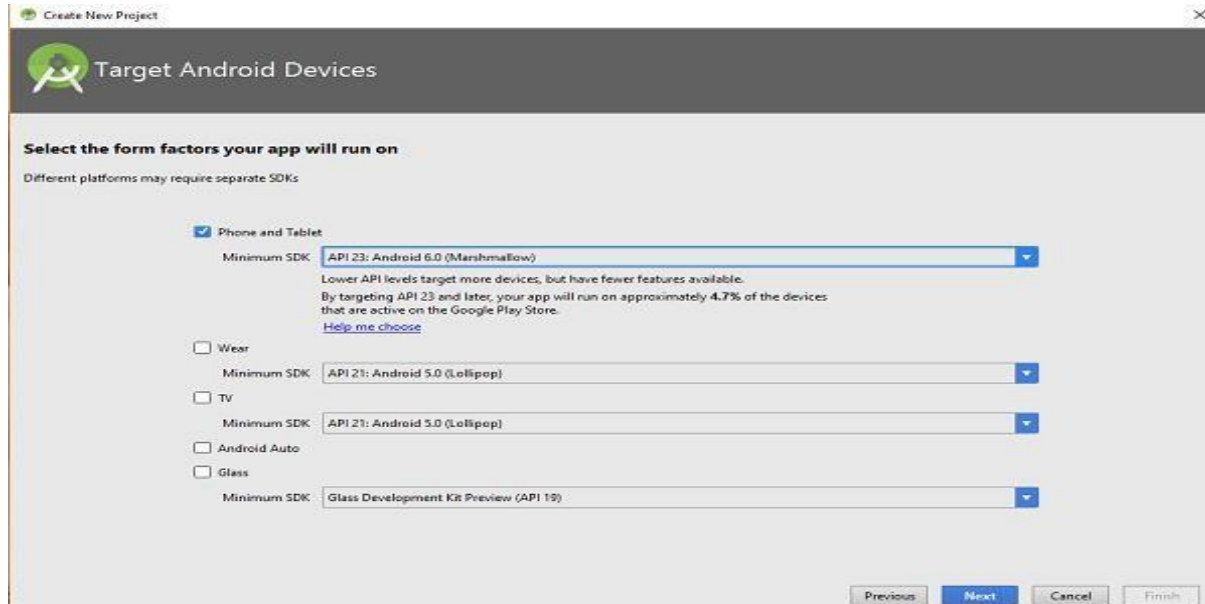
The first step is to create a simple Android Application using Android studio. When you click on Android studio icon, it will show screen as shown below



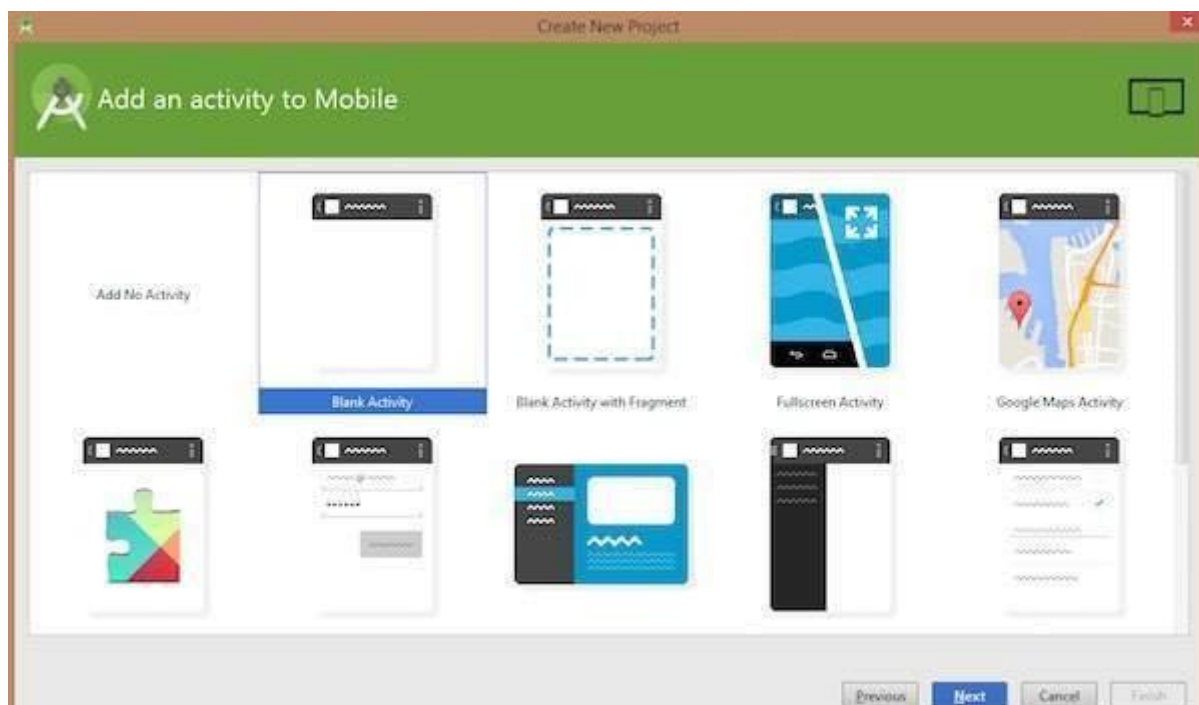
You can start your application development by calling start a new android studio project. in a new installation frame should ask Application name, package information and location of the project.—



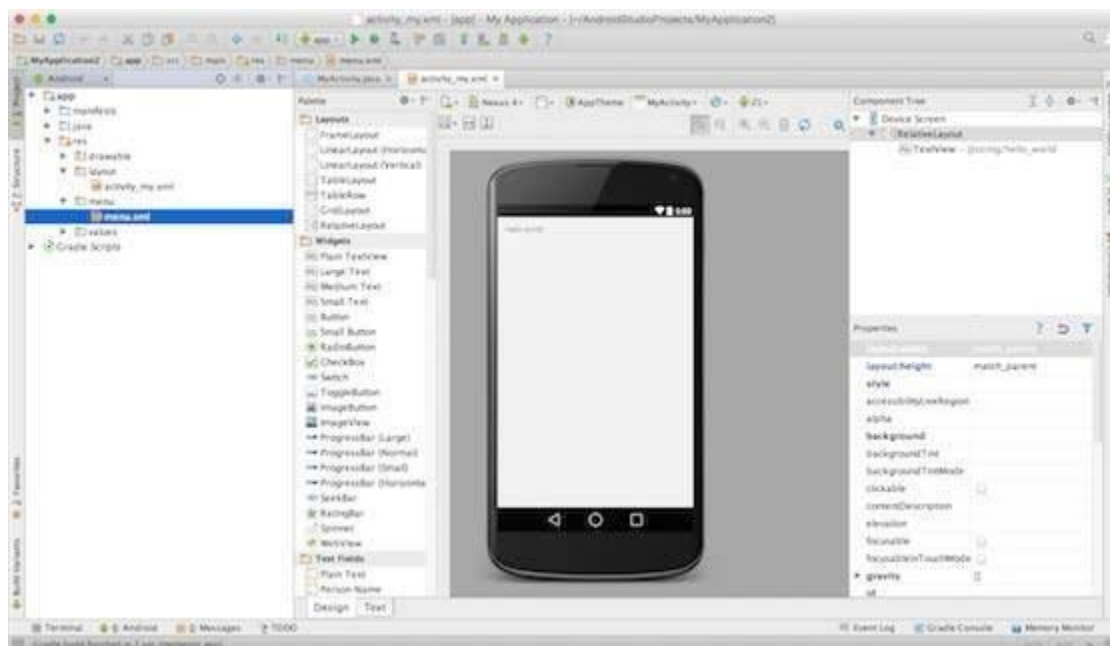
After entered application name, it going to be called select the form factors your application runs on, here need to specify Minimum SDK, in our tutorial, I have declared as API23: Android 6.0(Mashmallow) –



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

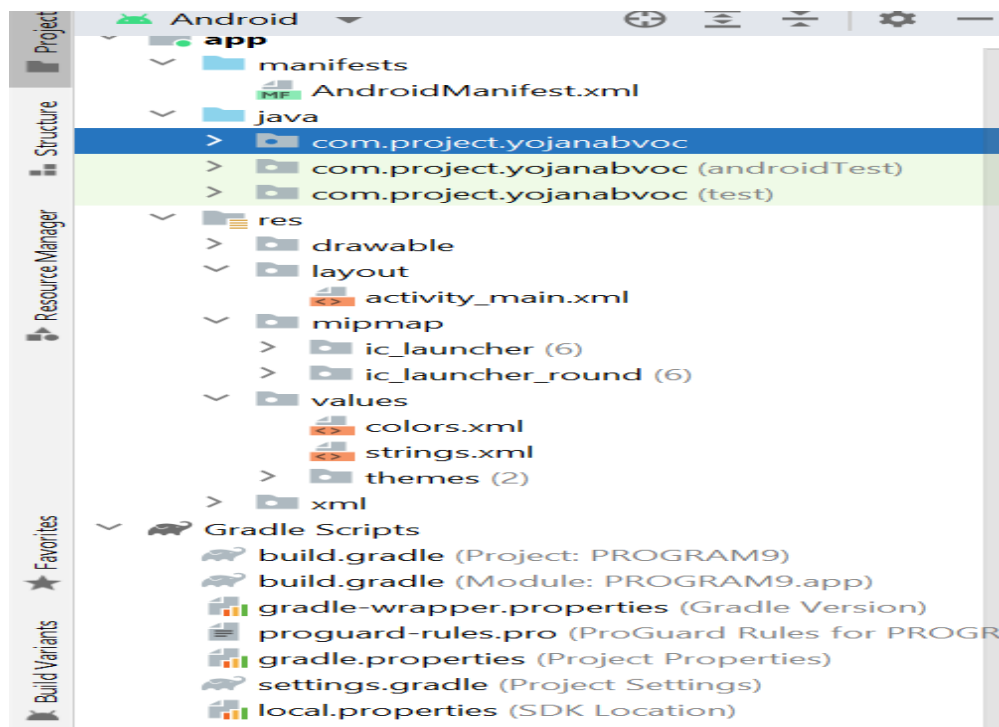


At the final stage it going to be open development tool to write the application code.



Anatomy of Android Application

Before you run your app, you should be aware of a few directories and files in the Android project –



Sl.No.	Folder, File & Description
1	Java This contains the .java source files for your project. By default, it includes an <i>MainActivity.java</i> source file having an activity class that runs when your app is launched using the app icon.
2	res/drawable-hdpi This is a directory for drawable objects that are designed for high-density screens.
3	res/layout This is a directory for files that define your app's user interface.
4	res/values This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
5	AndroidManifest.xml This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.
6	Build.gradle This is an auto generated file which contains <code>compileSdkVersion</code> , <code>buildToolsVersion</code> , <code>applicationId</code> , <code>minSdkVersion</code> , <code>targetSdkVersion</code> , <code>versionCode</code> and <code>versionName</code>

Following section will give a brief overview of the important application files.

The Main Activity File

The main activity code is a Java file **MainActivity.java**. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application. Following is the default code generated by the application wizard for *Hello World!* application –

```
package com.example.helloworld;
```

```
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);
    }
}
```

Here, *R.layout.activity_main* refers to the *activity_main.xml* file located in the *res/layout* folder. The *onCreate()* method is one of many methods that are figured when an activity is loaded.

The Manifest File

Whatever component you develop as a part of your application, you must declare all its components in a *manifest.xml* which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. For example, a default manifest file will look like as following file –

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Here `<application>...</application>` tags enclosed the components related to the application. Attribute *android:icon* will point to the application icon available under *res/drawable-hdpi*. The application uses the image named *ic_launcher.png* located in the *drawable* folders

The `<activity>` tag is used to specify an activity and *android:name* attribute specifies the fully qualified class name of the *Activity* subclass and the *android:label* attributes specifies a string to use as the label for the activity. You can specify multiple activities using `<activity>` tags.

The **action** for the intent filter is named *android.intent.action.MAIN* to indicate that this activity serves as the entry point for the application. The **category** for the intent-filter is named *android.intent.category.LAUNCHER* to indicate that the application can be launched from the device's launcher icon.

The *@string* refers to the *strings.xml* file explained below. Hence, *@string/app_name* refers to the *app_name* string defined in the *strings.xml* file, which is "HelloWorld". Similar way, other strings get populated in the application.

Following is the list of tags which you will use in your manifest file to specify different Android application components –

- `<activity>` elements for activities
- `<service>` elements for services
- `<receiver>` elements for broadcast receivers
- `<provider>` elements for content providers

The Strings File

The **strings.xml** file is located in the *res/values* folder and it contains all the text that your application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content. For example, a default strings file will look like as following file –

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

The Layout File

The **activity_main.xml** is a layout file available in *res/layout* directory, that is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout –

```
<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:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:padding="@dimen/padding_medium"
    android:text="@string/hello_world"
    tools:context=".MainActivity" />
```

```
</RelativeLayout>
```

This is an example of simple *RelativeLayout* which we will study in a separate chapter. The *TextView* is an Android control used to build the GUI and it have various attributes like *android:layout_width*, *android:layout_height* etc which are being used to set its width and height etc.. The *@string* refers to the strings.xml file located in the res/values folder. Hence, *@string/hello_world* refers to the hello string defined in the strings.xml file, which is "Hello World!".

R. File

- ▶ **Android R.java** is an auto-generated file by *aapt* (Android Asset Packaging Tool) that contains resource IDs for all the resources of *res/* directory.
- ▶ If you create any component in the *activity_main.xml* file, id for the corresponding component is automatically created in this file. This id can be used in the activity source file to perform any action on the component.

/ AUTO-GENERATED FILE. DO NOT MODIFY. This class was automatically generated by the aapt tool from the resource data it found. It should not be modified by hand. */*

```
package com.example.helloandroid;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int menu_settings=0x7f070000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int activity_main=0x7f060000;
    }
}
```

```

}
public static final class string {
    public static final int app_name=0x7f040000;
    public static final int hello_world=0x7f040001;
    public static final int menu_settings=0x7f040002;
}
public static final class style {
    /** Base application theme, dependent on API level. This theme is replaced by AppBaseTheme from res/values-
vXX/styles.xml on newer devices. Theme customizations available in newer API levels can go in res/values-
vXX/styles.xml, while customizations related to backwardcompatibility can go here. Base application theme for API 11+.
This theme completely replaces AppBaseTheme from res/values/styles.xml on API 11+ devices.
API 11 theme customizations can go here.
Base application theme for API 14+. This theme completely replaces
AppBaseTheme from BOTH res/values/styles.xml and res/values-v11/styles.xml on API 14+ devices.
API 14 theme customizations can go here. */
    public static final int AppBaseTheme=0x7f050000;
    /** Application theme. All customizations that are NOT specific to a particular API-level can go here. */
    public static final int AppTheme=0x7f050001;
}
}

```

“HELLO WORLD” Program

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.javatpoint.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);  
    }  
}
```

Running the Application

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

