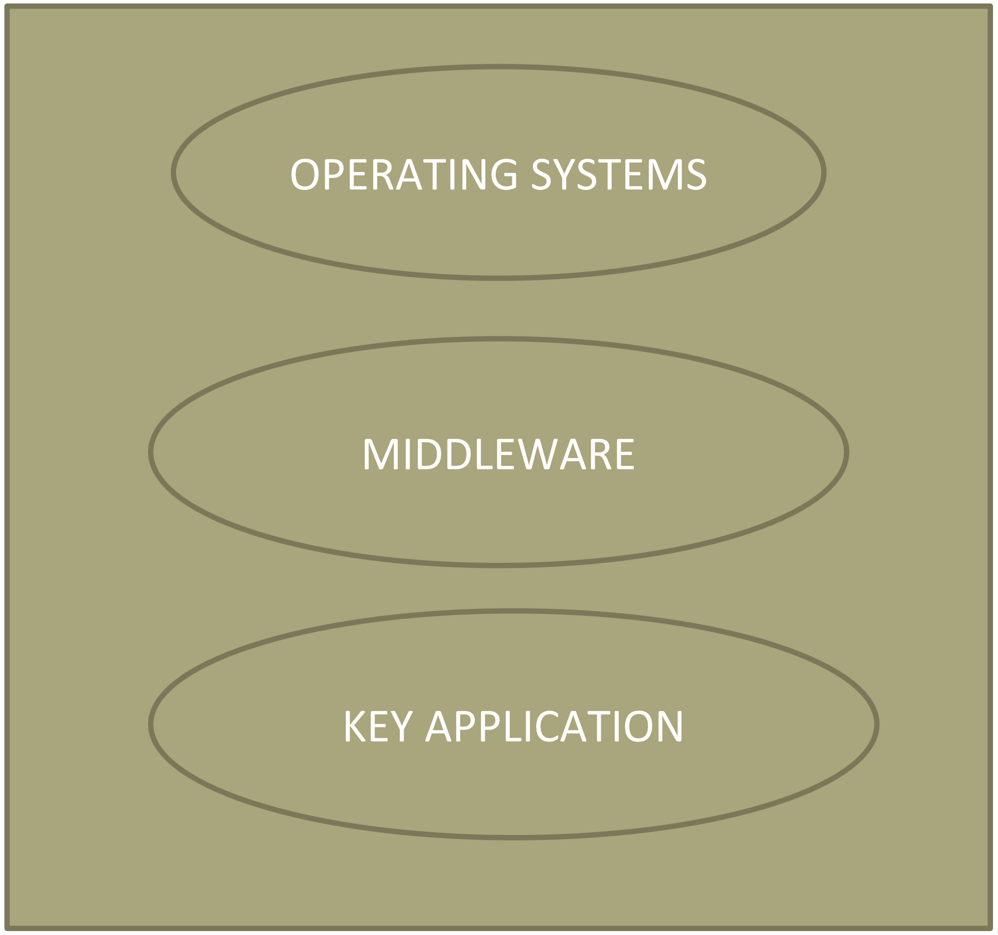
MODULE 1

What is Android??

- **Android** is a Linux based operating system it is designed primarily for touch screens mobile devices such as smartphones and tablet computers.
- An open source technology that allows software to be freely modified and distributed by device manufacturers, wireless carriers and developers.
- Android was unveiled during 2007 along with the founding of “**Open Handset Alliance”.**

- Android is a stack of software for mobile devices such as Operating system, Middleware and Key applications.

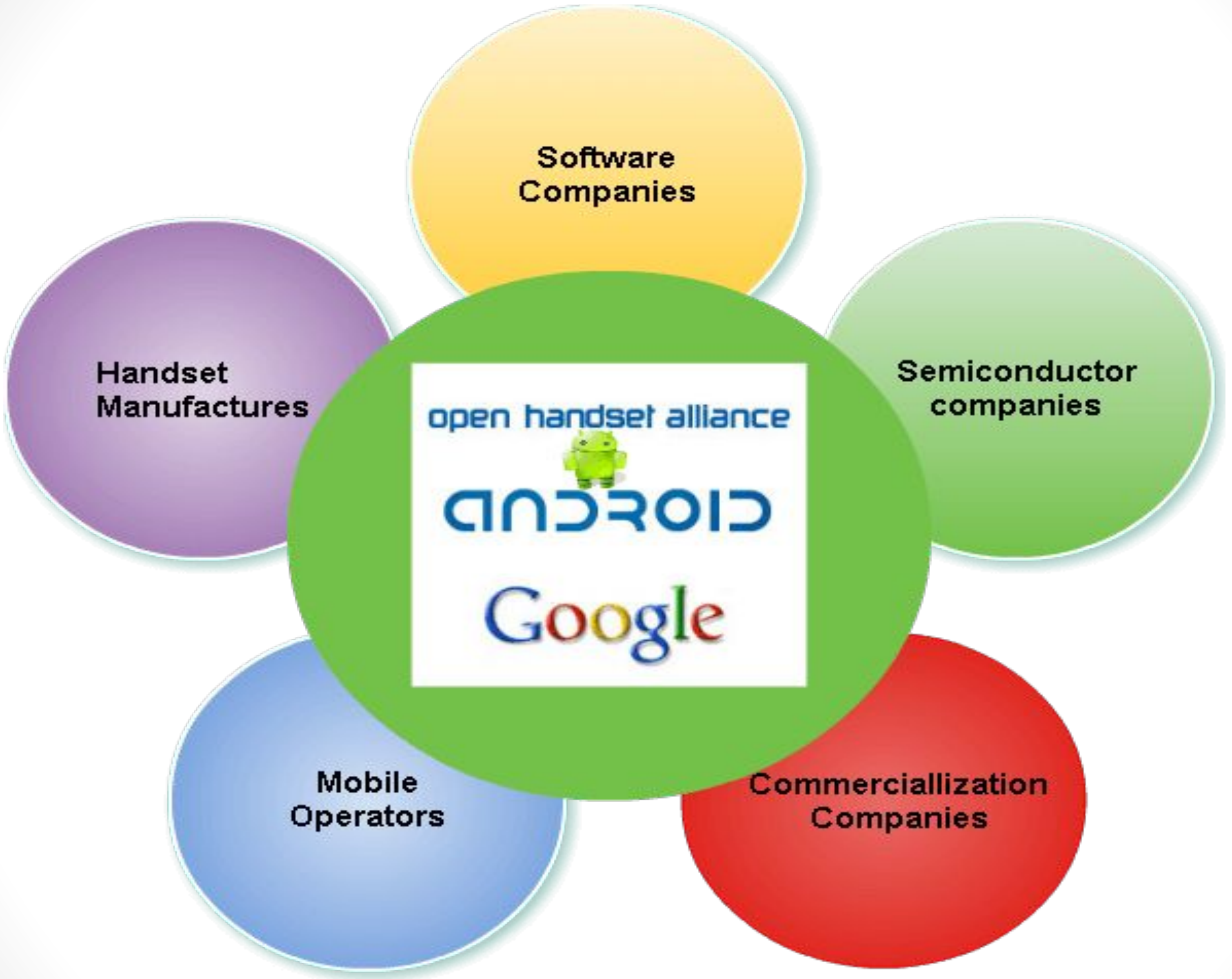


OPEN HANDSET ALLIANCE

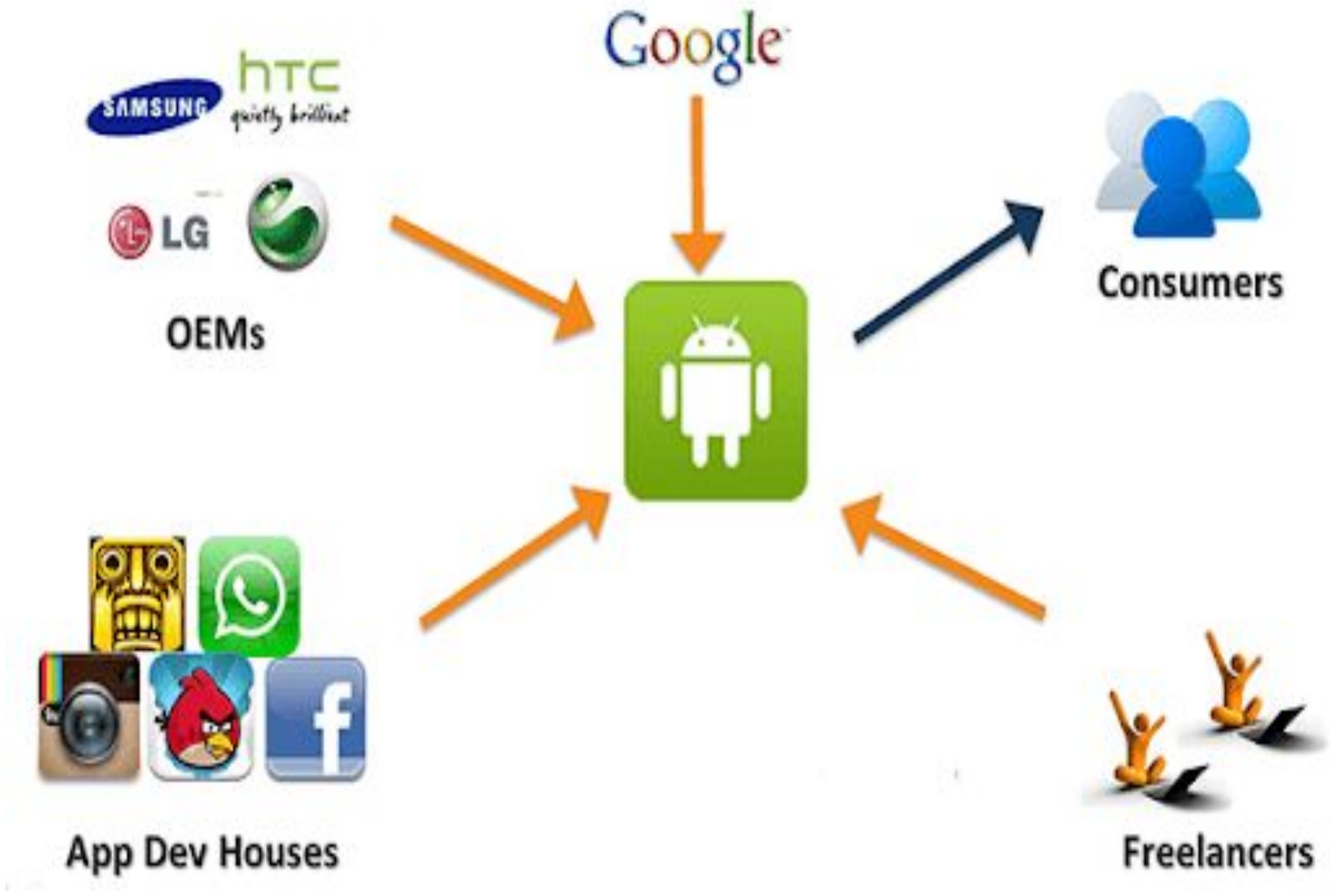
- An organization founded in 2007 by Google, T-Mobile, Qualcomm, Motorola and others that sponsors and promotes the Android open mobile phone platform.

- Based on Linux, Android was **developed to compete with all cellphone platforms including Windows Mobile and Apple's iPhone** by offering an open platform that encourages third-party application development.

- OHA members are primarily **mobile operators, handset manufacturers, software development firms, semiconductor companies and commercialization companies**. Members share a commitment to expanding the commercial viability of open platform development.


Software Companies
Handset Manufactures
Semiconductor companies
open handset alliance
ANDROID
Google
Mobile Operators
Commerciallization Companies

ANDROID ECHOSYSTEM



Stake holders of Android Echosystem-Consumers that own Android devices

- **Google**-develops Android
- **OEMs(Original Equipment Manufacturers)-**manufacture hardware as well as custom application components
- **Application Development Companies**-employ Android developers and also contract out the product development to services companies.
- **Freelance Android developers**-Developers create their own applications and publish them on Google playstore.

Why ANDROID??

- Browser
- Desktop
- Connectivity
- Multi-Notification
- Endless Personalization
- Market
- Google Integration
- Open Source
- Open to Carrier
- Future

WHAT IS AN APK?

APK(ANDROID PACKAGE KIT or ANDROID APPLICATION PACKAGE)-file format used by Android for the distribution and installation of mobile apps, mobile games etc.

This package contains all the elements an app needs to install correctly on your device.

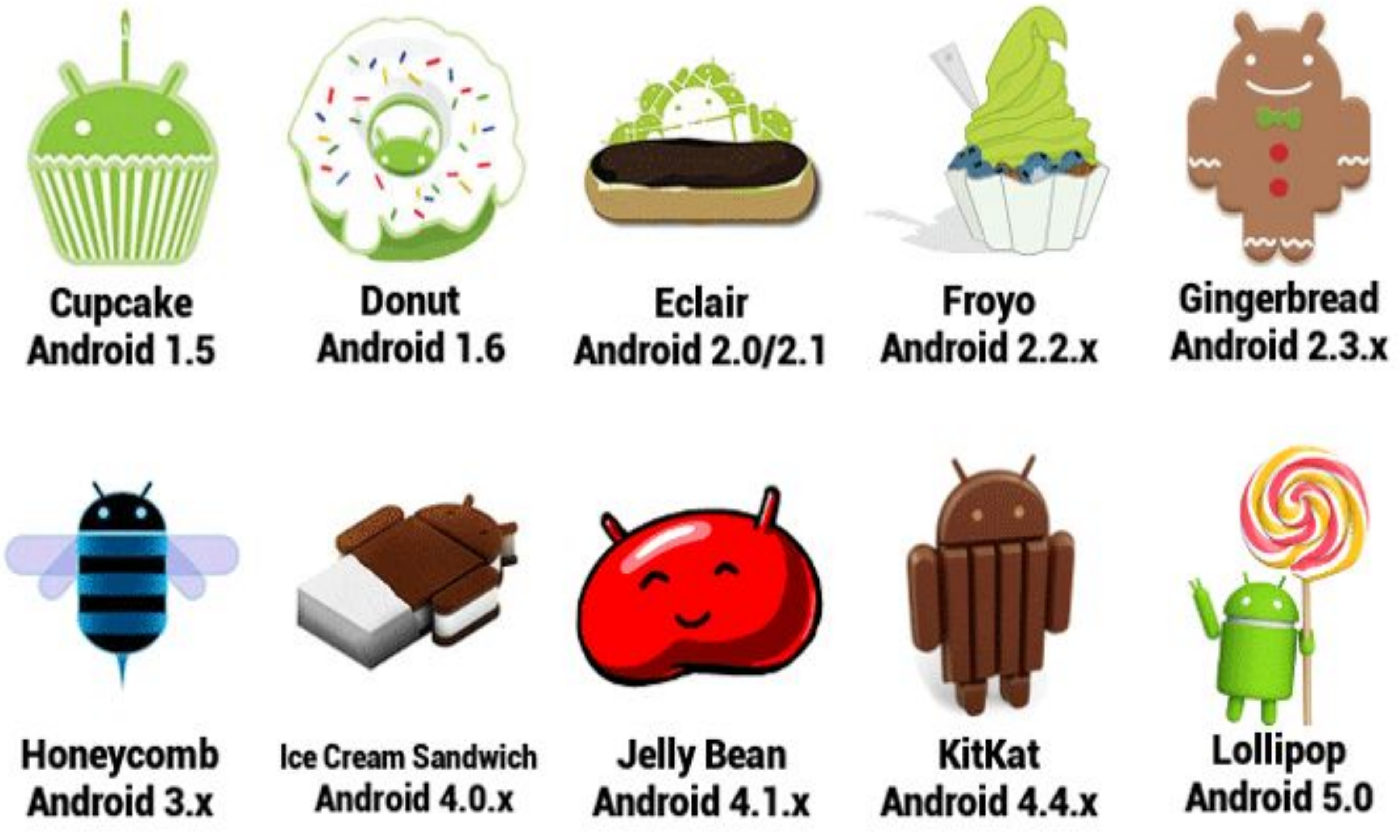
What is an SDK?

The Android SDK (software development kit) is a set of development tools used to develop applications for the Android platform .

The Android SDK includes the following:

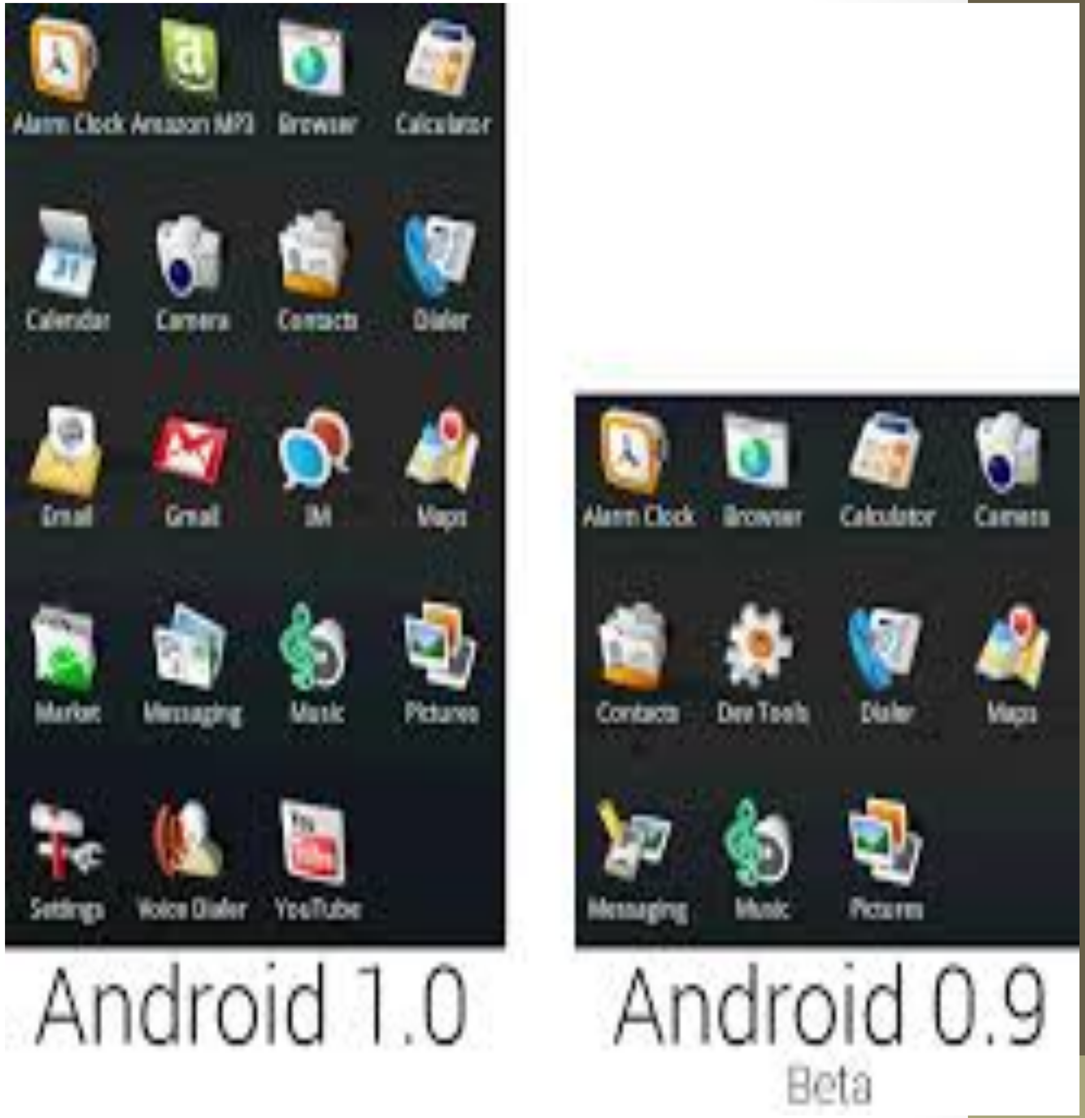
- Required libraries.
- Debugger.
- An emulator.
- Relevant documentation for the Android application program interfaces (APIs).
- Sample source code.
- Tutorials for the Android OS.

Android Versions



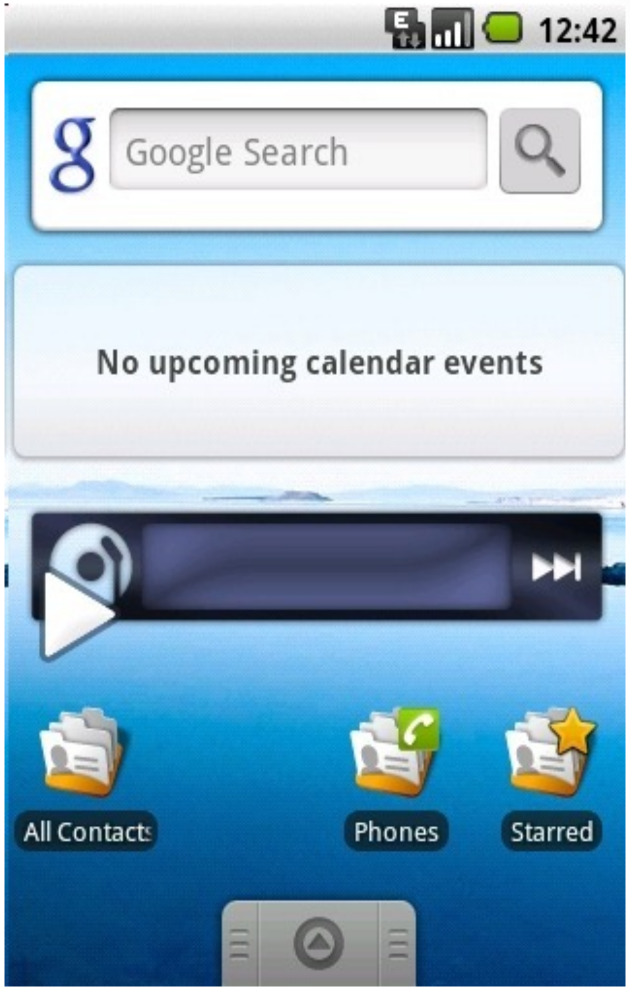
Version 1.0-APK 1

- Released on September 23,2008
- Supports Youtube player
- Wifi and Bluetooth support
- Camera maintain this version



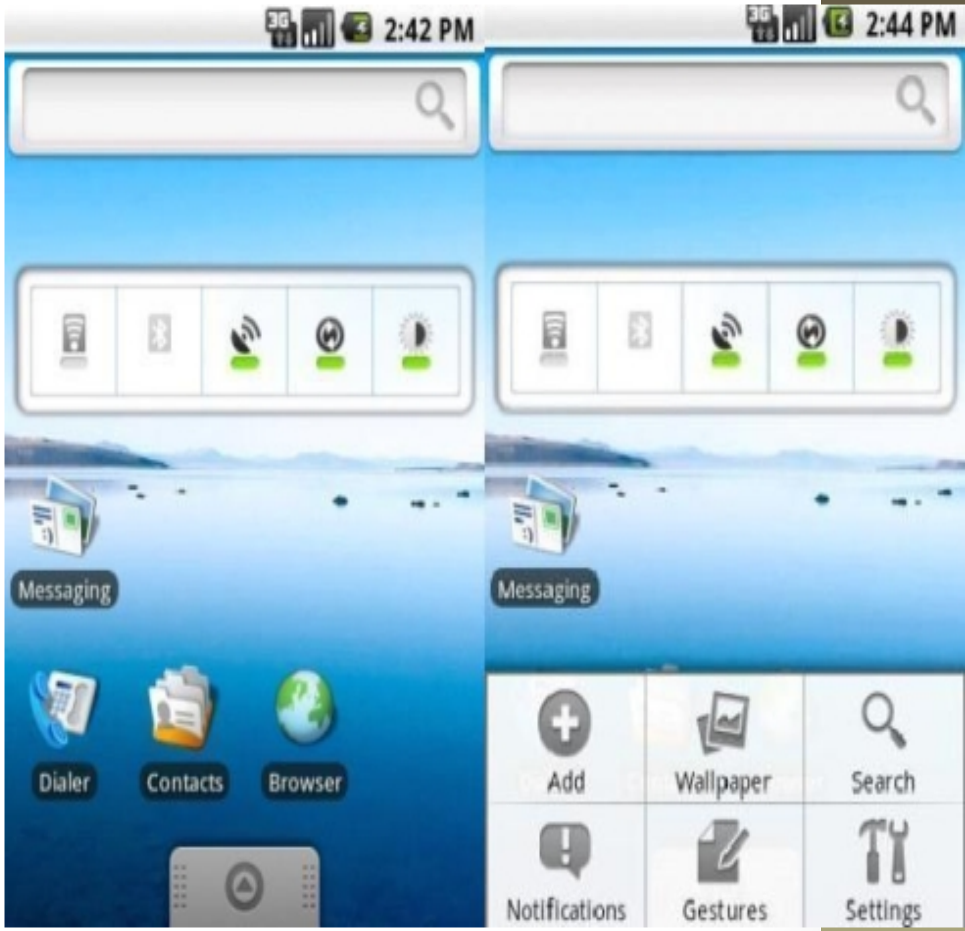
Version 1.5 (Cupcake)-APK 3

- Released on April 29,2009
- Recording and watching videos in MPEG-4 and 3GP formats
- Widgets in the home screen and animated screen transition
- Cupcake introduced numerous refinements to the Android interface, including the first on-screen keyboard.
- provided the platform's first-ever option for video recording.



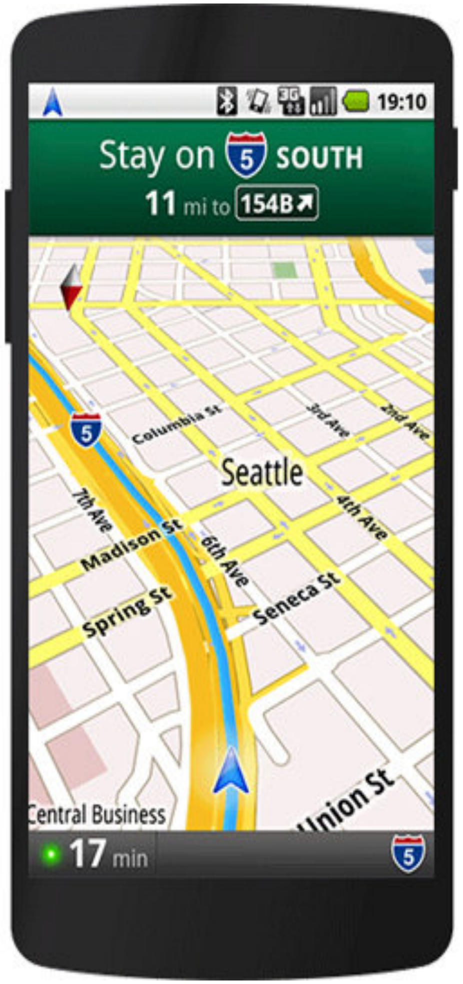
Version 1.6(Donut) based on Linux Kernel 2.6.29-APK 4

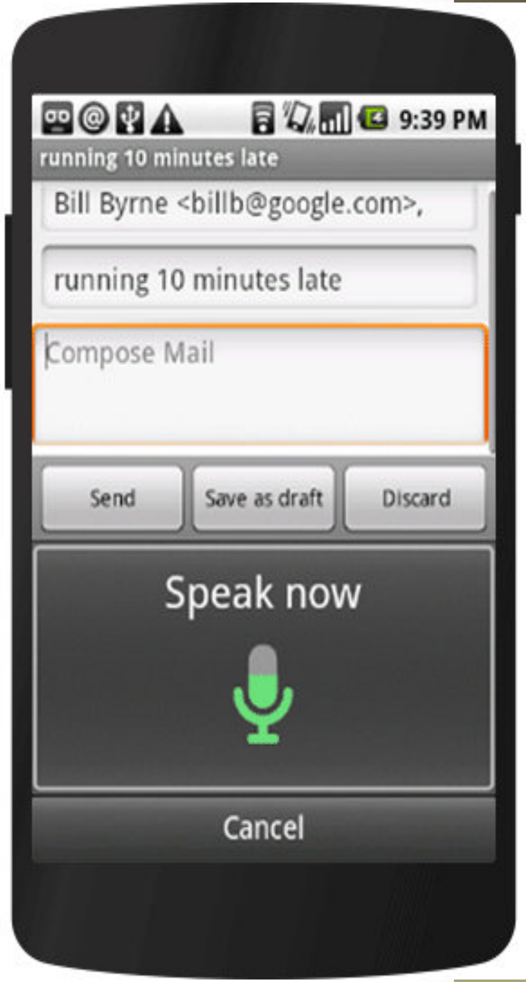
- Released SDK 1.6 on September 15 during 2009.
- Donut filled in some important holes in Android's center, including the ability for the OS to operate on a variety of different screen sizes and resolutions.
- Updated search experience.
- Integrated as camera, camcorder and gallery interface.
- Added support for CDMA networks like Verizon



Version 2.0/2.1(Éclair)based on Linux kernel 2.6.29-APK 5,6,7

- Released SDK 2.0 on October 26,2009.
- Flash support, digital zoom, scene mode, color effect etc
- Improved typing speed on virtual keyboard, smarter dictionary.
- Addition of voice-guided turn-by-turn navigation and real-time traffic info

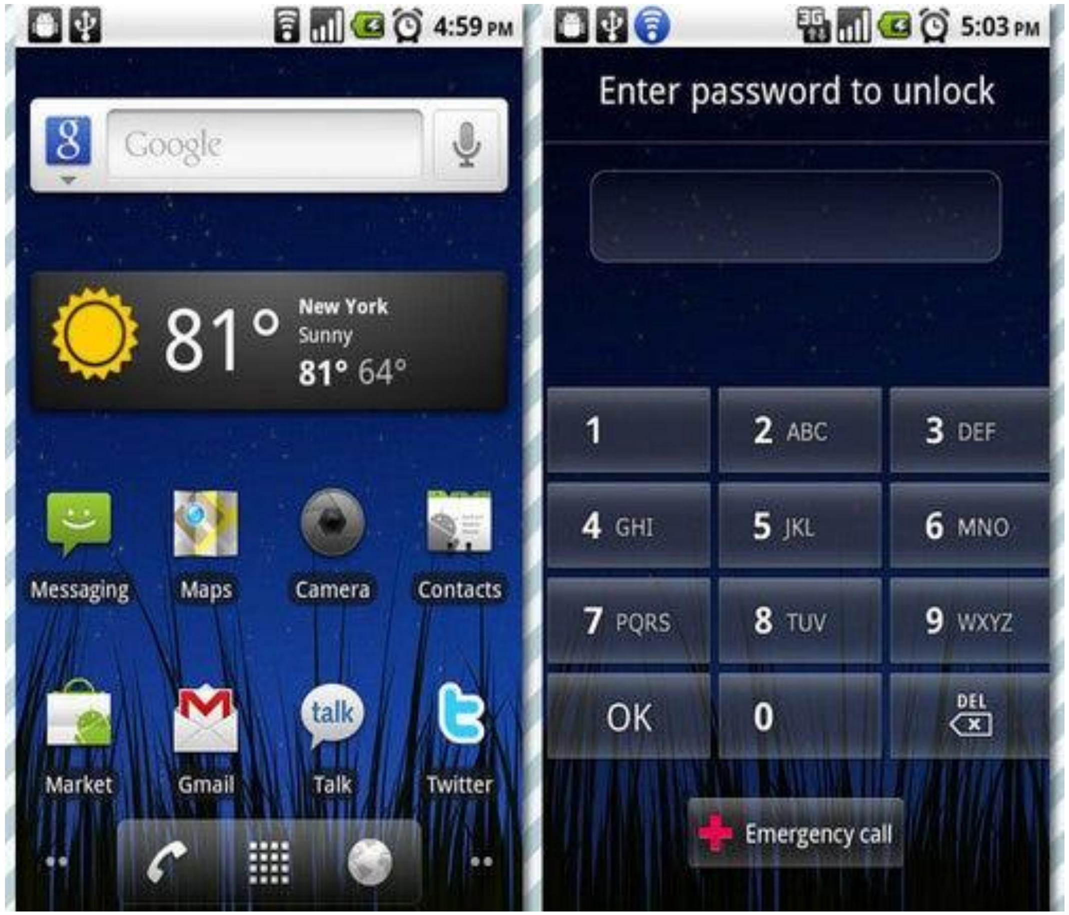




running 10 minutes late
Bill Byrne <billb@google.com>,
running 10 minutes late
Compose Mail

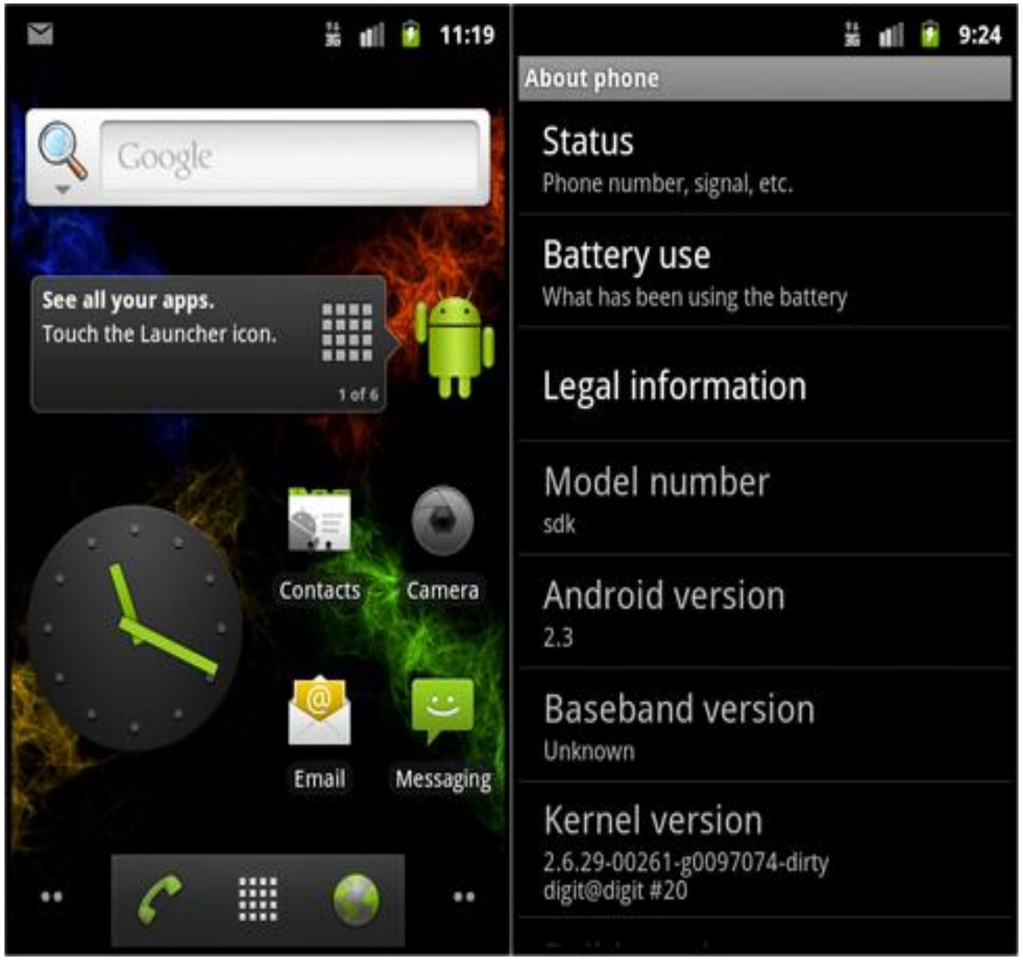
Version 2.2(Froyo) based on 2.6.32-APK 8-released on 2010 4 months after eclair

- Voice dailing,contact sharing via bluetooth,adobe flash support, USB tethering, Wifi function.
- Addition of the now-standard dock at the bottom of the home screen



Version 2.3(Gingerbread) based on 2.6.35-APK 9/10

- Released SDK 2.3 on dec 6,2010.
- Copy/paste functionality to select a word by press-hold, copy and paste.
- Black and green seeped all over the UI.



About phone

Status
Phone number, signal, etc.

Battery use
What has been using the battery

Legal information

Model number
sdk

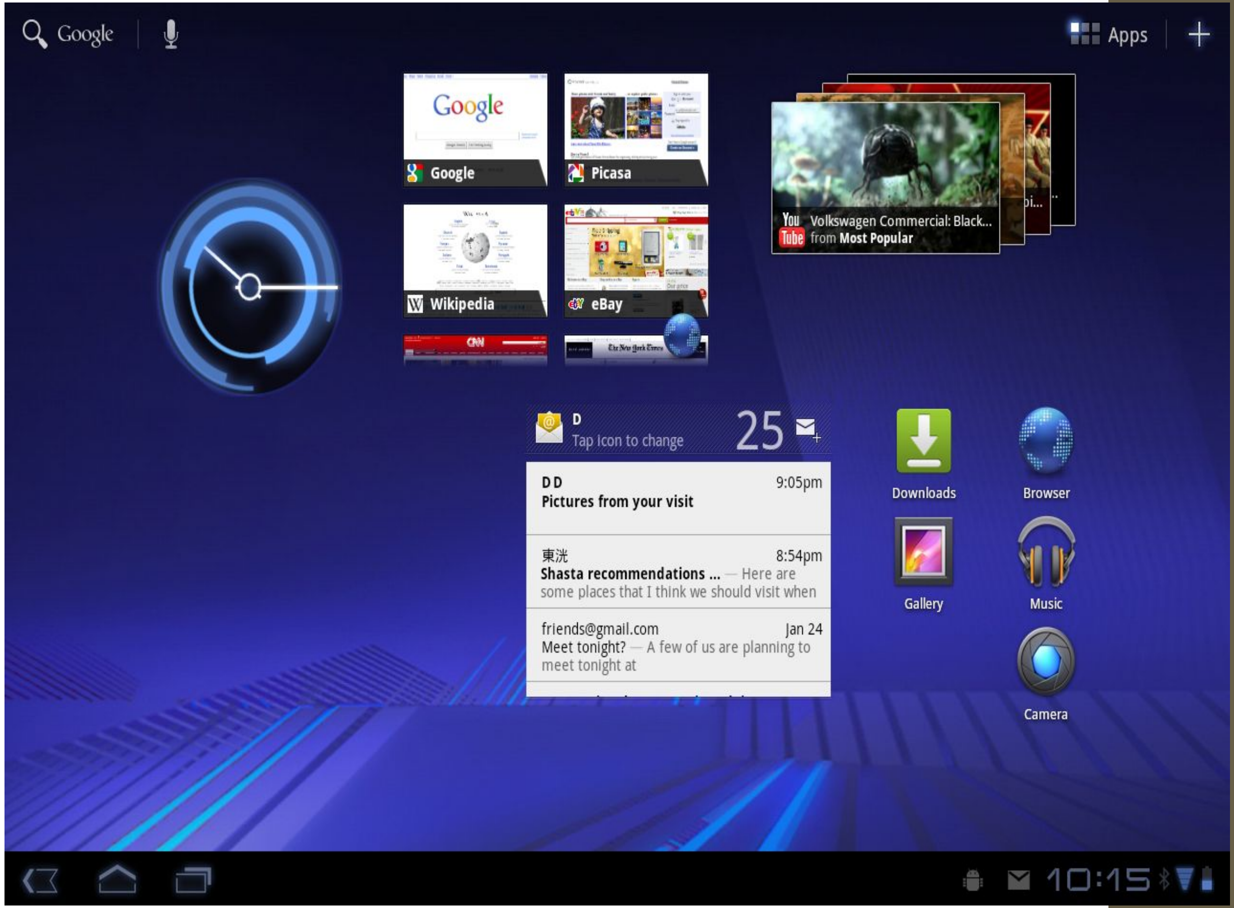
Android version
2.3

Baseband version
Unknown

Kernel version
2.6.29-00261-g0097074-dirty
digit@digit #20

Version 3.0(Honeycomb based on 2.6.36)-APK 14

- Released as sdk 3.0 on feb 22,during 2011.
- Improves multitasking.
- Supports 3d desktop with redesigned widgets.
- Supports multicore processor, hardware accelerations.
- Recently opened apps-card view
- space-like "holographic" design that traded the platform's trademark green for blue and placed an emphasis on making the most of a tablet's screen space.



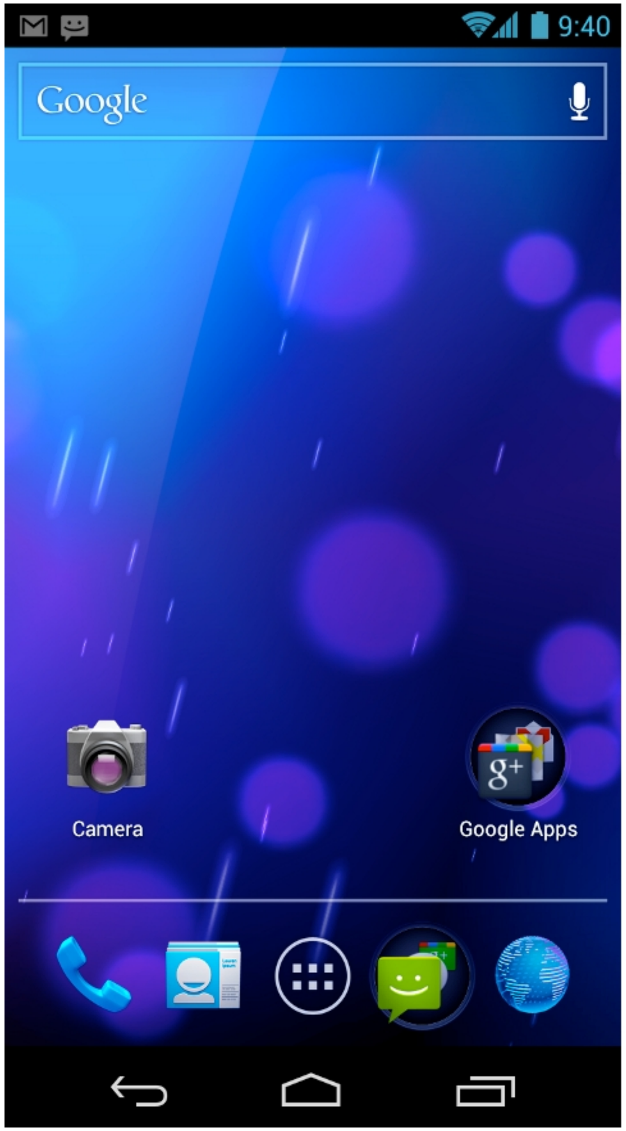
DD 9:05pm
Pictures from your visit

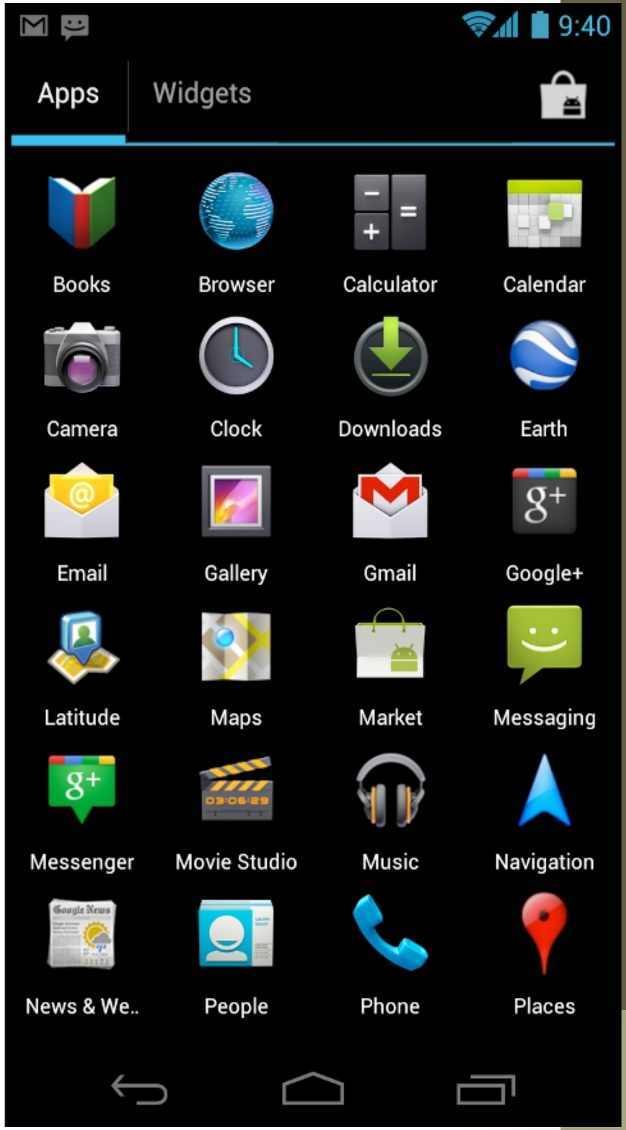
東洸 8:54pm
Shasta recommendations ... — Here are some places that I think we should visit when

friends@gmail.com Jan 24
Meet tonight? — A few of us are planning to meet tonight at

Version 4.0(Icrecream Sandwich) based on 3.0.1

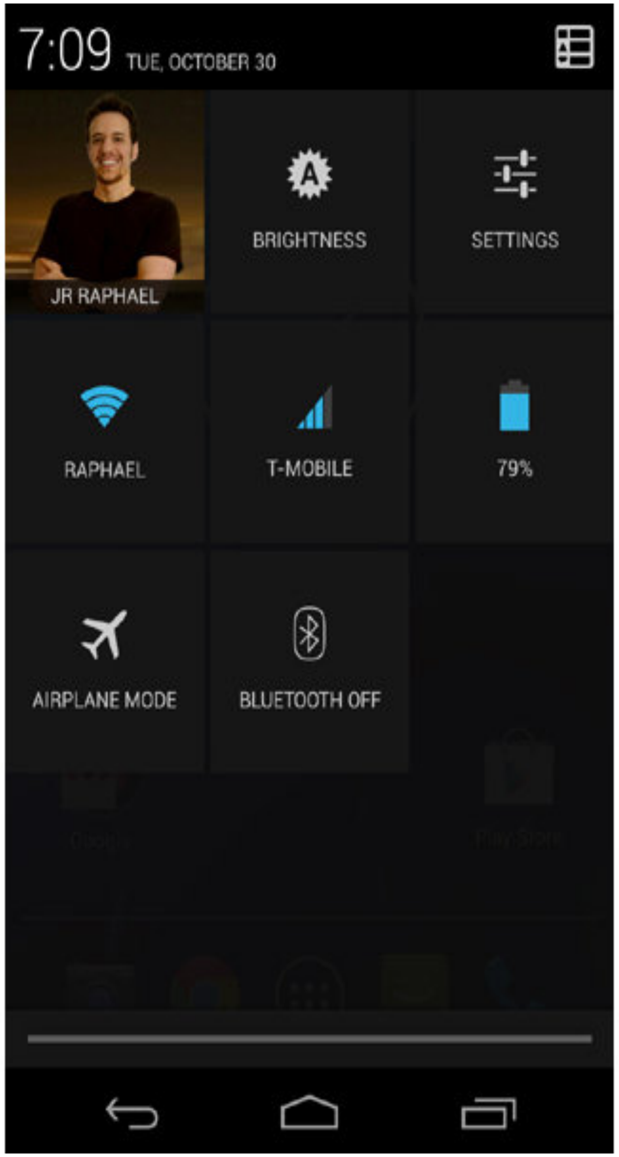
- Released SDK 4.0.1 on October 19,2011.
- Official entry into modern design.
- Eliminated holographic design,but maintained blue theme
- Include 1080 P recording and a customizable launcher

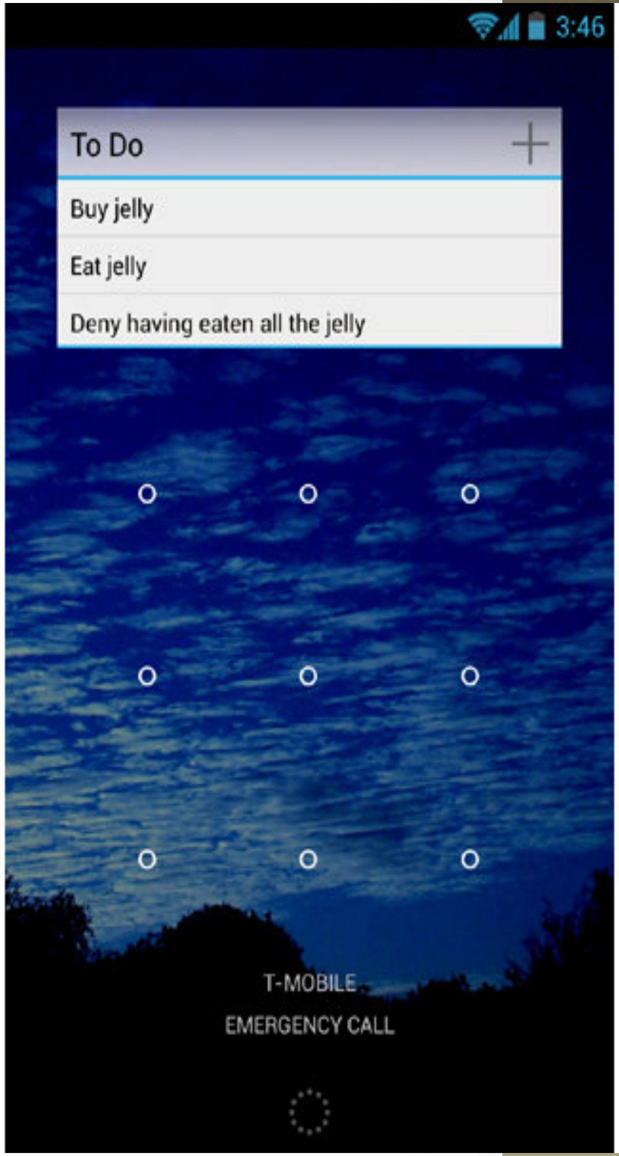




Version 4.1(Jelly Bean)

- Released SDK 4.1 on June 27,2012.
- Improved voice search and multichannel audio.
- Automatically resizable app widgets.
- Jelly Bean ushered in a heavily hyped system for placing widgets on your lock screen.





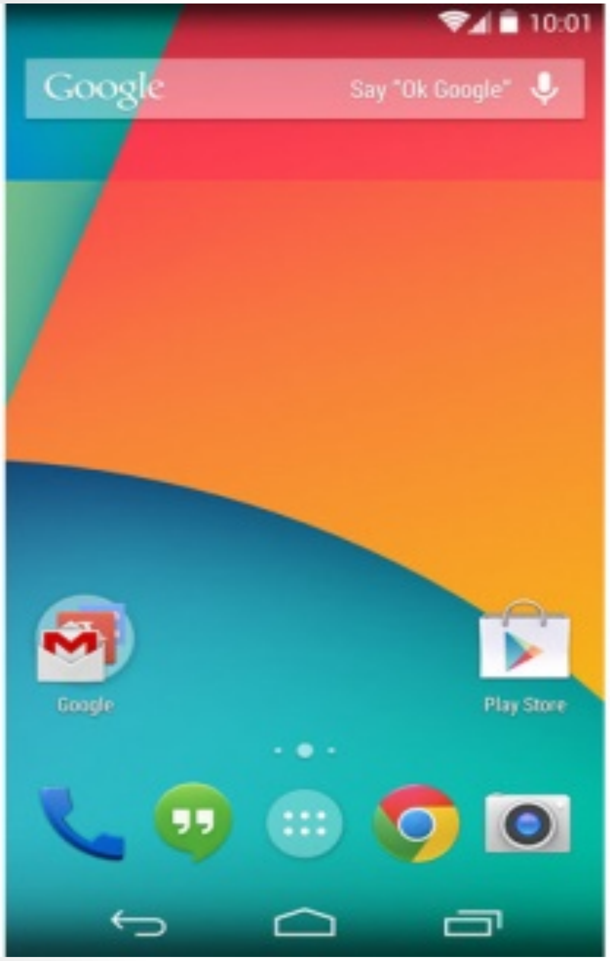
To Do

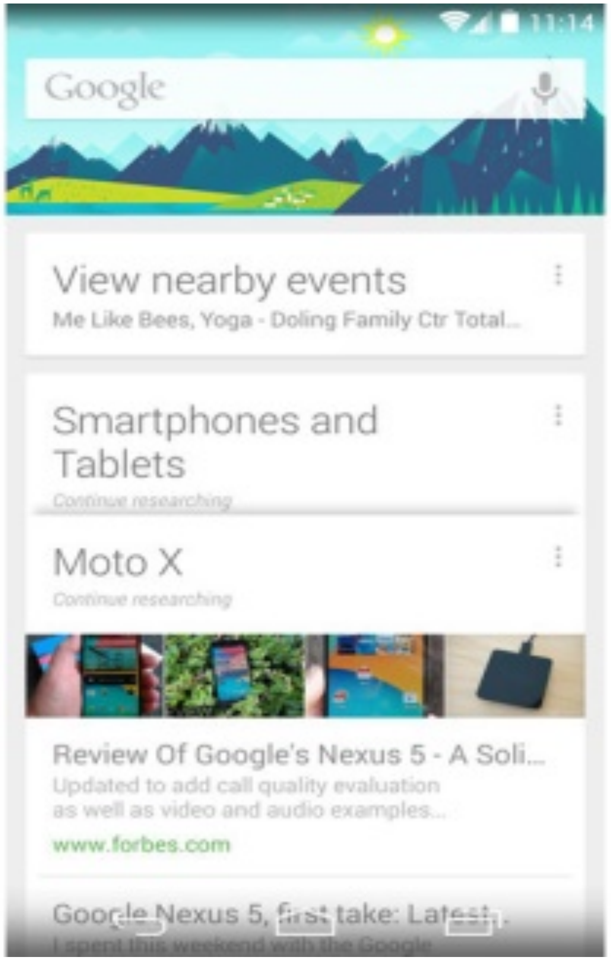
Buy jelly

Eat jelly

Deny having eaten all the jelly

Version 4.4 KITKAT





View nearby events
Me Like Bees, Yoga - Doling Family Ctr Total...

Smartphones and Tablets
Continue researching

Moto X
Continue researching

Review Of Google's Nexus 5 - A Soli...
Updated to add call quality evaluation as well as video and audio examples...
www.forbes.com

Google Nexus 5, first take: Latest...
I spent this weekend with the Google

- Lighter backgrounds and more neutral highlights took their places, with a transparent status bar and white icons giving the OS a more contemporary appearance.
- Android 4.4 also saw the first version of "OK, Google" support.

Other versions

- ❖ Android 5.0 and 5.1 Lollipop
- ❖ Android version 6.0: Marshmallow
- ❖ Android versions 7.0 and 7.1: Nougat
- ❖ Android version 8.0 and 8.1: Oreo
- ❖ Android version 9: Pie
- ❖ Android version 10
- ❖ Android version 11

Android Activity

- 2001-search service for wireless device
- 2005-Acquire Android

(Acquire Skia(2d graphics for mobile device)
Acquire RegWireless(Browser and email for mobile device)
Move engineers from PlamSource))

- 2007(nov 5)-Android announced
- 2007(nov 12)-Android SDK is released by OHA.
- 2007(Dec 14)-Bug fix SDK released
- 2008(Jan 3)-Android Developer Challenge I starts accepting submissions
- 2008(feb 13)-m5-rc15 SDK released
- 2008(April 14)-1788 total submissions for challenge
- 2008(may)-top 50 applications announced in challenge I
- 2008(Nov)-Android phone(G1 phone by HTC/T mobile)
- 2008(nov)-full source open
- 2009(april)-HTC Magic
- 2009(july)-HTC Hero,Samsung i7500,Android Notebook
- 2009(Aug)-Android Developer Challenge

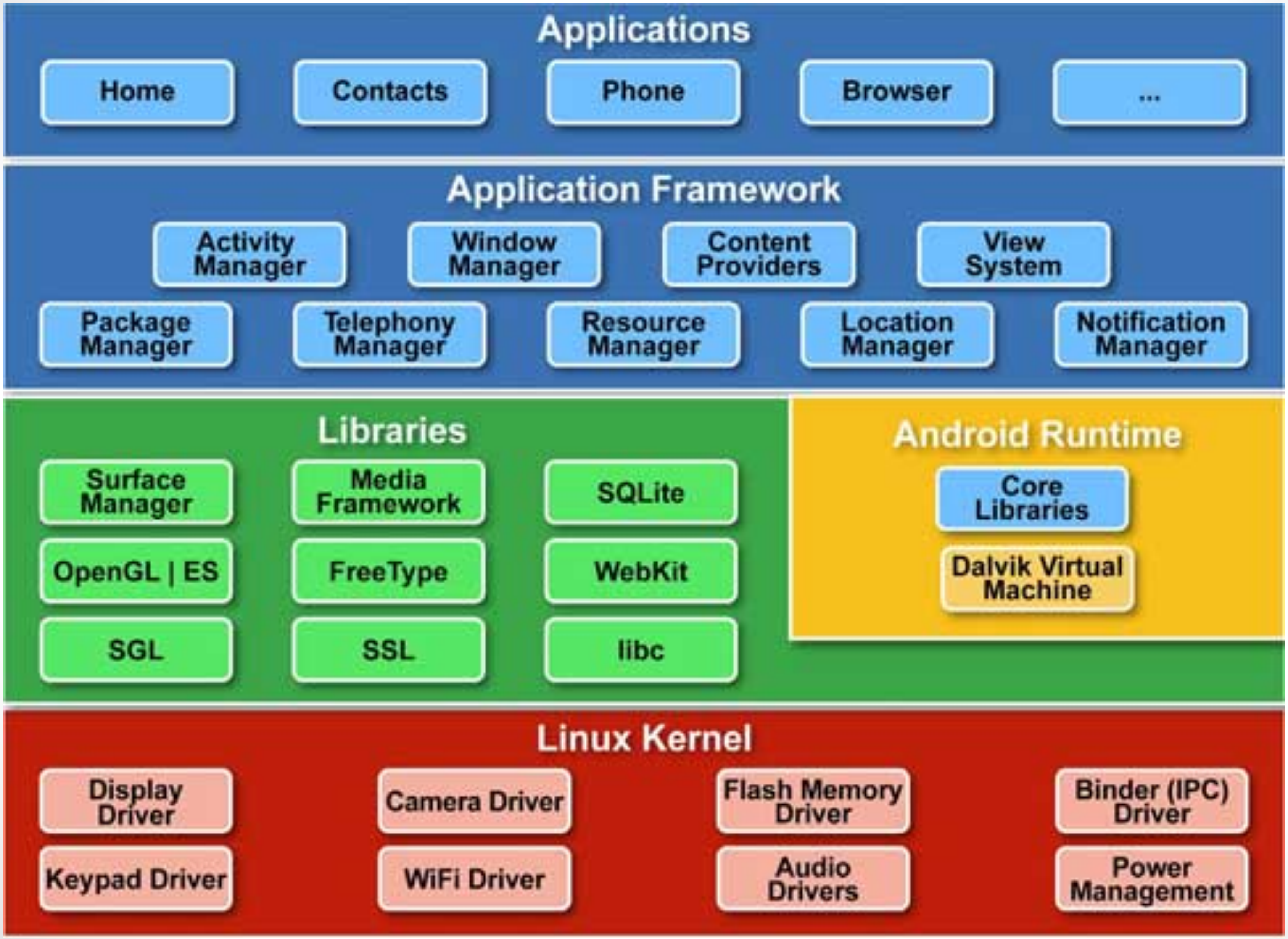
FEATURES OF ANDROID

- **OPEN SOURCE**
- **STORAGE**-SQLLite
- **MEDIA SUPPORT**(webM,H.263,H.264,AAC(IN 3GP OR MP4 CONTAINER),WAV,JPEG,GIF,BMP,FLAC etc
- **STREAMING MEDIA SUPPORT**-RTP/RTSP streaming,HTML progressive download(HTML 5 video tag),Adobe flash streaming(RTMP),HTTP dynamic streaming supported by flash plugin
- **MULTITOUCH**
- **WEB BROWSER-** based on open source WebKit layout engine,attached with Chrome V8 Javascript engine.
- **VIDEO CALLING-** Skype 2.1 offers video calling in android 2.3,including front camera support. Google talk video calling available from Android 2.3.4.
- **MULTITASKING**
- **ACCESSIBILITY-** text to speech option built in
- **VOICE BASED FEATURES-** voice activities for calling, texting, navigation supported from android 2.2.
- **EXTERNAL STORAGE-** most Android devices include microSd slots. USB ‘A’ receptacle also enabled for allowing the use of high storage capacity storage media like USBHDDs and USB Flash Drives

ANDROID ARCHITECTURE

Mostly Android has 4 layers:

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


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

1.APPLICATIONS

- ❖ All applications in this layer are written using Java.
- ❖ The applications that an android device provide includes
  - Email client
  - SMS programs
  - Maps
  - Browser
  - Calender
  - Contacts
  - Games etc

2.APPLICATION FRAMEWORKS

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 −

1.**Activity Manager** − Controls all aspects of the application lifecycle and provides a common backstack for navigation.

2.**Window manager-** organizes the screen layout and locates the drawing surface and performs other window related jobs

3.**View Manager** − An extensible set of views used to create application user interfaces.

4.**Content Providers** − Allows applications to publish and share data with other applications.

5.**Notifications Manager** − Allows applications to display alerts and notifications to the user.

6.Package Manager- manages the other packages in the system
7.Telephony Manager- to handle the receiver call or voice calls
8.Resource Manager − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
9.Location Manager-helps to locate the mobile device.example-GPS,Ex-map
10.XMPP Service Manager-Services like music,ringtone,browser etc managed here.

3.Linux kernel

- The entire Android OS is built on the top of Linux 2.6 with architectural changes made by Google.
- Offers better security and networking
- Android run time gives .dex file which goes to the linux kernel and calls the suitable drivers so that it can communicate with the drivers.
- Following are the drivers:
  - Display drivers
  - Camera Drivers
  - Bluetooth driver
  - Flash memory driver
  - Usb Driver
  - Power management driver
  - Keypad driver
  - Wifi driver etc

4.Libraries

- Written in C and C++ libraries
- This layer is communicated through Java Native Interface package.
- 9 major components

❑ **Media** library provides support to play and record an audio and video formats.

❑ **Surface manager** responsible for managing access to the display subsystem.

❑ **SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.

❑ **SQLite** provides database support

❑ **FreeType** provides font support.

❑ **Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.

❑ **SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.

5.ANDROID RUN TIME

- Android Runtime environment is **one of the most important part of Android**. It contains components like **core libraries and the Dalvik virtual machine(DVM**).
- Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

5.1 CORE LIBRARIES

Different from java SE libraries, provides functionalities like

- data structure
- file access
- network access
- Utilities
- graphics etc

5.2 Dalvik Virtual Machine (DVM)

Developed by Dan Bornstein of Google.

- Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently.
- Runs .dex files. These files are built from .class files at compile time and provide higher efficiency in low resource environments.
- It depends on the layer Linux kernel for threading and low-level memory management.

CONFIGURATION OF ANDROID ENVIRONMENT

OPERATING SYSTEM

To develop an android program, the necessary supporting operating systems used in a computer can be as follows:

a.Windows XP(32 bit),Vista(32 or 64 bit) or Windows 7(32 bit or 64 bit) or later.
2GB RAM minimum,4GB ram recommended

b.Mac OS X(Intel) 10.8.5 or higher(x86 only)
2GB RAM minimum,4GB ram recommended

c.Linux(i386)-GNU C Library 2.7 or later required
On Ubuntu,version 8.4 or later required
2GB RAM minimum,4GB ram recommended

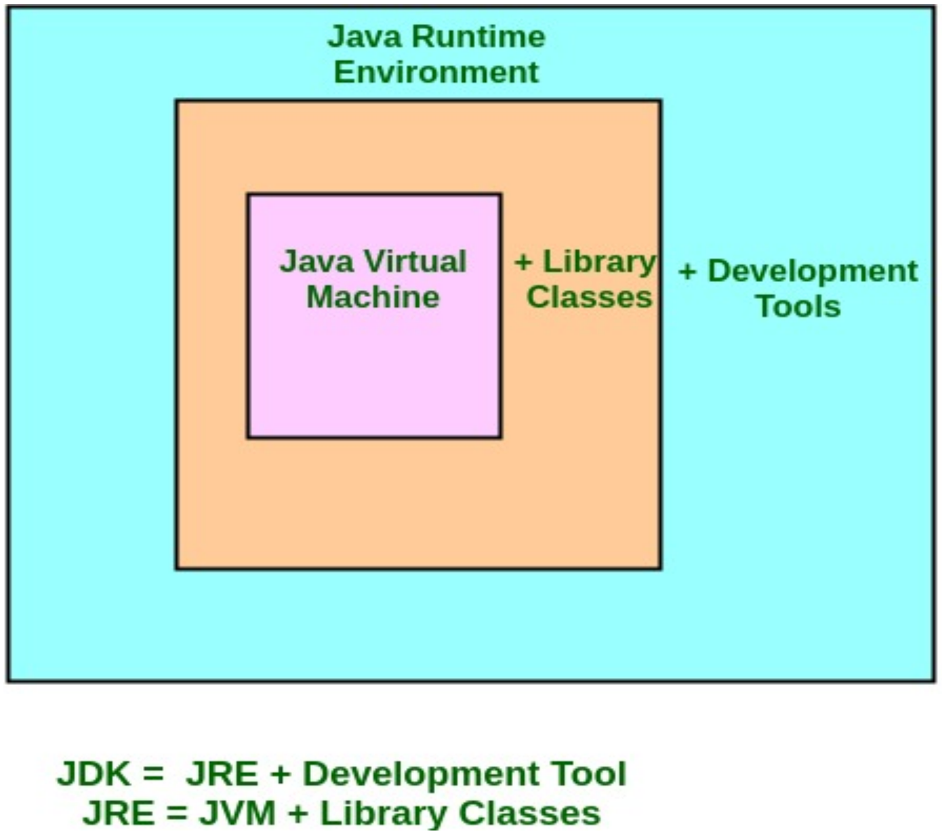
Java Development Kit(JDK)

As android programs are developed in Java programming language we have to install **JDK** which is a free software and which includes **Java RunTime Environment(JRE)**

JDK

The **Java Development Kit (JDK)** is a software development environment that offers a collection of tools and libraries necessary for developing Java applications. You need the JDK to convert your source code into a format that the Java Runtime Environment (JRE) can execute.

- The JDK includes the Java Runtime Environment (JRE), an interpreter (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc), and some other development tools.
- The Java Runtime Environment itself consists of the Java Virtual Machine (JVM), supporting files, and core classes.



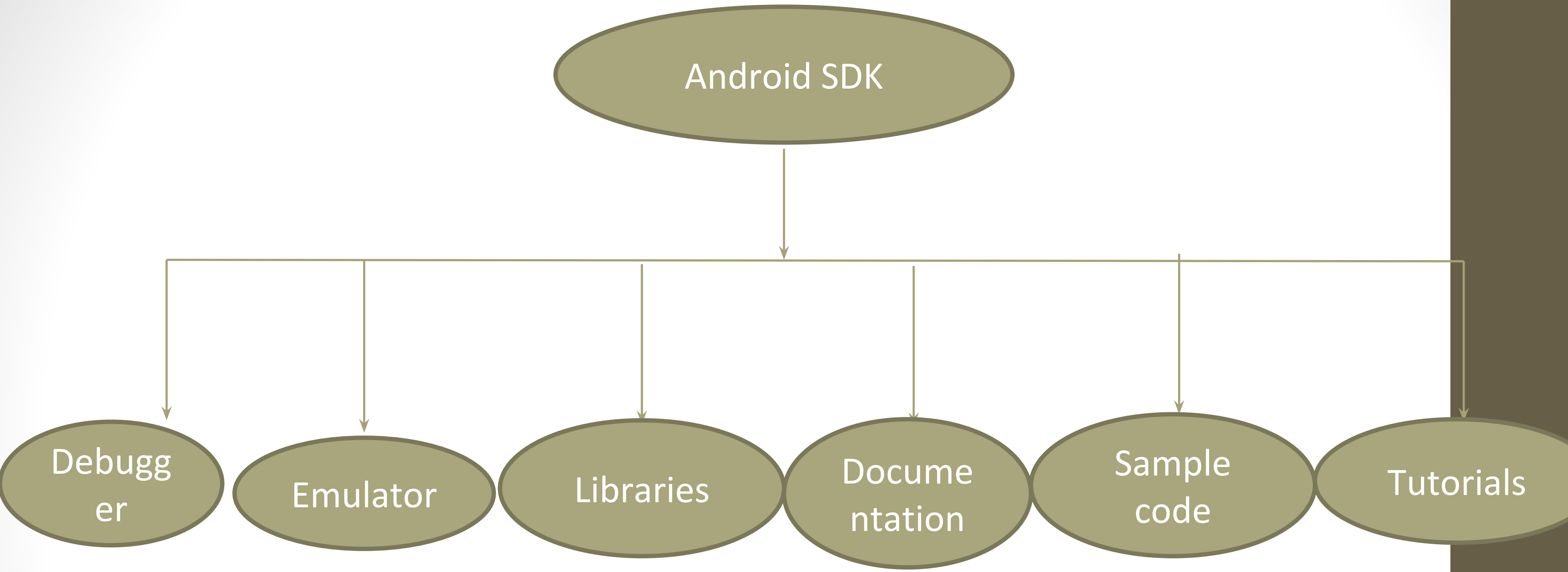
Android SDK

- **Android SDK** is a software development kit developed by Google for the **Android** platform.

- The Android SDK (Software Development Kit) is a set of development tools that are used to develop applications for the Android platform.

- Whether you create an application using *Java*, *Kotlin* or *C#*, you need the SDK to get it to run on any Android device. You can also use an emulator in order to test the applications that you have built.

- Nowadays, the Android SDK also comes bundled with Android Studio, the integrated development environment where the work gets done and many of the tools are now best accessed or managed.



Can be downloaded from http://developer.android.com/sdk/index.html using android manager which is free.

- SDK is made up of two parts-
    - tools
    - packages
- When you first install you get the basic tools(executables and supporting files) to develop apps
- Packages are records specific to a particular version of Android(platform) or a particular add-on to a platform.

SDK FEATURES

- ❖ No licensing,development fees,distribution
- ❖ Wifi hardware access
- ❖ Comprehensive API s design for location based services such as GPS
- ❖ Open source webkit based browser
- ❖ Media libraries
- ❖ IPC message passing
- ❖ Shared data stores

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
- IntelliJ
- Eclipse IDE

versions- Eclipse Galileo(3.5 or higher)
Eclipse Helios(3.6.2 or higher)
Eclipse Indigo

ANDROID DEVELOPMENT TOOLS

- Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications.
- ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, **create** an application UI, **add packages** based on the Android Framework API, **debug your applications** using the Android SDK tools, and even **export** signed (or unsigned) .apk files in order to distribute your application.
- ADT provides custom XML editors and debug output pane
- ADT BUNDLE provides
    - ❖ Eclipse+ ADT plugin
    - ❖ Android SDK Tools
    - ❖ Android Platform tools
    - ❖ The most recent Android platform
    - ❖ Newest Android system image for the emulator

EMULATOR

- The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.

- The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.

- Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.

- The emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.

Emulator supports many hardware features found on mobile devices like

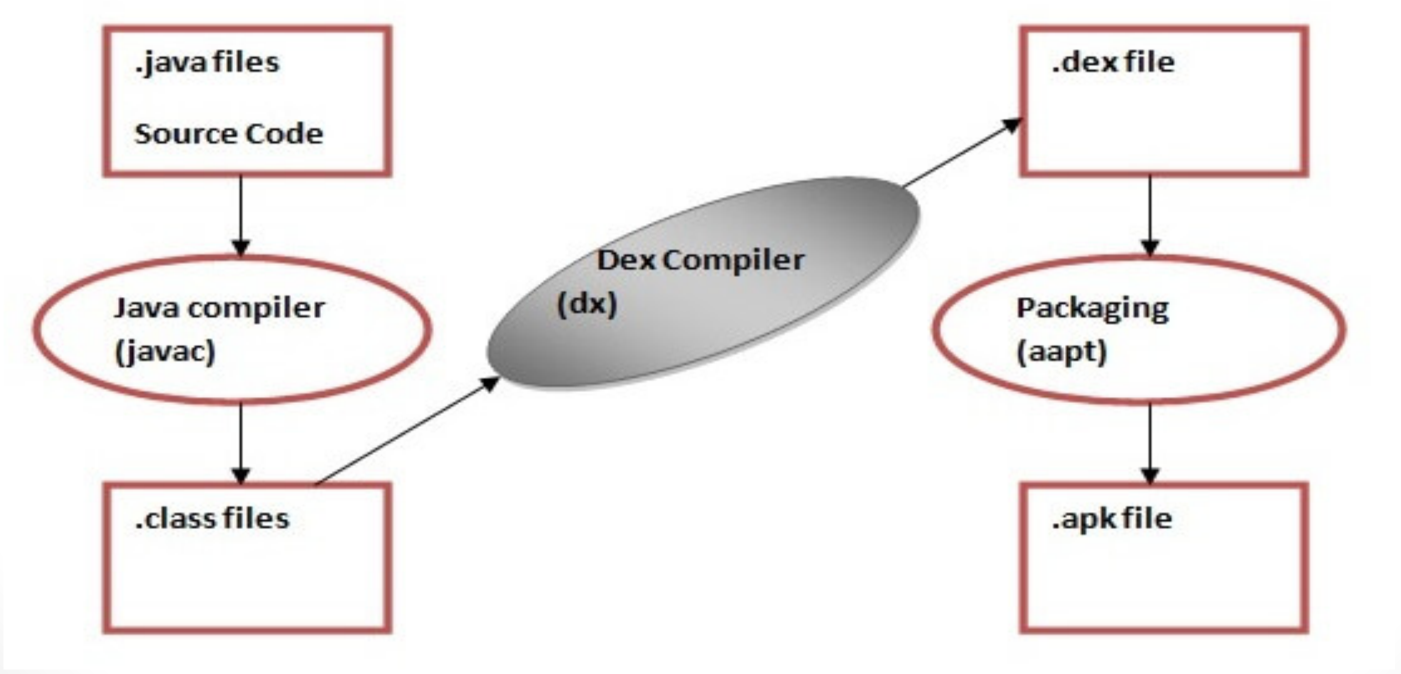
- An ARMV5 CPU and corresponding MMU
- A 16bit LCD display
- One or more keyboards(A querty based keyboard and associated dialpad)
- A GSM modem including a simulated SIM card
- A camera using a webcam connected to our computer
- Sensors like accelerometer

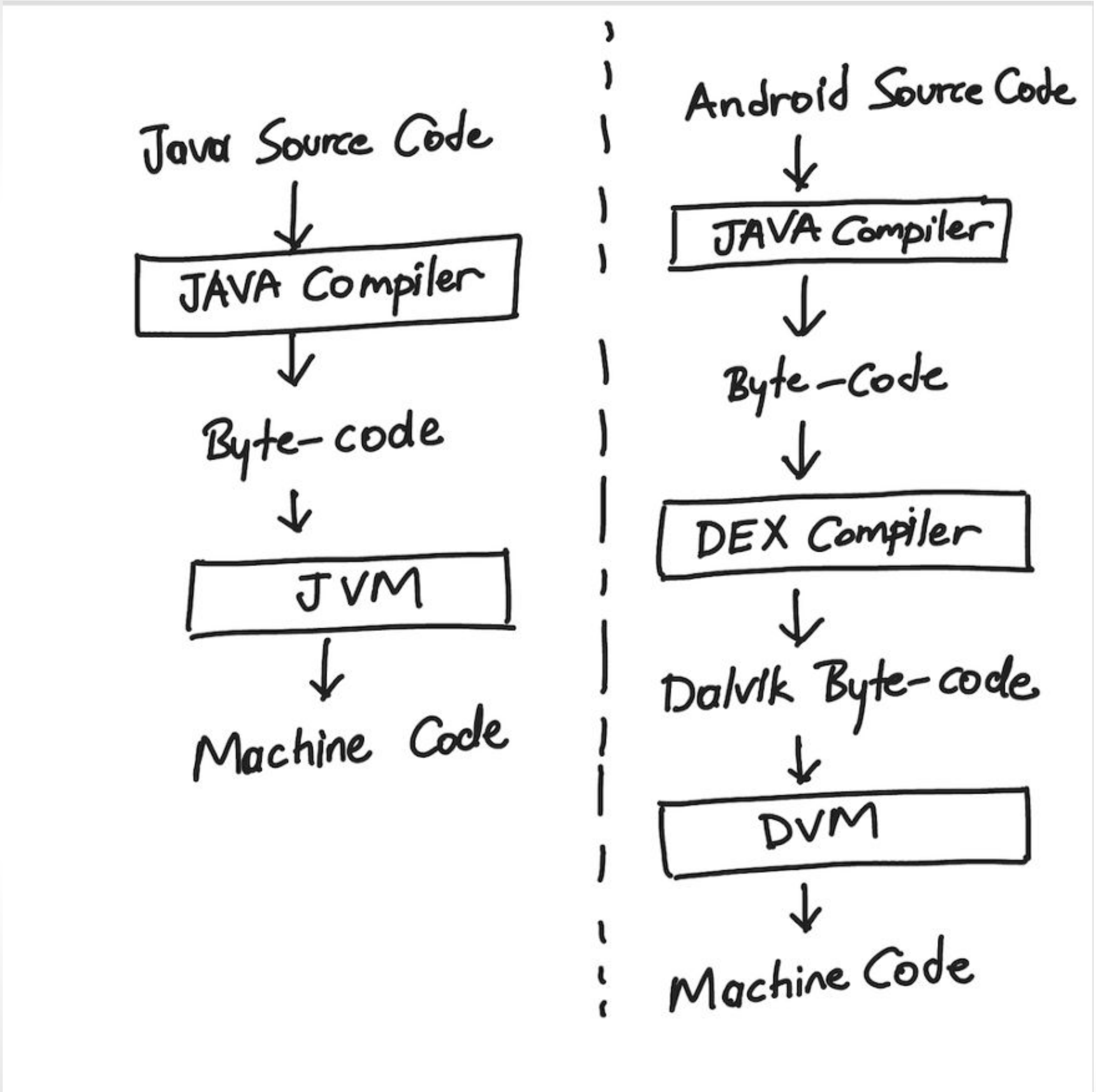
Android Virtual Device

- An Android Virtual Device (AVD) is an emulator configuration that allows developers to test the application by simulating the real device capabilities.
- We can configure the AVD by specifying the hardware and software options. AVD manager enables an easy way of creating and managing the AVD with its graphical interface.
- An AVD consists of
  - ❖ **A hardware profile** -defines hardware features of the virtual device
  - ❖ **A mapping to a system image** -you can define what version of Android platform will run on the virtual machine.
  - ❖ You can specify the **emulator skin you want to use with the AVD,** which lets you control the screen dimensions,look and so on.You can also specify the emulated SD card to store with the AVD.
  - ❖ **A dedicated storage area on your development machine** -includes the device user data(installed applications,settings)and emulated sdcard stored here.

Dalvik Virtual Machine | DVM

- modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.
- 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.
- The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.




Java Source Code
JAVA Compiler
Byte-code
JVM
Machine Code
Android Source Code
JAVA Compiler
Byte-Code
DEX Compiler
Dalvik Byte-code
DVM
Machine Code

- Dalvik uses Dalvic byte code which is different from Java byte code. Java class files cannot directly run on Android, they need to be converted into Dalvic byte code format by the Dex compiler.

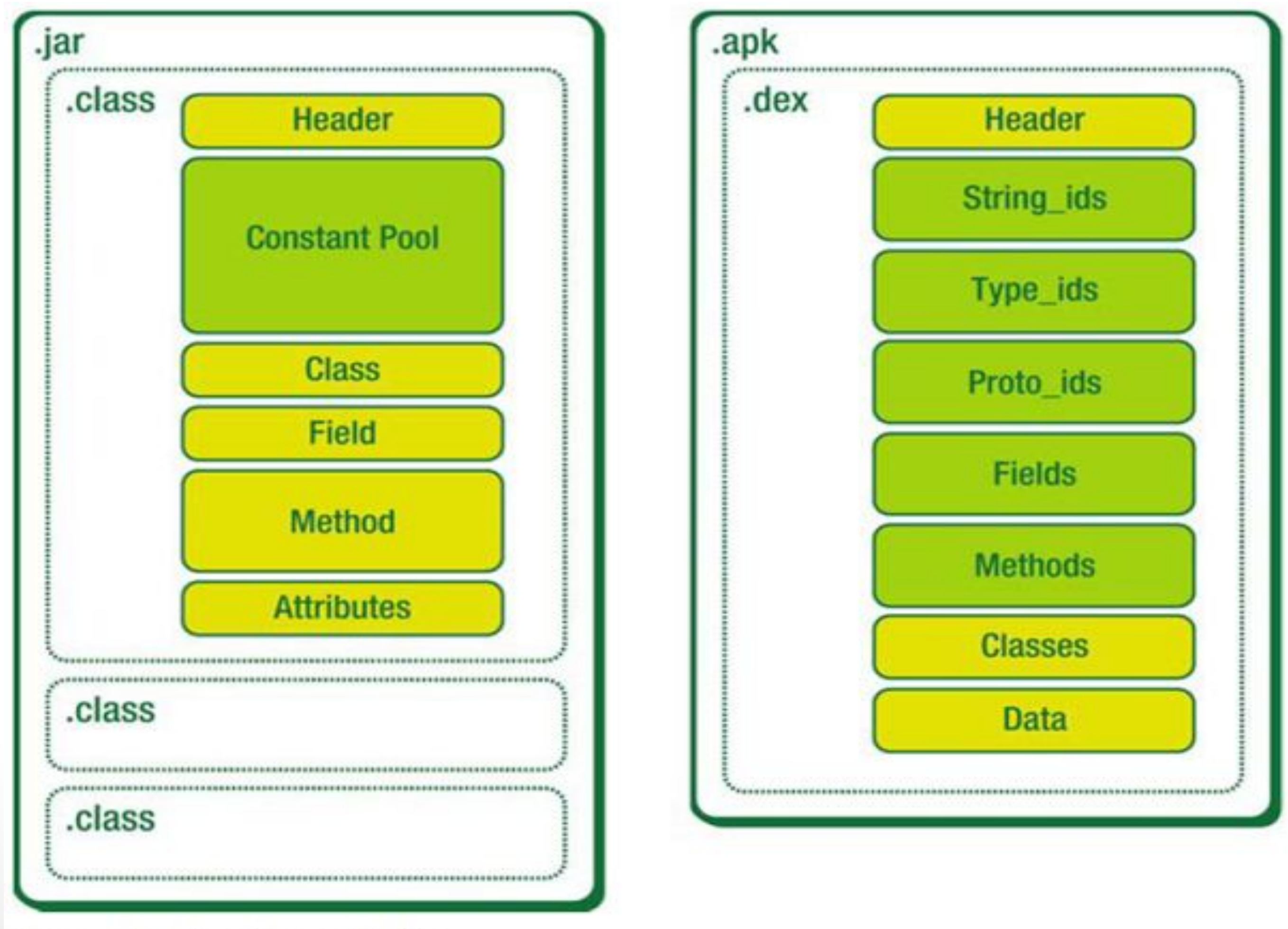


Figure 3-2. *Class file vs DEX file*

- **Header**-Dex files start with a simple header with some checksums and offsets to other structures.
- **String table**-stores the length and offsets for every string in the dex file
- **Class List**-A list of classes contained in the dex file
- **Field table** -A table of fields of all classes defined in the dex file
- **Method Table**-A table of methods of all classes defined in the dex file
- **Class Defenition table**-A table of class defenitions for all classes either defined in this dex file or has a method or field accessed by code in this dex file.
- **Field list** -stores data for preinitialized fields in a class.
- **Method list-** A list of methods for a particular class
- **Code header**- contains information about code that implements a method.
- **Local variable list**- A list of local variables for a particular method

DVM (Dalvik Virtual Machine)	JVM (Java Virtual Machine)
It is Register based which is designed to run on low memory.	It is Stack based.
DVM uses its own byte code and runs ".Dex" file. From Android 2.2 SDK Dalvik has got a **Just in Time compiler**	JVM uses java byte code and runs ".class" file having JIT (Just In Time).
DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance.	Single instance of JVM is shared with multiple applications.
DVM supports Android operating system only. Runs on less memory	JVM supports multiple operating systems. Runs on more memory
There is constant pool for every application. Here the executable is APK.	It has constant pool for every class. Here the executable is JAR.

INSTALLATION OF ANDROID STUDIO

ANDROID STUDIO

- The official Integrated Development Environment (IDE) for developing Android Apps is Android Studio, which Google supports. Java was replaced by Kotlin on May 7, 2019, as a preferred language for developing Android Apps. But still, Java is being used for developing Android Apps.

- Android Studio has the following features.
  - ❖ Gradle-based build support.(advance build toolkit to automate and manage the build process)
  - ❖ Fast and feature rich emulator
  - ❖ C++ and NDK support(**NDK** is a toolset that lets you implement, compile and debug your app in native code. For certain types of apps, this can help you reuse code libraries written in C and C++.)

- Built-in support for Google Cloud Platform enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine.

- Lint tools(**Linting** is the automated checking of your source code for programmatic and stylistic errors) to catch performance, usefulness, version compatibility, and other problems

- Unified environment where you can develop for all android devices

- Plug-in architecture for extending android studio via plugins.

System Requirements

Before downloading and installing Android Studio, the following requirements are essential.

Operating System Version - Microsoft Windows 7/8/10 (32-bit or 64-bit).
Random Access Memory (RAM) - Minimum 4 GB RAM and 8 GB RAM recommended.
Free Disk Space - Minimum 2 GB and 4 GB recommended.
Minimum Required JDK Version - Java Development Kit (JDK) 8.
Minimum Screen Resolution - 1280 * 800.resolution

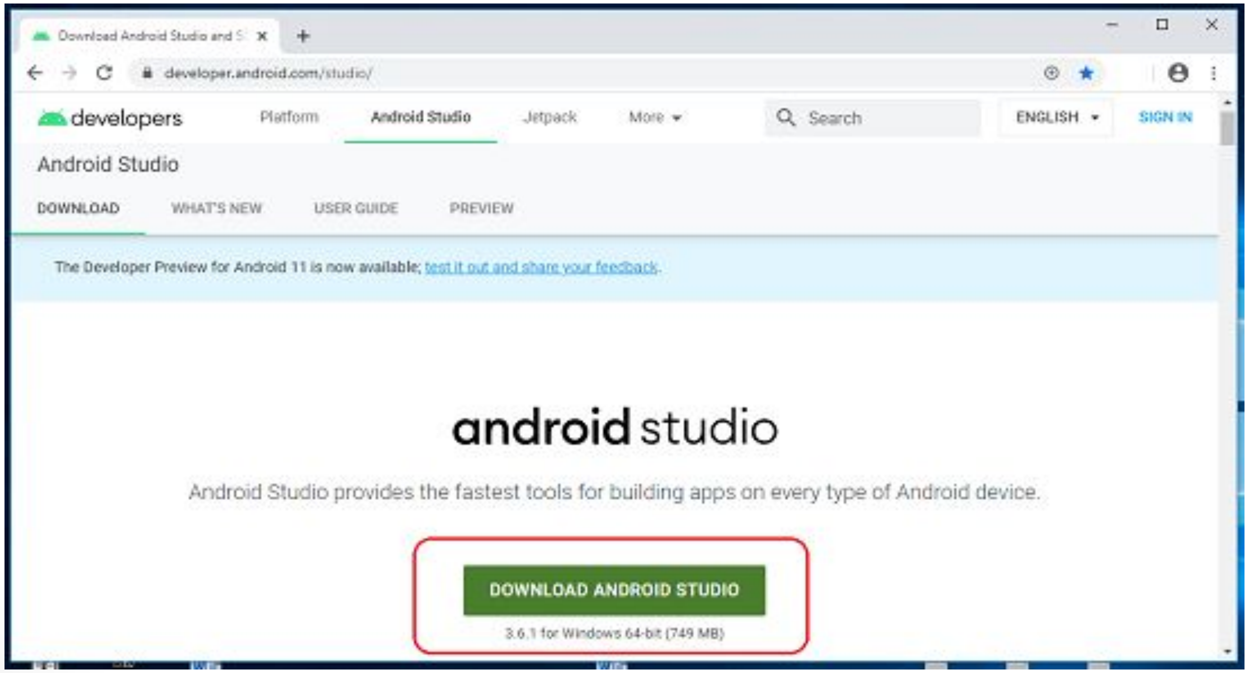
Download and Install Android Studio

Step 1

To download the Android Studio, visit the official Android Studio website in your web browser.

Step 2

Click on the "Download Android Studio" option.



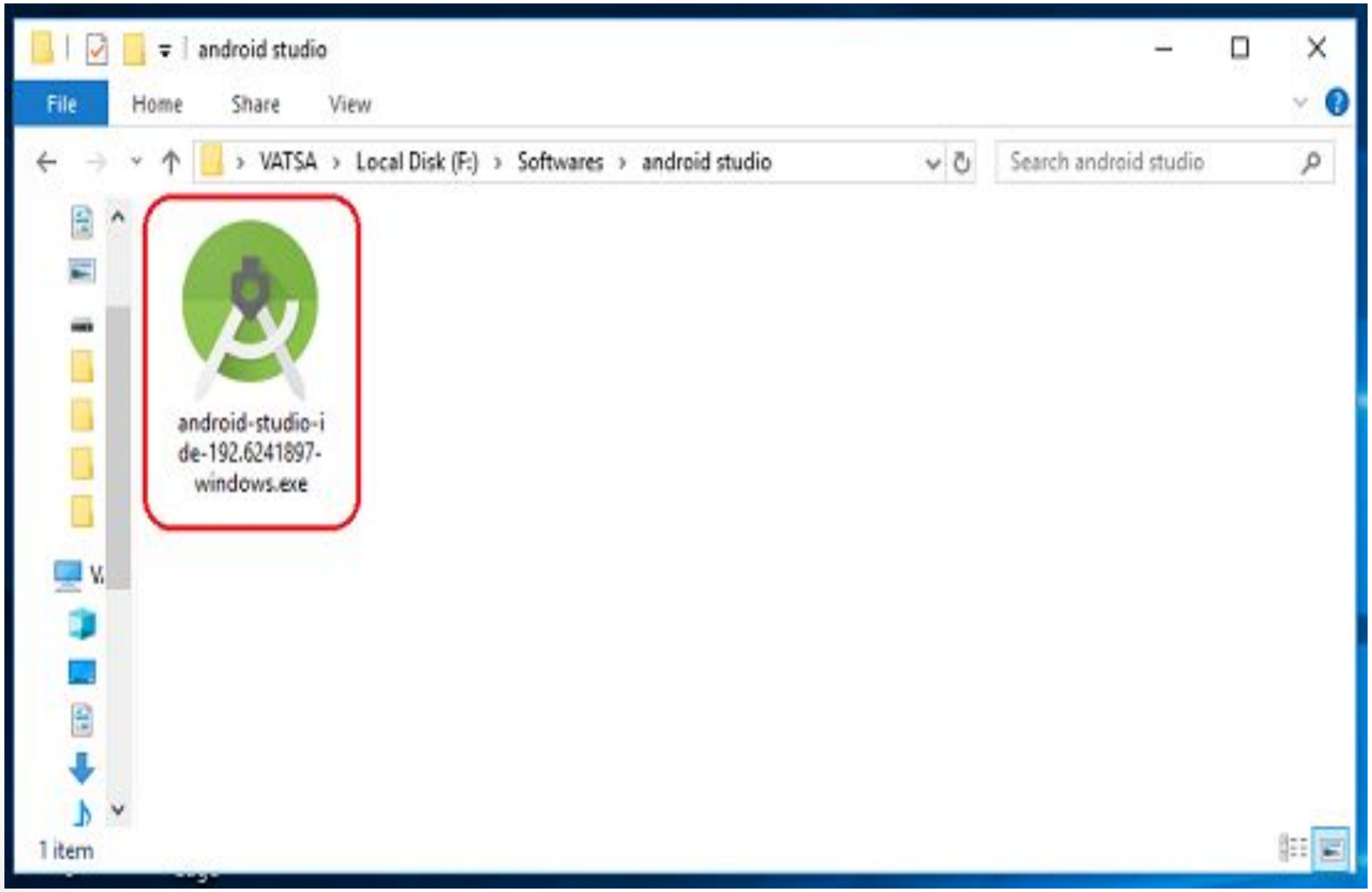
The Developer Preview for Android 11 is now available; test it out and share your feedback.

android studio

Android Studio provides the fastest tools for building apps on every type of Android device.

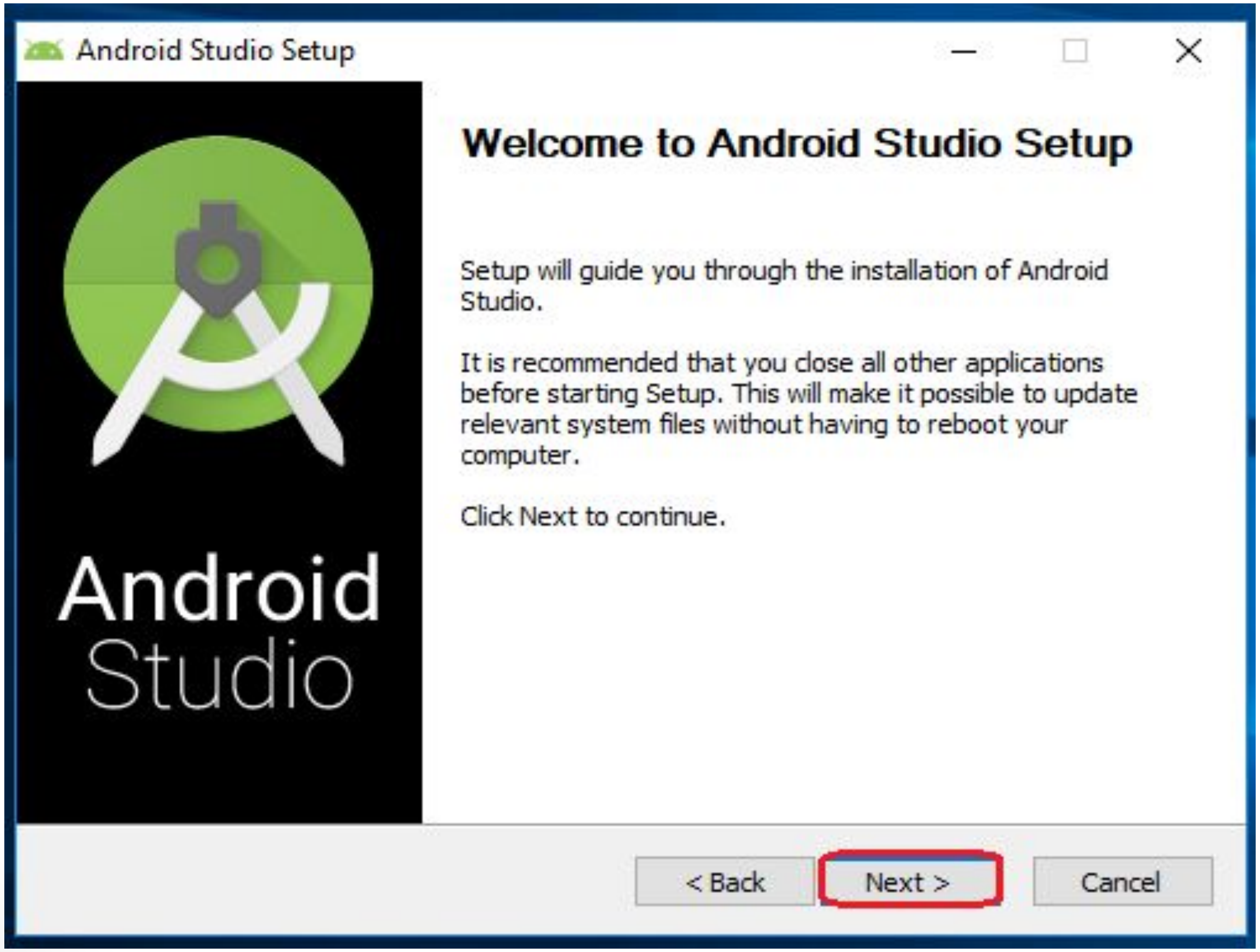
Step 3

Double click on the downloaded "Android Studio-ide.exe" file.


android studio
File
Home
Share
View
VATSA › Local Disk (F:) › Softwares › android studio
Search android studio
android-studio-ide-192.6241897-windows.exe
1 item

Step 4

"Android Studio Setup" will appear on the screen and click "Next" to proceed.



Android Studio Setup

Welcome to Android Studio Setup

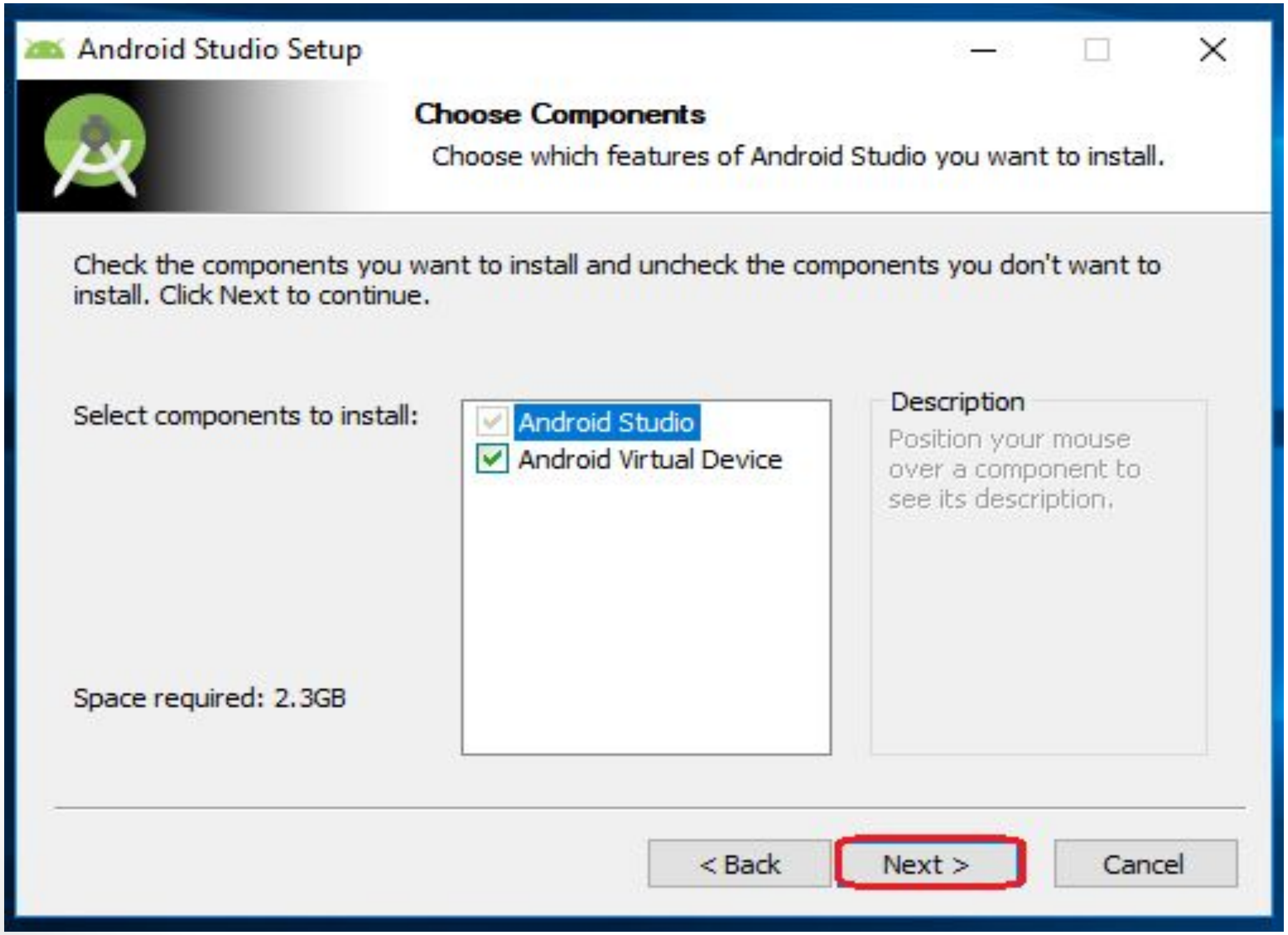
Setup will guide you through the installation of Android Studio.

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.

Step 5

Select the components that you want to install and click on the "Next" button.



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 Virtual Device

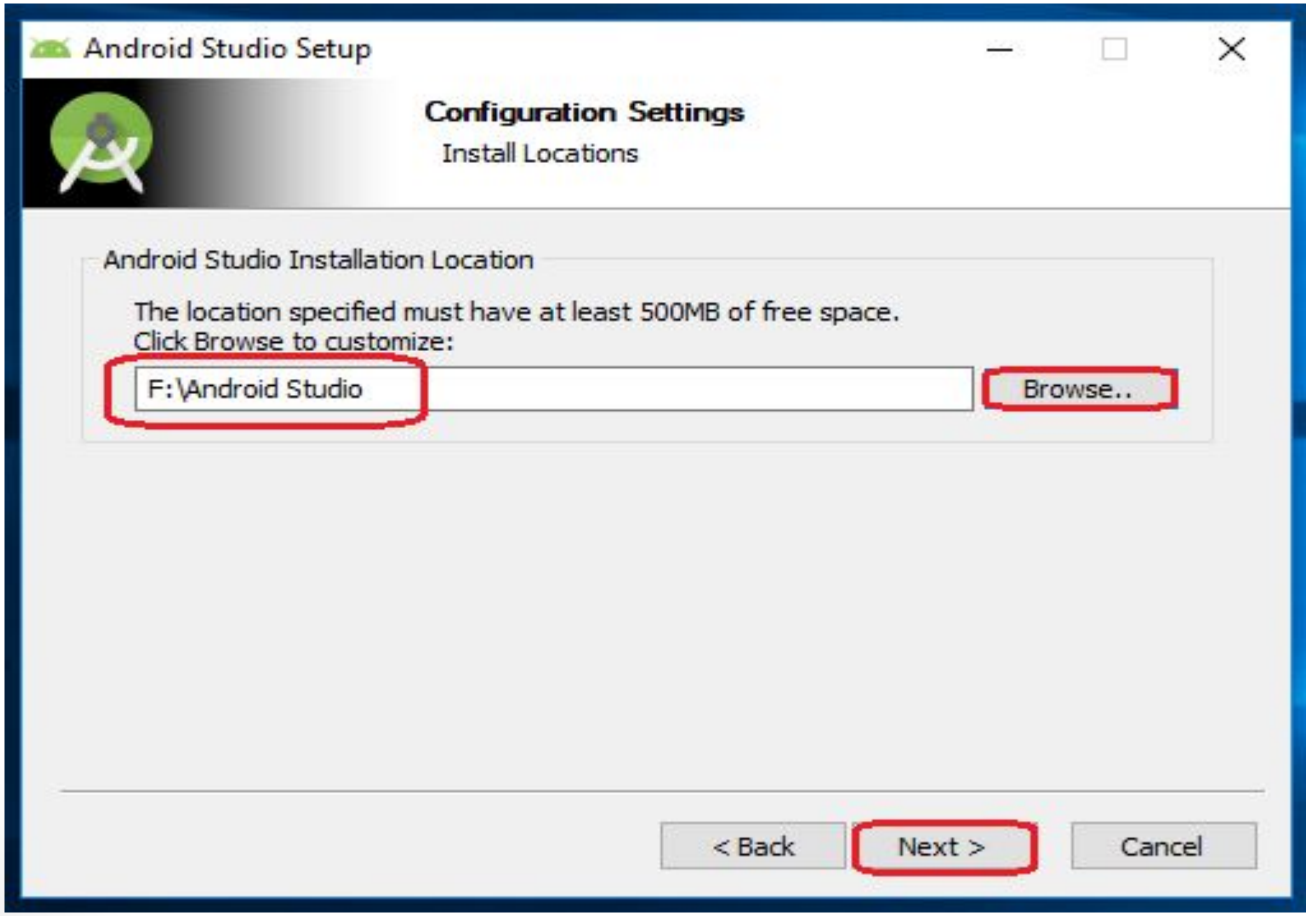
Description

Position your mouse over a component to see its description.

Space required: 2.3GB

Step 6

Now, browse the location where you want to install the Android Studio and click "Next" to proceed.


Android Studio Setup

Configuration Settings

Install Locations

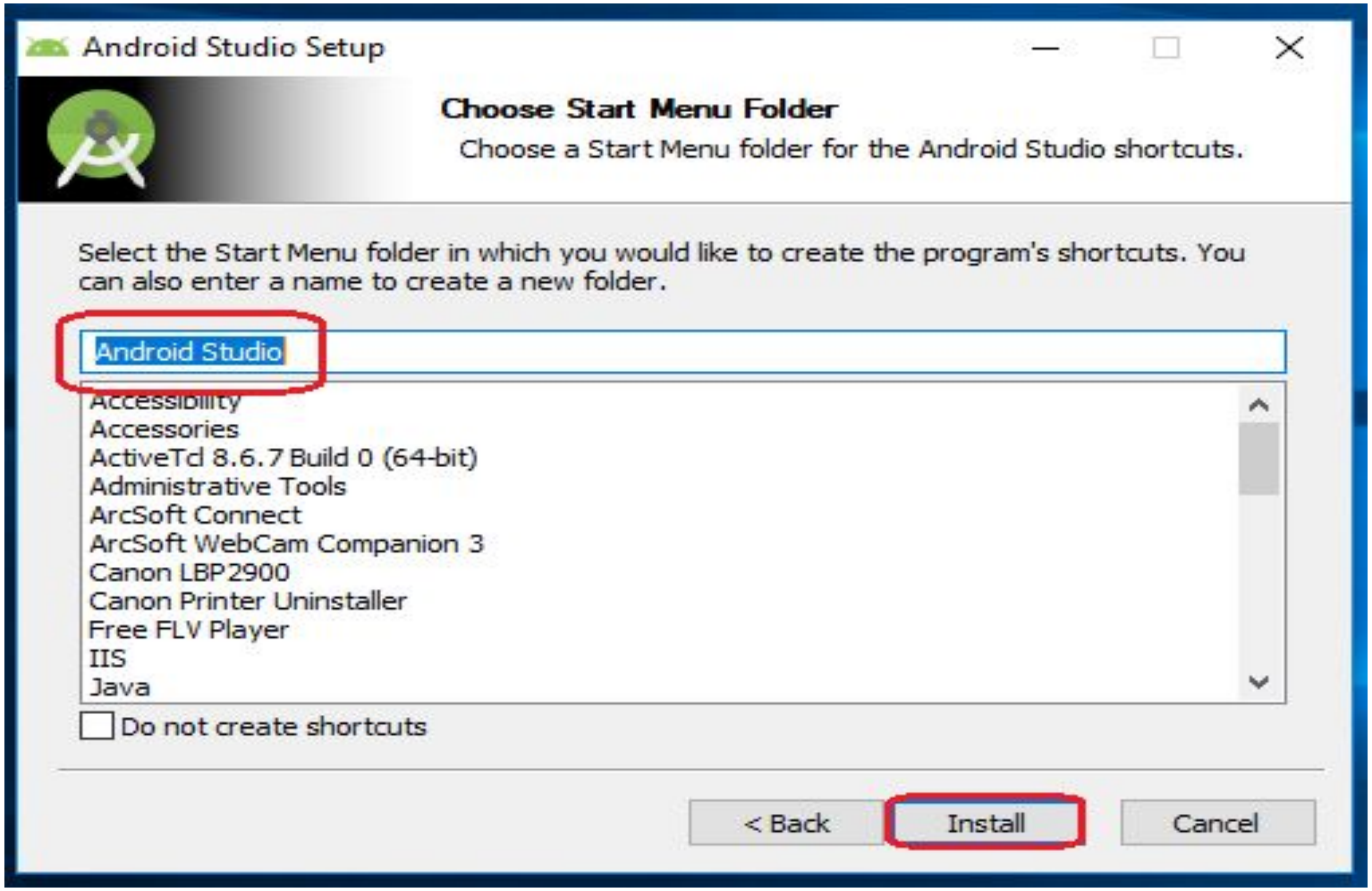
Android Studio Installation Location

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

F:\Android Studio

Step 7

Choose a start menu folder for the "Android Studio" shortcut and click the "Install" button to proceed.



Android Studio Setup

Choose Start Menu Folder

Choose a Start Menu folder for the Android Studio shortcuts.

Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

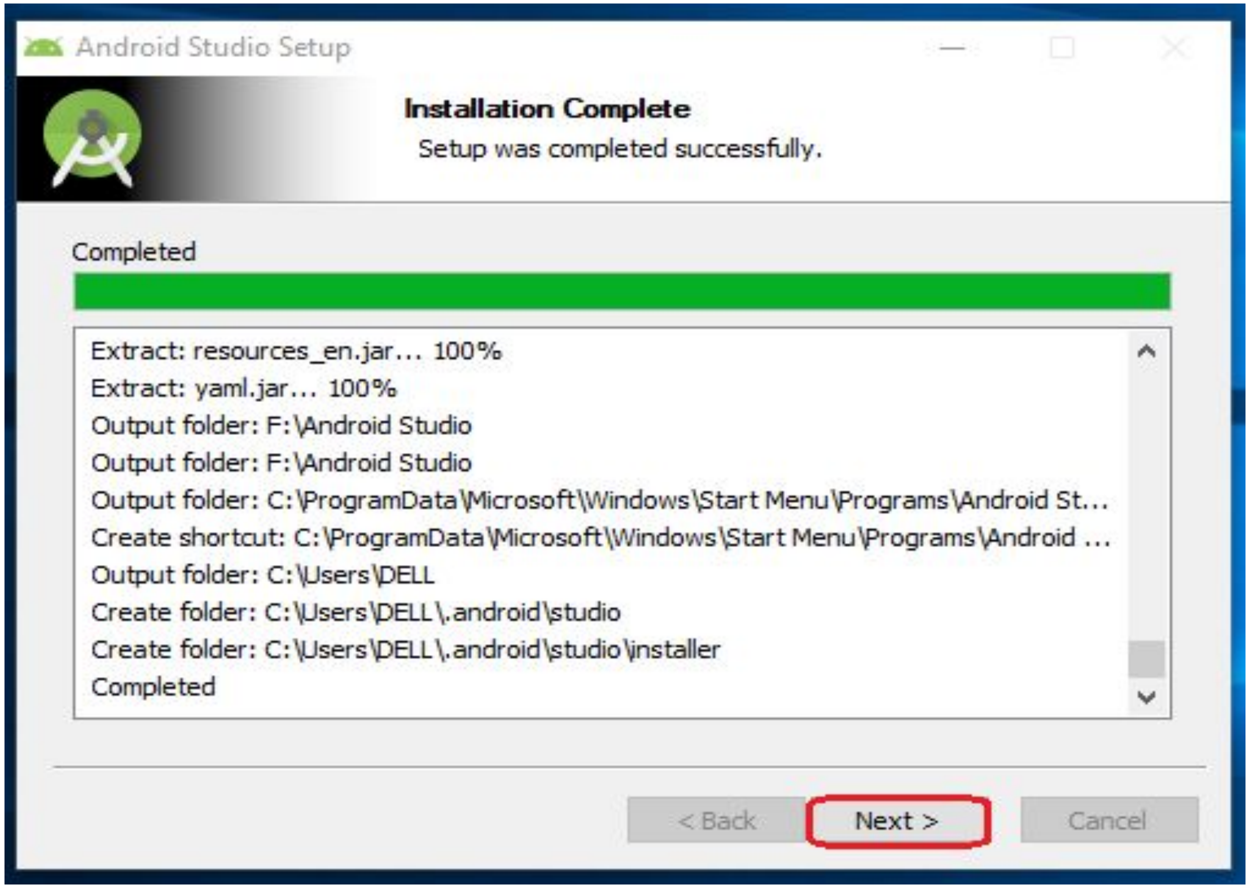
Android Studio

Accessibility
Accessories
ActiveTcl 8.6.7 Build 0 (64-bit)
Administrative Tools
ArcSoft Connect
ArcSoft WebCam Companion 3
Canon LBP2900
Canon Printer Uninstaller
Free FLV Player
IIS
Java

☐ Do not create shortcuts

Step 8

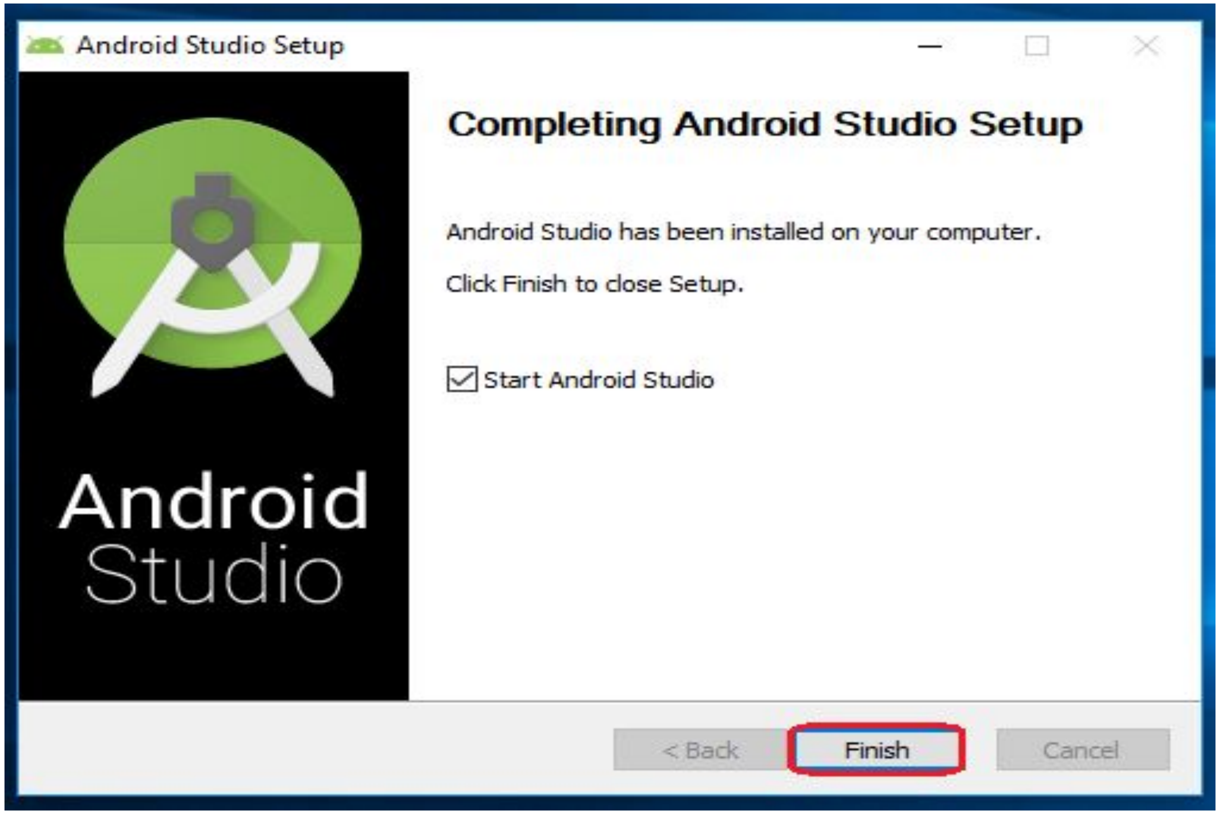
After the successful completion of the installation, click on the "Next" button.



```
Extract: resources_en.jar... 100%
Extract: yaml.jar... 100%
Output folder: F:\Android Studio
Output folder: F:\Android Studio
Output folder: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Android St...
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Android ...
Output folder: C:\Users\DELL
Create folder: C:\Users\DELL\.android\studio
Create folder: C:\Users\DELL\.android\studio\installer
Completed
```

Step 9

Click on the "Finish" button to proceed.


Android Studio Setup

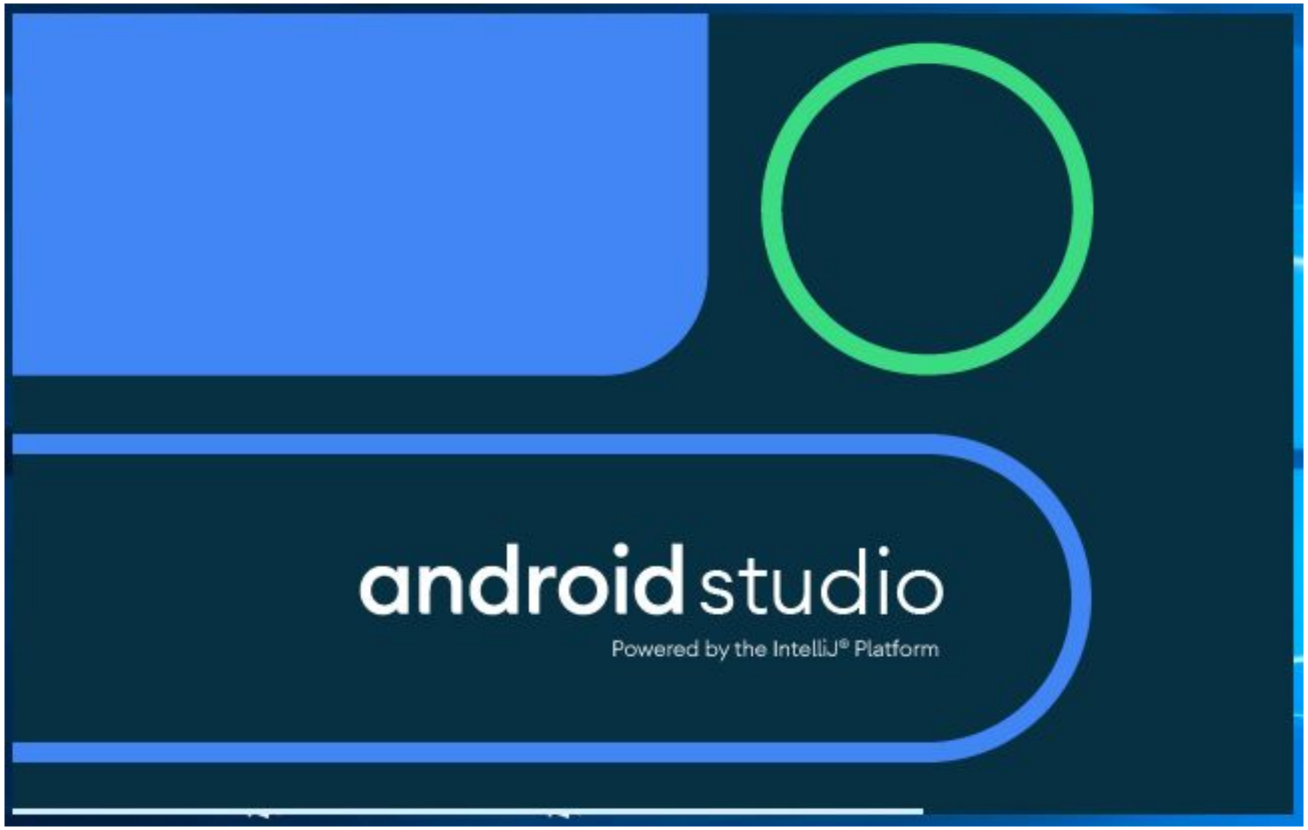
Completing Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

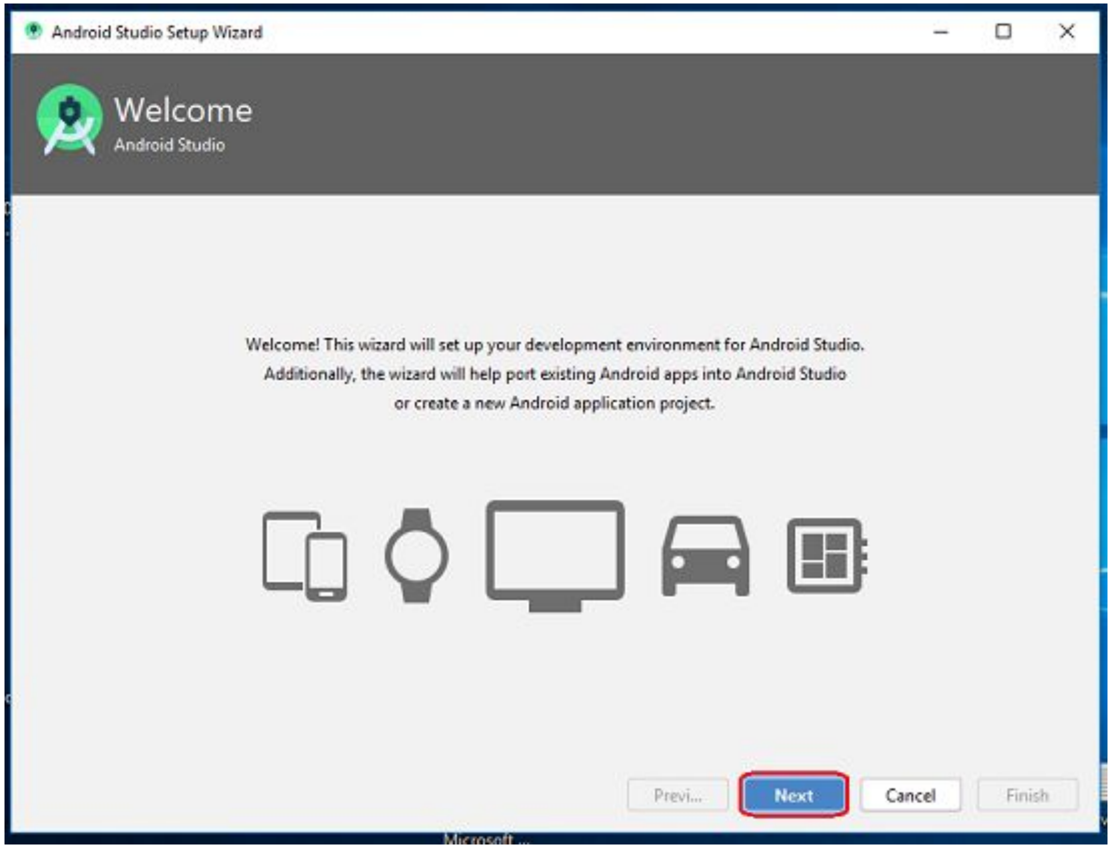
[✓] Start Android Studio

Now, your Android studio welcome screen will appear on the screen.



Step 10

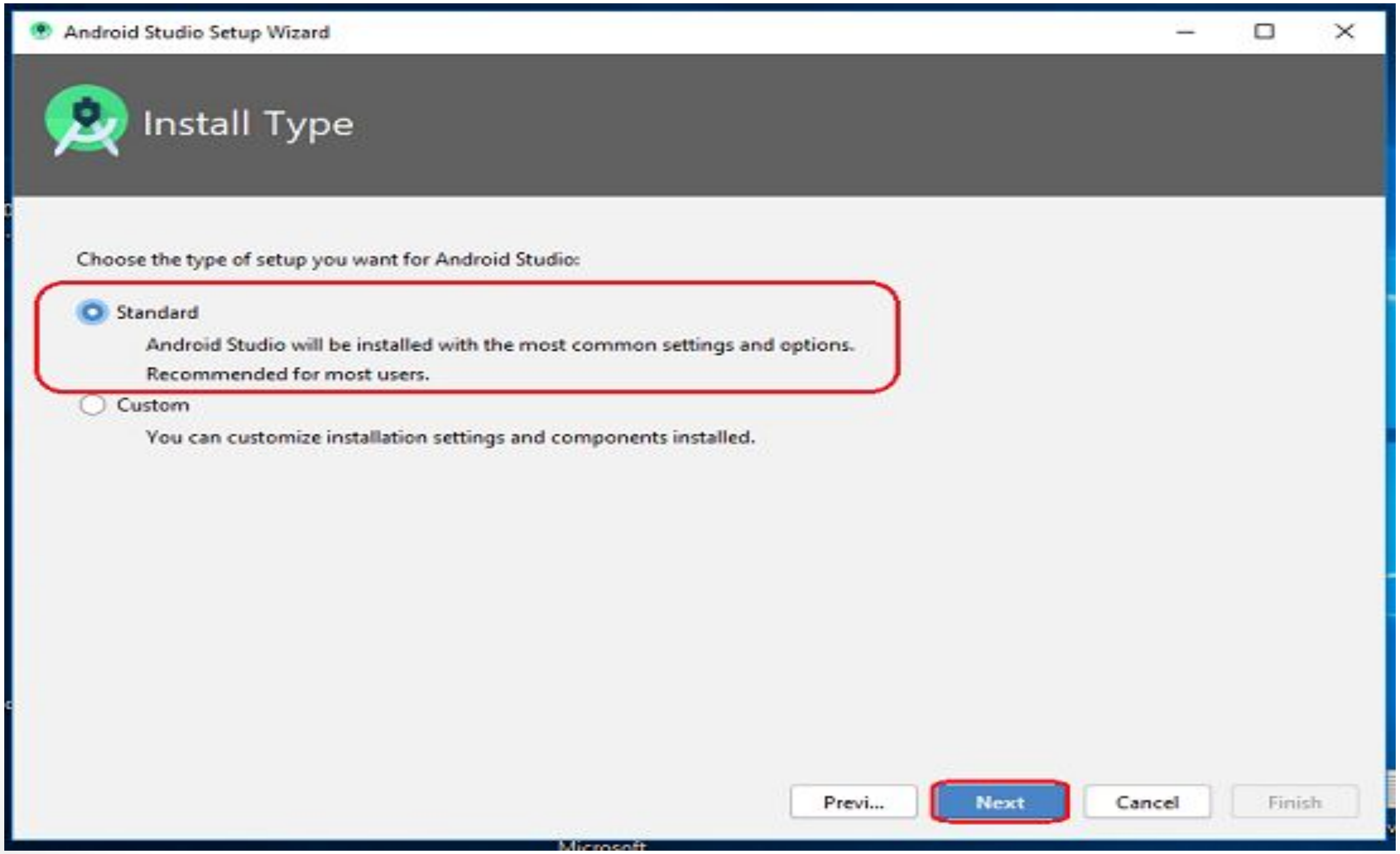
"Android Studio Setup Wizard" will appear on the screen with the welcome wizard. Click on the "Next" button.



Welcome! This wizard will set up your development environment for Android Studio. Additionally, the wizard will help port existing Android apps into Android Studio or create a new Android application project.

Step 11

Select (check) the "Standard" option if you are a beginner and do not have any idea about Android Studio. It will install the most common settings and options for you. Click "Next" to proceed.



Android Studio Setup Wizard

Install Type

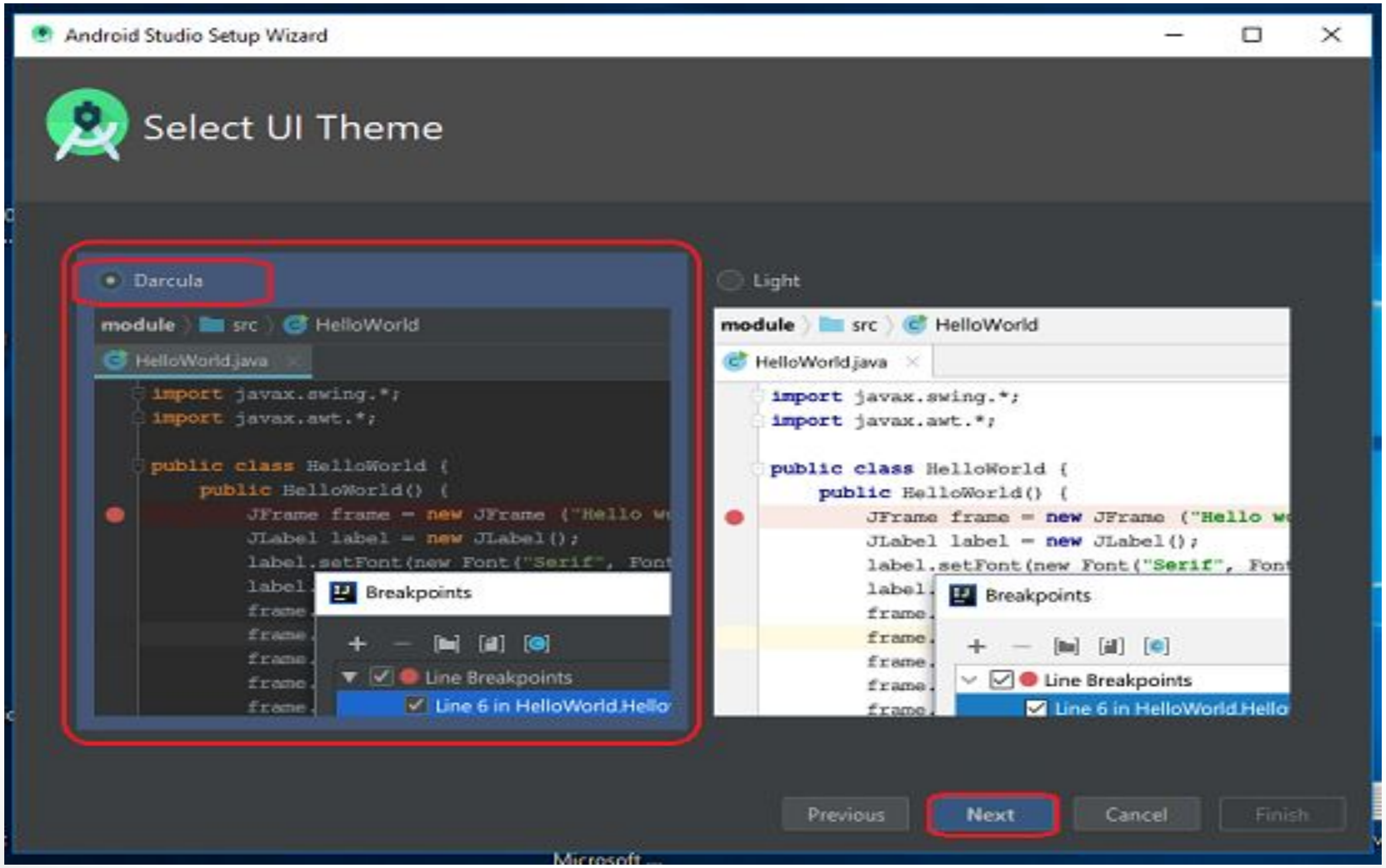
Choose the type of setup you want for Android Studio:

[✓ Standard]
Android Studio will be installed with the most common settings and options.
Recommended for most users.

Custom
You can customize installation settings and components installed.

Step 12

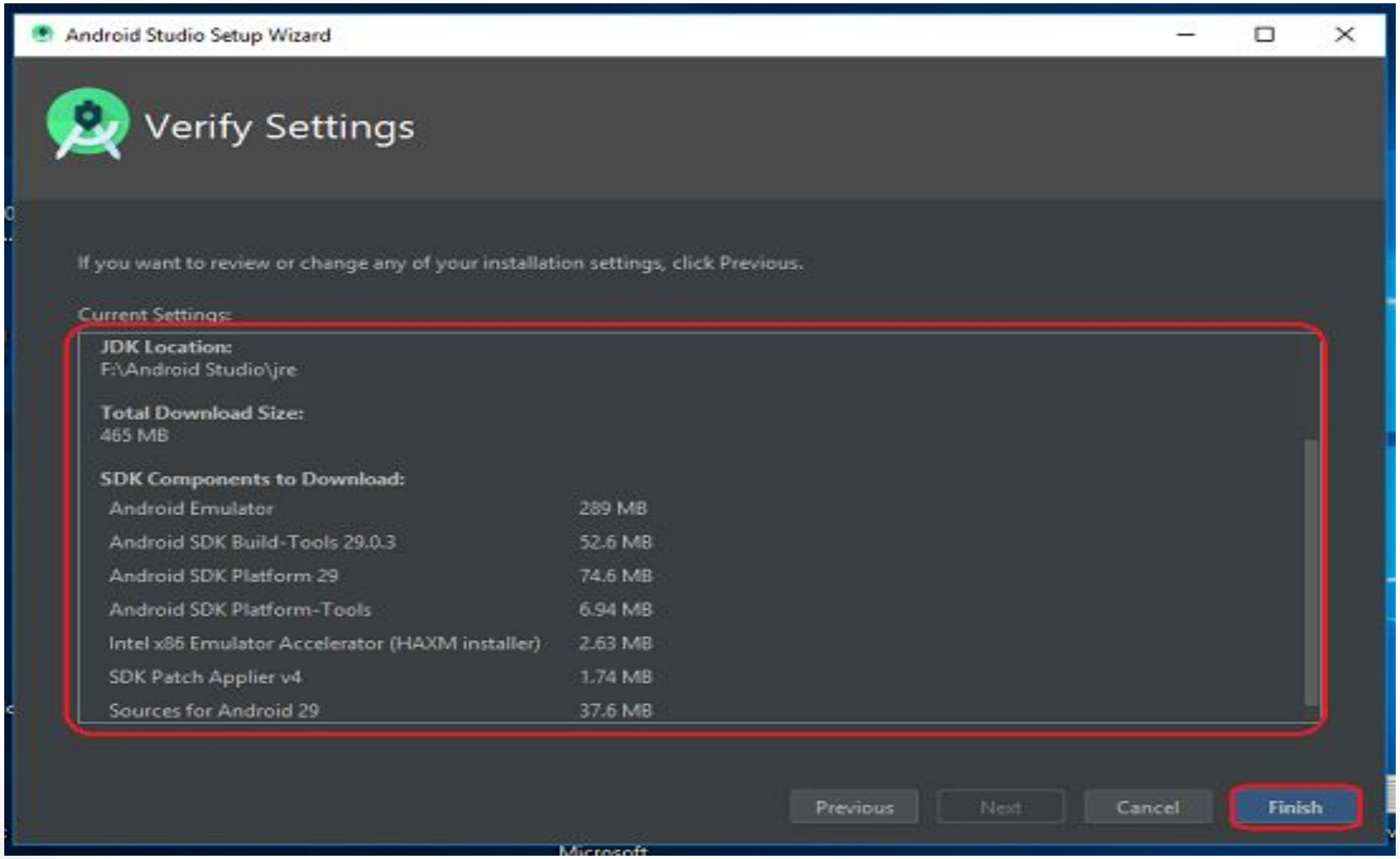
Now, select the user interface theme as you want. (Dark theme (Dracula) is most liked by the coders). Then, click on the "Next" button.



```
module 〉 src 〉 HelloWorld
HelloWorld.java
import javax.swing.*;
import javax.awt.*;

public class HelloWorld {
    public HelloWorld() {
        JFrame frame = new JFrame ("Hello wo
        JLabel label = new JLabel();
        label.setFont(new Font("Serif", Font
        label.
        frame.
        frame.
        frame.
        frame.
        frame.
```

Step 13

Now, click on the "Finish" button to download all the SDK components.



If you want to review or change any of your installation settings, click Previous.

Current Settings:

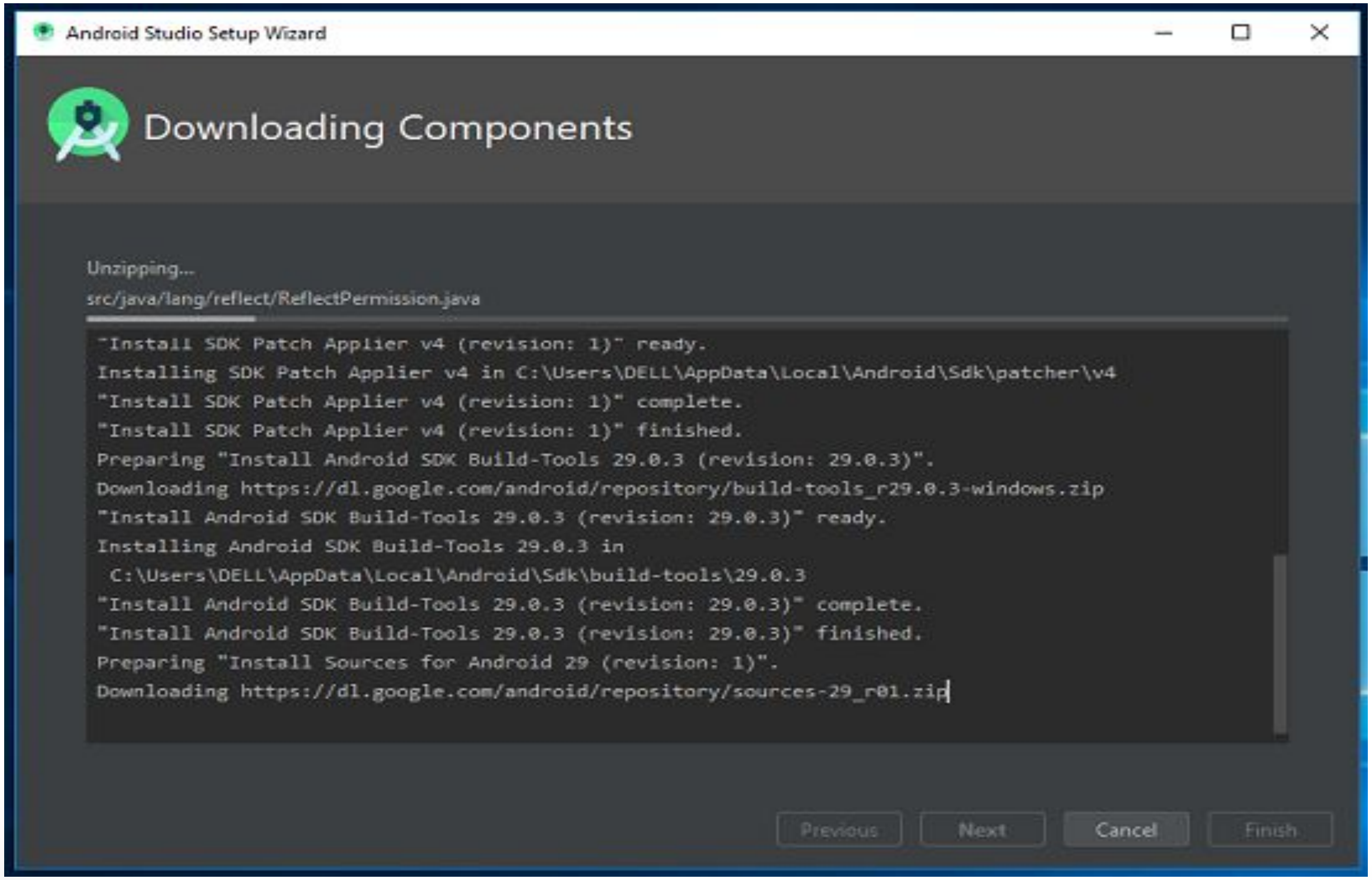
JDK Location:
F:\Android Studio\jre

Total Download Size:
465 MB

SDK Components to Download:

Android Emulator	289 MB
Android SDK Build-Tools 29.0.3	52.6 MB
Android SDK Platform 29	74.6 MB
Android SDK Platform-Tools	6.94 MB
Intel x86 Emulator Accelerator (HAXM installer)	2.63 MB
SDK Patch Applier v4	1.74 MB
Sources for Android 29	37.6 MB

And, the downloading and installation process of components gets started.



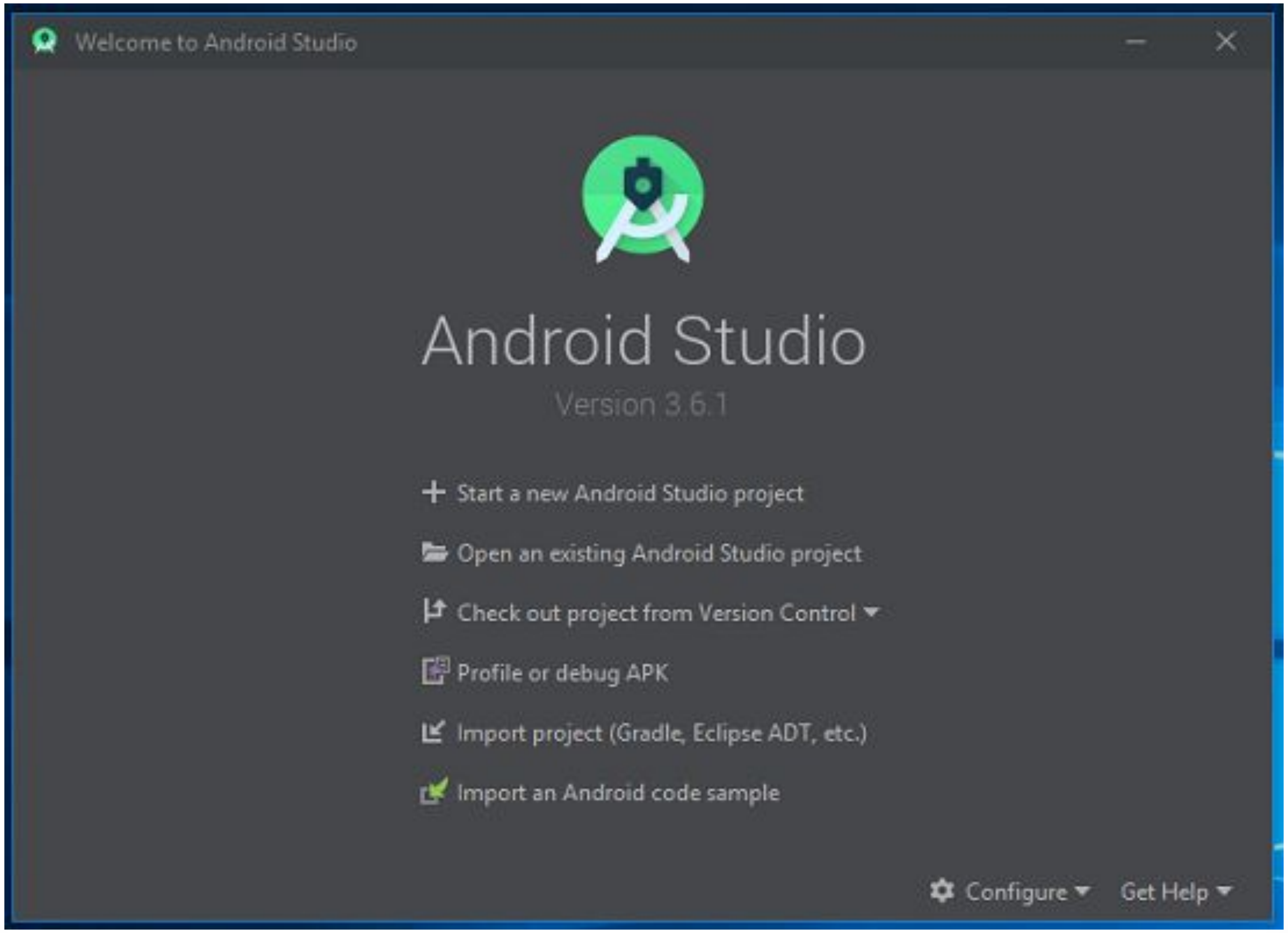
Unzipping...

src/java/lang/reflect/ReflectPermission.java

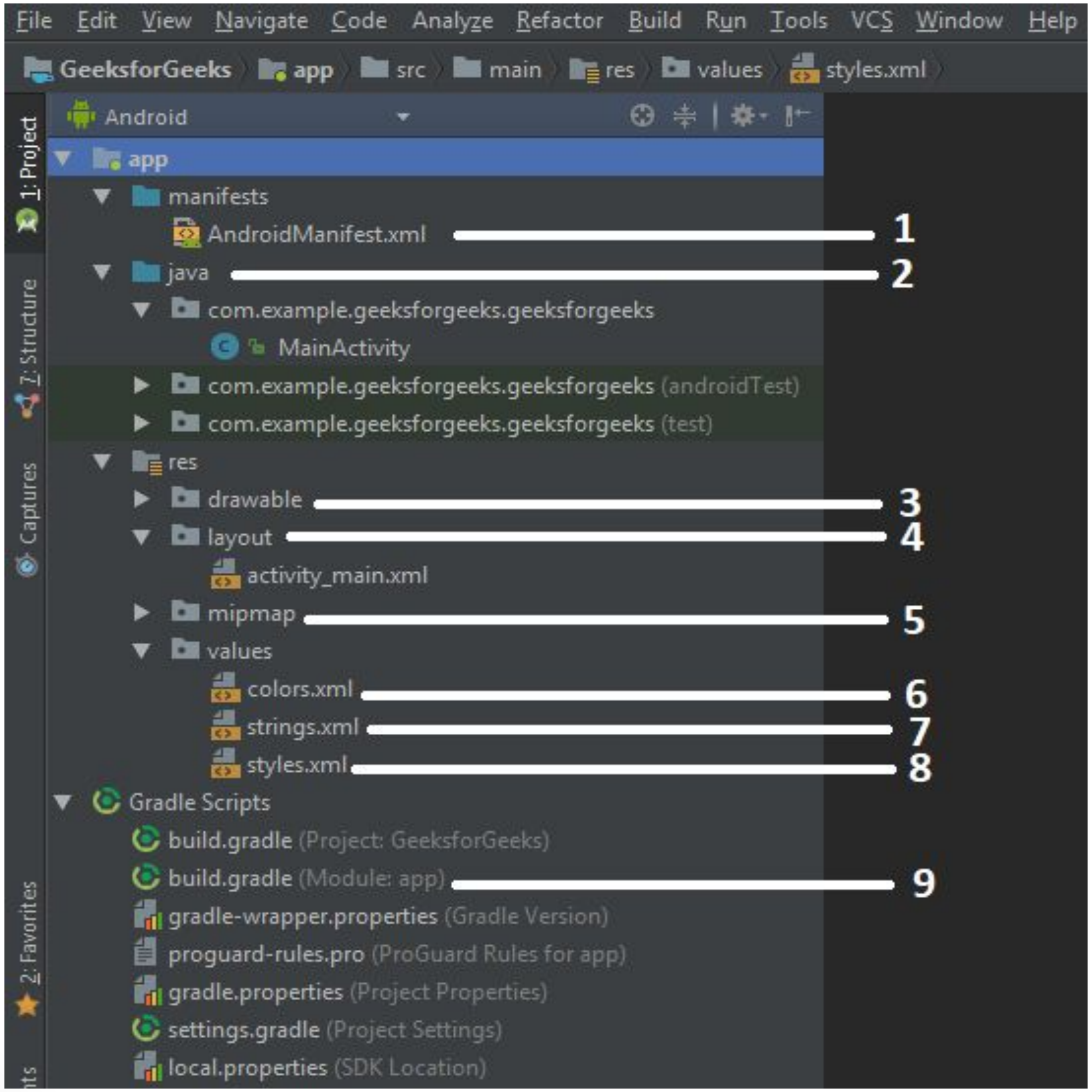
```
"Install SDK Patch Applier v4 (revision: 1)" ready.
Installing SDK Patch Applier v4 in C:\Users\DELL\AppData\Local\Android\Sdk\patcher\v4
"Install SDK Patch Applier v4 (revision: 1)" complete.
"Install SDK Patch Applier v4 (revision: 1)" finished.
Preparing "Install Android SDK Build-Tools 29.0.3 (revision: 29.0.3)".
Downloading https://dl.google.com/android/repository/build-tools_r29.0.3-windows.zip
"Install Android SDK Build-Tools 29.0.3 (revision: 29.0.3)" ready.
Installing Android SDK Build-Tools 29.0.3 in
 C:\Users\DELL\AppData\Local\Android\Sdk\build-tools\29.0.3
"Install Android SDK Build-Tools 29.0.3 (revision: 29.0.3)" complete.
"Install Android SDK Build-Tools 29.0.3 (revision: 29.0.3)" finished.
Preparing "Install Sources for Android 29 (revision: 1)".
Downloading https://dl.google.com/android/repository/sources-29_r01.zip
```

Step 14

After downloading all the necessary components, click on the "Finish" button.


Welcome to Android Studio
Android Studio
Version 3.6.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

STRUCTURE OF AN ANDROID APPLICATION


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
GeeksforGeeks › app › src › main › res › values › styles.xml
1: Project
7: Structure
Captures
2: Favorites
Android

```
▼ app
    ▼ manifests
        AndroidManifest.xml ——————————— 1
    ▼ java ——————————————————————— 2
        ▼ com.example.geeksforgeeks.geeksforgeeks
            MainActivity
        ▶ com.example.geeksforgeeks.geeksforgeeks (androidTest)
        ▶ com.example.geeksforgeeks.geeksforgeeks (test)
    ▼ res
        ▶ drawable ——————————————————— 3
        ▼ layout ————————————————————— 4
            activity_main.xml
        ▶ mipmap ————————————————————— 5
        ▼ values
            colors.xml ——————————————— 6
            strings.xml —————————————— 7
            styles.xml ——————————————— 8
▼ Gradle Scripts
    build.gradle (Project: GeeksforGeeks)
    build.gradle (Module: app) ——————— 9
    gradle-wrapper.properties (Gradle Version)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

1. **AndroidManifest.xml**:

- Every project in Android includes a manifest file, which is AndroidManifest.xml, stored in the root directory of its project hierarchy.

- An interface btw android os and your application.

- The manifest file is an important part of our app because it defines the structure and metadata of our application, its components, and its requirements.

- This file includes nodes for each of the **Activities, Services, Content Providers and Broadcast Receiver** that make the application and using Intent Filters and Permissions, determines how they co-ordinate with each other and other applications.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.geeksforgeeks.geeksforgeeks">
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

- *<application>…..</application>* tags enclose components related to the application.

- *Android::icon* -refers to the application icon under res/drawable-hdpi

ie application uses image ic_launcher located in the drawable folders

- *<activity>* tag is used to specify an activity and *android::name* specifies the fully qualified classname of the activity subclass.

- *Android:label* -specifies the string to use as label for the activity.

- The action for intent filter is *android.intent.action.MAIN* to indicate that this activity serves as the entry point for the application.

- Category for the intent filter is named as *android.intent.category.LAUNCHER* –indicates that app can be launched from device’s launcher icon.

2.Java: The Java folder contains the .Java source code files. These files are used as a controller for controlled UI (Layout file). It gets the data from the Layout file and after processing that data ,output will be shown in the UI layout.

By default, includes a **MainActivity.java** source file having an activity class that runs when your app is launched using app icon.(this file is converted to dalvic executable code and runs application)

3.drawable: A Drawable folder contains resource type file (something that can be drawn). Drawables may take a variety of files like Bitmap (PNG, JPEG), Nine Patch, Vector (XML), Shape, Layers, States, Levels, and Scale.

Res/drawable-hdpi-Directory for drawable objects for high density screens.

4.layout: A layout defines the **visual structure for a user interface**, such as the UI for an Android application. This folder stores Layout files that are written in XML language. You can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
  <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
  <Button android:id="@+id/button"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="Hello, I am a Button" />
</LinearLayout>
```

5.mipmap: Mipmap folder contains the **Image Asset file** that can be used in Android Studio application. You can generate the following icon types like **Launcher icons, Action bar and tab icons, and Notification icons**.

6. **colors.xml**: colors.xml file contains color resources of the Android application. Different color values are identified by a unique name that can be used in the Android application program.

Below is a sample colors.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="colorPrimary">#3F51B5</color>
   <color name="colorPrimaryDark">#303F9F</color>
   <color name="colorAccent">#FF4081</color>
</resources>
```

7.strings.xml: The strings.xml file (located in res/values folder)contains string resources of the Android application. The different string value is identified by a unique name that can be used in the Android application program. Names of buttons, labels,default text and similar type of strings go to this file.

```
<resources>
   <string name="app_name">Hello World</string>
 <string name=“menu_settings">Settings</string>
</resources>
```

8.styles.xml: The styles.xml file contains resources of the theme style in the Android application. This file is written in XML language

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

9.build.gradle(Module: app): This defines the module-specific build configurations.This is an **autogenerated file** containing compile sdk version,build tools version,min sdk version,target sdk version,version code and version name. Here you can add dependencies what you need in your Android application

```
apply plugin: 'com.android.application'
android {
   compileSdkVersion 26
   defaultConfig {
      applicationId "com.example.geeksforgeeks.geeksforgeeks"
      minSdkVersion 16
      targetSdkVersion 26
      versionCode 1
      versionName "1.0"
      testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
   }
   buildTypes {
      release {
         minifyEnabled false
         proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
      }
   }
}
dependencies {
   implementation fileTree(dir: 'libs', include: ['*.jar'])
   implementation 'com.android.support:appcompat-v7:26.1.0'
   implementation 'com.android.support.constraint:constraint-layout:1.0.2'
   testImplementation 'junit:junit:4.12'
   androidTestImplementation 'com.android.support.test:runner:0.5'
   androidTestImplementation 'com.android.support.test.espresso:espresso-core:2.2.2'
}
```

LAYOUTS

How to use layouts, fragments, place views to create functional UI.

FAMILIARISATION OF SCREEN COMPONENTS

An activity displays the user interface of our application and contains widgets like button, image button, textbox, label etc.

We define our UI using an Xml file named activity_main.xml located in the res folder inside Layout folder.

activity_main.xml

```
<?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=".MainActivity">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

MainActivity.java:

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

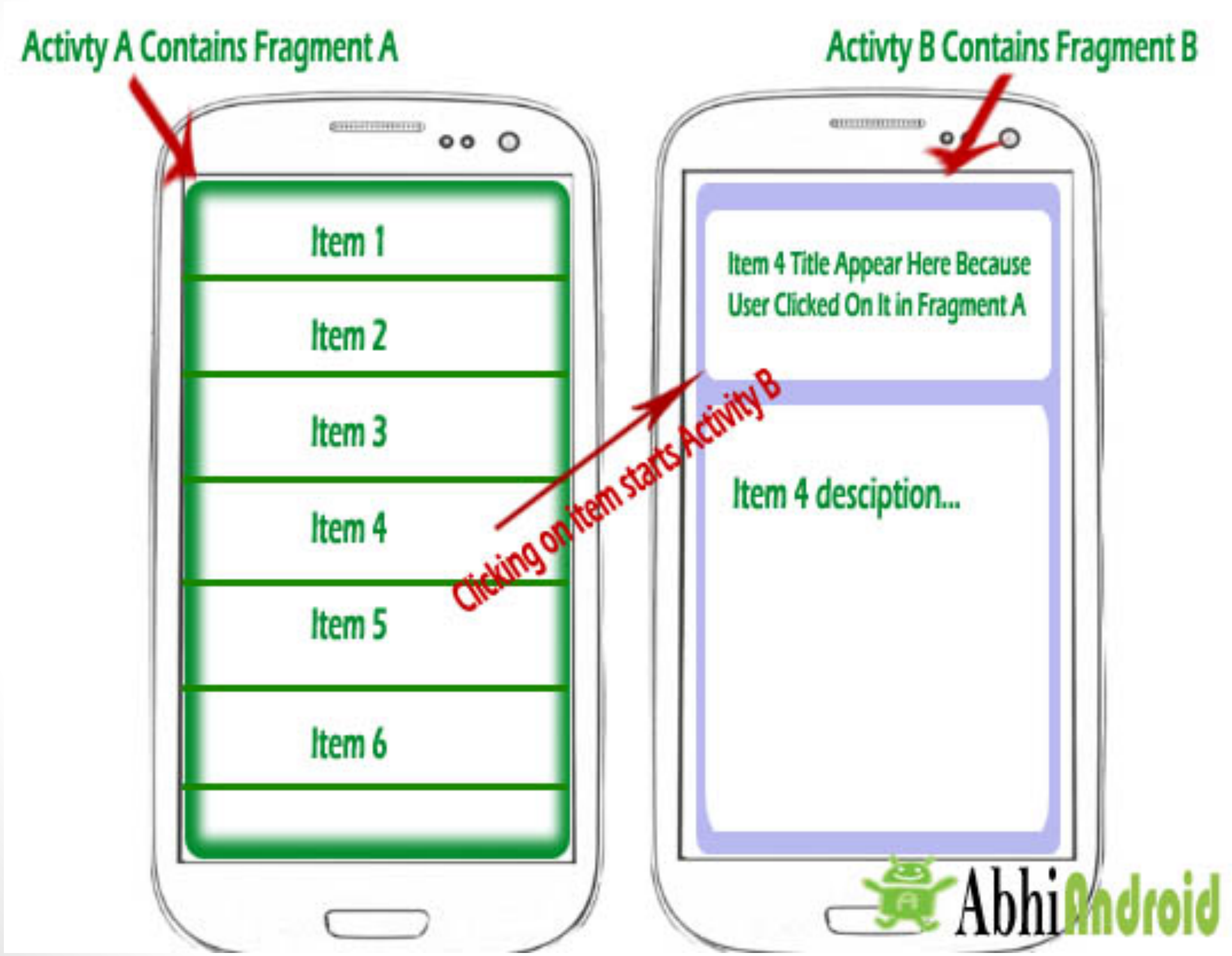
During runtime,we can load the xml UI in the onCreate() eventhandler in Activity class using setContentView() method of the Activity class.

FUNDAMENTAL USER INTERFACE DESIGN

- Every component in a UI is a subclass of Android View class which is present in the package android.view.
- Android SDK provides a set of pre-built views to construct user interfaces like Buttons,ImageButtons,TextView class etc

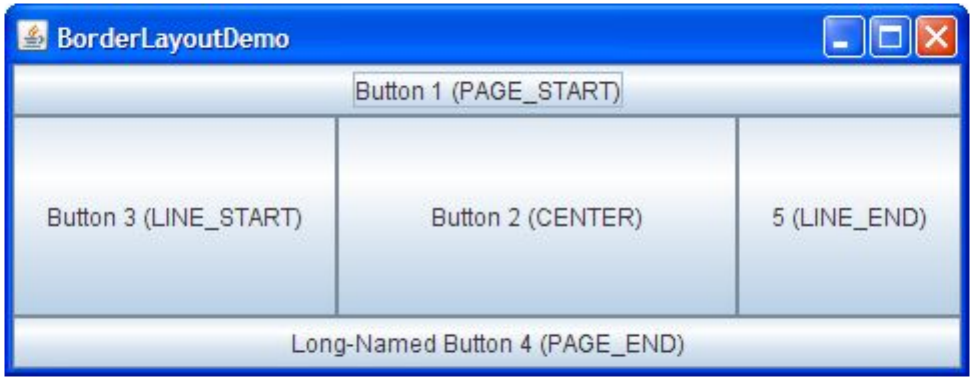
- **1.Views**-base class for all visual interface elements (commonly known as controls or widgets) and occupies a rectangular area on the screen and is responsible for drawing and event handling. All UI controls including the layout classes are derived from view.
- **2.ViewGroups**-extension of View class that can contain multiple child views.
- **3.Activity**-Android equivalent of forms(window or screen to be displayed).To display a UI,you assign a view(layout) to a activity.
- **4.Fragments**-to encapsulate portions of ui.(useful when optimizing ui layouts for different screen sizes)

A **Fragment** is a piece of an activity which enable more modular activity design. A fragment is a kind of **sub-activity**.

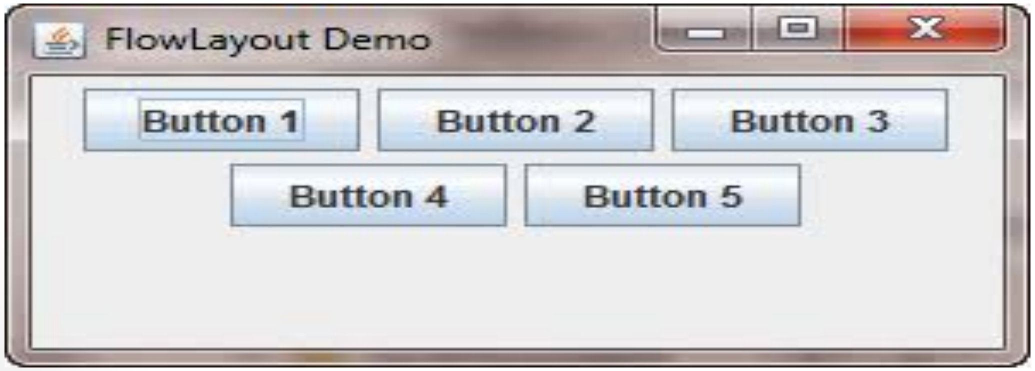

Activty A Contains Fragment A
Activty B Contains Fragment B
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Clicking on Item starts Activity B
Item 4 Title Appear Here Because User Clicked On It in Fragment A
Item 4 desciption...
AbhiAndroid

Layouts in java

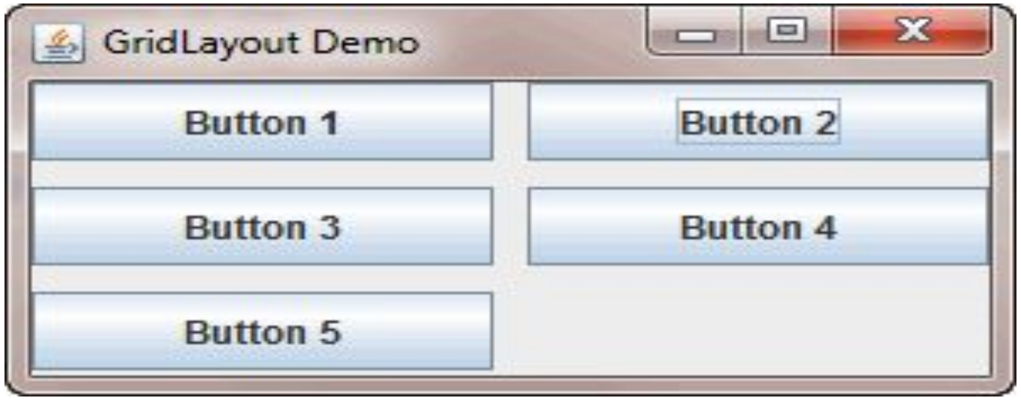
Border layout



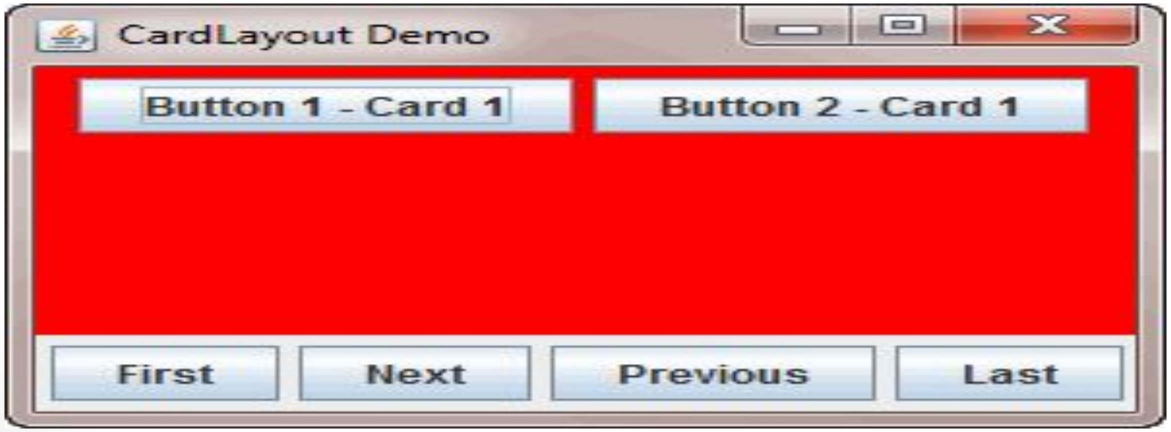
Flow layout -default layout of applet and panel



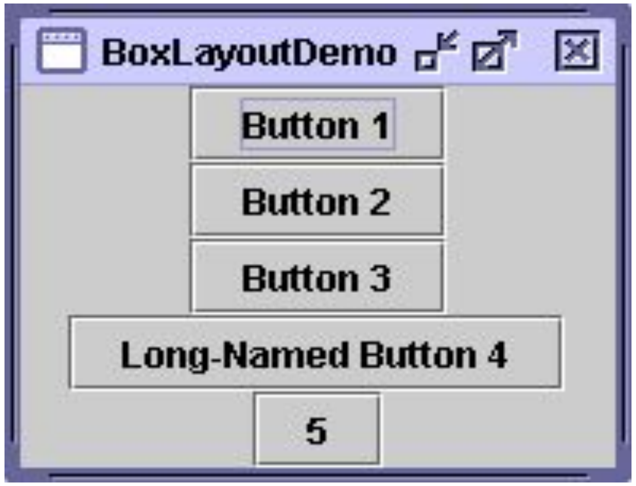
Grid layout



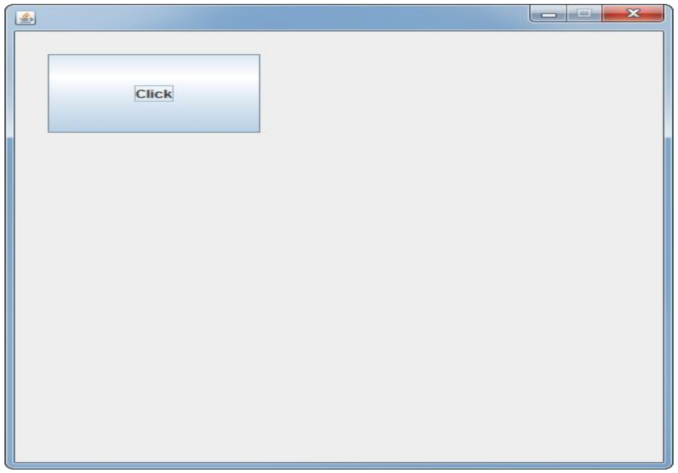
Card layout



Box layout



Null layout

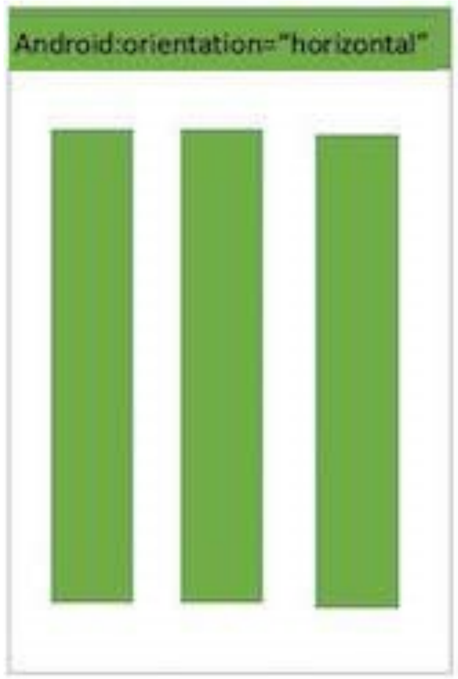


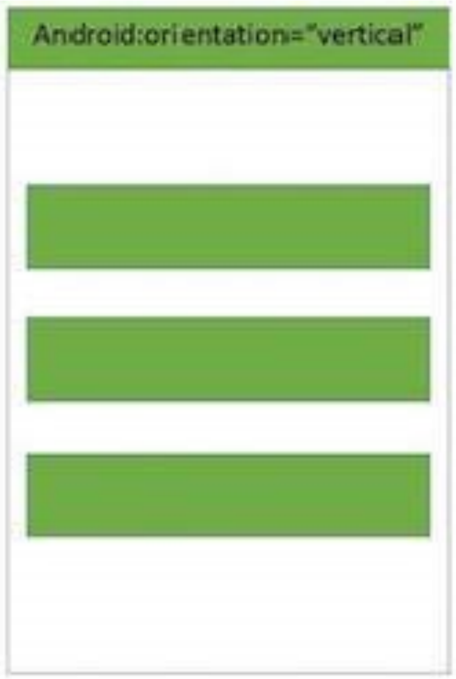
Layout views in Android SDK

Android contains the following commonly used ViewGroupsubclasses:

1.LINEAR LAYOUT

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





res/layout/activity_main.xml

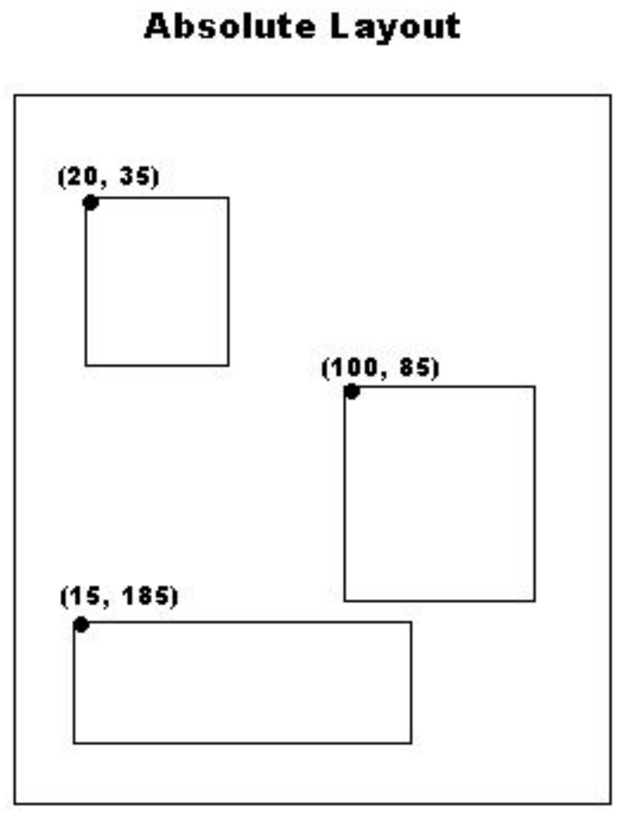
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >

 <Button android:id="@+id/btnStartService“
 android:layout_width="270dp"
android:layout_height="wrap_content"
android:text="start_service"/>

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

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

Attributes and Description

Android:id- uniquely identifies the view
android:orientation—Used for arranging the controls in the container in horizontal or vertical order
android:layout_width—Used for defining the width of a control
android:layout_height—Used for defining the height of a control
android:padding—Used for increasing the whitespace between the boundaries of the control and its actual content
android:layout_weight—Used for shrinking or expanding the size of the control to consume the extra space relative to the other controls in the container
android:gravity—Used for aligning content within a control
android:layout_gravity—specifies how child views are positioned
Android:layout_x-specifies x coordinate of the layout
Android:layout_y-specifies y coordinate of the layout
Android:layout_width=wrap_content tell your view to size itself to the dimensions required by its content.
Android:layout_width=fill_parent tells your view to become as big as its parent view

2.ABSOLUTE LAYOUT

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



ATTRIBUTES

1	**android:id** This is the ID which uniquely identifies the layout.
2	**android:layout_x** This specifies the x-coordinate of the view.
3	**android:layout_y** This specifies the y-coordinate of the view.

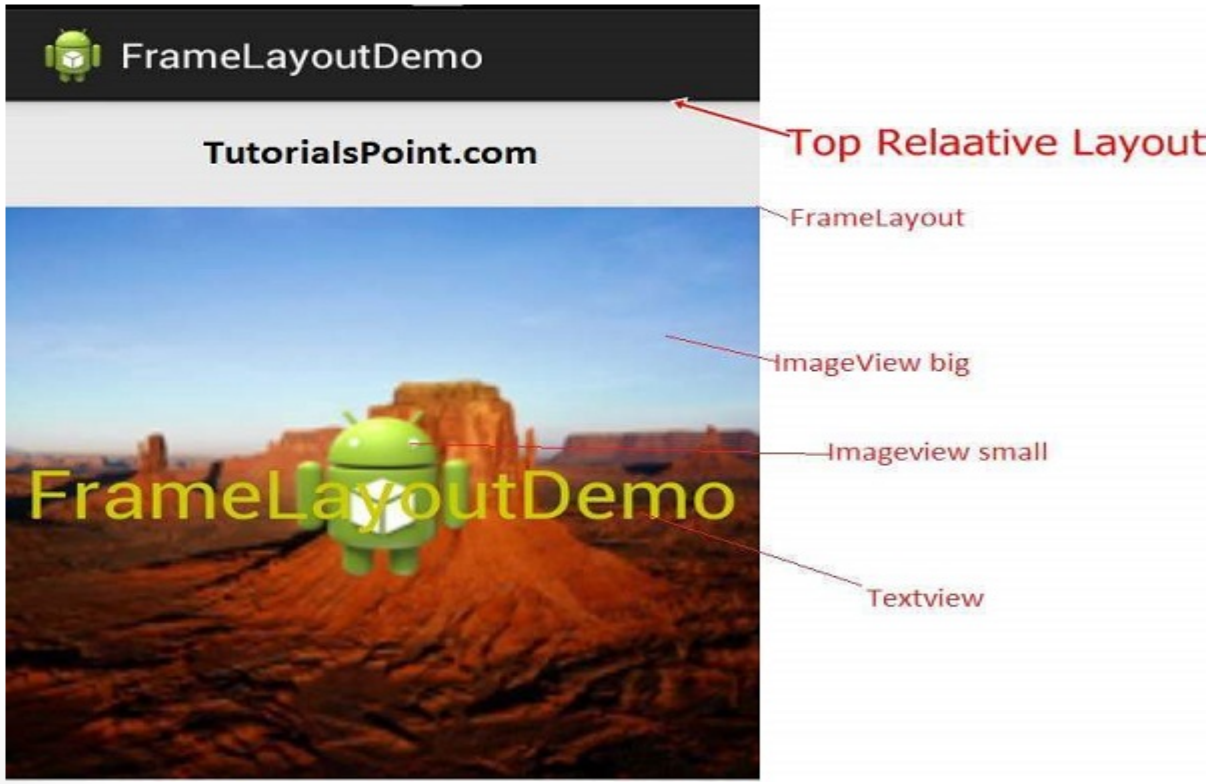
res/layout/activity_main.xml

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

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

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

3.FRAME LAYOUT

Frame Layout is designed to allocate an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, (ie a single item once).We can have multiple elements, each element will be positioned based on top left of screen.

You can, however, add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the android:layout_gravity attribute.



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

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

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

 </FrameLayout>
```

Sr.No	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the layout.
2	**android:foreground** This defines the drawable to draw over the content and possible values may be a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
3	**android:foregroundGravity** Defines the gravity to apply to the foreground drawable. The gravity defaults to fill. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
4	**android:measureAllChildren** Determines whether to measure all children or just those in the VISIBLE or INVISIBLE state when measuring. Defaults to false.

4.RELATIVE LAYOUT

RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left or center).

A RelativeLayout is a very powerful utility for designing a user interface because it can eliminate nested view groups and keep your layout hierarchy flat, which improves performance.

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

By default, all child views are drawn at the top-left of the layout, so you must define the position of each view using the various layout properties available from **RelativeLayout.LayoutParams** and few of the important attributes are given below

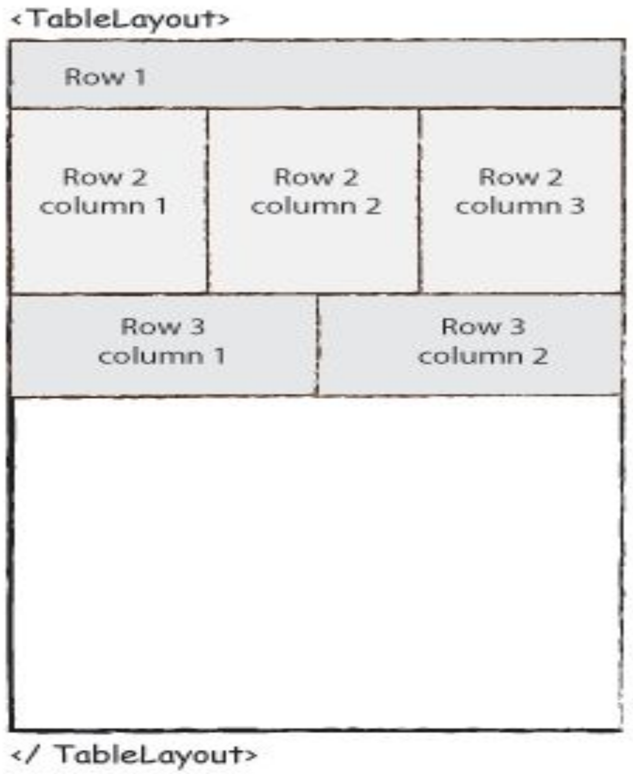
4	**android:layout_alignParentBottom** If true, makes the bottom edge of this view match the bottom edge of the parent. Must be a boolean value, either "true" or "false".
5	**android:layout_alignParentEnd** If true, makes the end edge of this view match the end edge of the parent. Must be a boolean value, either "true" or "false".
6	**android:layout_alignParentLeft** If true, makes the left edge of this view match the left edge of the parent. Must be a boolean value, either "true" or "false".
7	**android:layout_alignParentRight** If true, makes the right edge of this view match the right edge of the parent. Must be a boolean value, either "true" or "false".

9	**android:layout_centerHorizontal** If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".
10	**android:layout_centerInParent** If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".
11	**android:layout_below** Positions the top edge of this view below the given anchor view ID
12	**android:layout_centerVertical** If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".
13	**android:layout_alignParentStart** If true, makes the start edge of this view match the start edge of the parent. Must be a boolean value, either "true" or "false".

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent" android:layout_height="fill_parent"
android:paddingLeft="16dp" android:paddingRight="16dp" >

<EditText  android:id="@+id/name"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="“Enter your name" />

<LinearLayout android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="fill_parent“
 android:layout_alignParentStart="true“
 android:layout_below="@+id/name">

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

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

TABLE LAYOUT

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



Attributes

1	**android:id** This is the ID which uniquely identifies the layout.
2	**android:collapseColumns(make columns invisible)** This specifies the zero-based index of the columns to collapse. The column indices must be separated by a comma: 1, 2, 5.
3	**android:shrinkColumns** The zero-based index of the columns to shrink. The column indices must be separated by a comma: 1, 2, 5.
4	**android:stretchColumns** The zero-based index of the columns to stretch. The column indices must be separated by a comma: 1, 2, 5.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:stretchColumns="2">
  <TableRow>
    <TextView
      android:text="OPEN"
      android:padding="8dip" />
    <TextView
      android:text="CONTROL +0"
      android:gravity="right"
      android:padding="3dip" />
  </TableRow>
  <TableRow>
    <TextView
      android:text="SAVE"
      android:padding="8dip" />
    <TextView
      android:text="CONTROL+SAVE"
      android:gravity="right"
      android:padding="3dip" />
  </TableRow>
</TableLayout>>
```