1. Basics of Android

What is Android...?

- Android is an open source and Linux-based Operating System for mobile devices such as smart phones and tablet computers.
- Android was developed by the Open Handset Alliance, led by Google, and other companies.

First Release

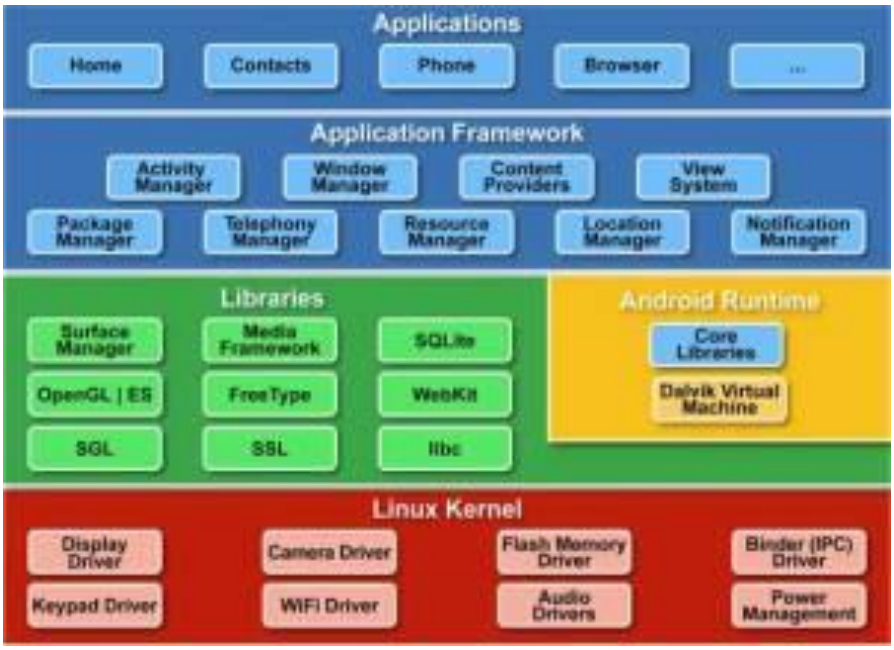
- The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007.
- The first commercial version, Android 1.0, was released in September 2008.
- The first android phone was launched by HTC on October 2008(HTC Dream – T-Mobile G1 in USA and some parts of Europe).

Application

- Set of Programs with dedicated functionality is known as Application
- Two types of Applications based on provided interface
    1. System application - Application which helps to communicate between hardware and user application.
        - Ex: Operating Systems, Driver Software etc...
    2. User application - Applications which provides solution for common problems.
        - Ex: MS Office, Banking Applications, Mail & Message Applications, Video Players, etc...
- There are 2 types of Applications based on where it is installed
    1 Standalone application (unshared)
    2 Web Application (shared)
- Standalone Applications are present in our own computer and they are dedicated per device.
- For ex : Adobe Reader , Web Browser , Media Player etc.
- Web applications are not present in our own computer but they are present in some other computer where our own computer and that computer are “Network Connected”
- For ex : Gmail , Facebook ,Twitter etc.
- There are two types of Standalone Applications
- Desktop Applications
- Mobile Applications
- Desktop Applications: As the name implies ,they are present in our “Desktop/Computer”.
- Mobile Applications: As the name implies ,they are present in our “Smart Phone”.

- Web Applications can be of 3 Tiered(Layered) or 2 tiered
- 3-Tier architecture application: accessed using “Web Browser”
    - Layer1 - Client
    - Layer2 - Server
    - Layer3 - DB
- 2-Tier architecture application: accessed using “Mobile App”
    - Layer1 - Client[DB maintained in Client's device].
    - Layer2 - Server
- Now a days users depend more on Mobile App.

Android architecture



Linux kernel

- It has Linux Version 2.6.x for core system services and thus android handles only “Kernel” portion in Linux
- A kernel is a central component of an operating system. It acts as an interface between the user applications and the hardware.
- The main tasks of the kernel are :
    - Process management
    - Device management
    - Memory management
    - Interrupt handling
    - I/O communication
    - File system etc

Android Runtime

- For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a byte code format designed especially for Android that's optimized for minimal memory footprint.

Core Libraries- Uses the JAVA Programming Language

Dalvik Virtual Machine

- Android Application operates in its own process with the specific instance of the Dalvik virtual machine (DVM).
- The DalvikVM is Java based licenses free VM.
- It executes files in the Dalvik Executable (.dex) format.

Libraries

1. *Libc*: it is c standard lib.
2. *SSL*: Secure Socket Layer for security
3. *SGL*: 2D picture engine where SGL is "Scalable Graphics Library"
4. *OpenGL|ES*: 3D image engine
5. *Media Framework*: essential part of Android multi-media
6. *SQLite*: Embedded database
7. *Web Kit*: Kernel of web browser
8. *Free Type*: Bitmap and Vector
9. *Surface Manager*: Manage different windows for different applications

Application framework

- A rich and extensible View System you can use to build an app's UI, including lists, grids, text boxes, buttons, and even an embeddable web browser
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files
- A Notification Manager that enables all apps to display custom alerts in the status bar
- An Activity Manager that manages the lifecycle of apps and provides a common navigation back stack
- Content Providers that enable apps to access data from other apps, such as the Contacts app, or to share their own data

Application

- Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more.

Android Studio project structure

Android View

manifests/

- AndroidManifest.xml is one of the most important file in the Android project structure.
- It contains information of the package, including components of the application
- It is responsible to protect the application to access any protected parts by providing the permissions
- It also declares the android api that the application is going to use

java/

- The java folder contains the Java source code files of the application organized into packages.

res/

- Res folder is where all the external resources for the application are stored.
- **Drawable:** The folders are to provide alternative image resources to specific screen configurations.
- **Layout:** It contains XML files that define the User Interface of the application
- **Mipmap:** The mipmap folder is used for placing the app icons only..
- **Values :** XML files that define simple values such as strings, arrays, integers, dimensions, colors, styles etc. are placed in this folder

Gradle Scripts

- Gradle scripts are used to automate tasks.
- For the most part, Android Studio performs application builds in the background without any intervention from the developer.
- This build process is handled using the Gradle system,

Project View

.idea/

- In this folder the project specific metadata is stored by Android Studio.

Project Module(app)

- This is the actual project folder where the application code resides. The application folder has following sub directories

- **build** : This has all the complete output of the make process i.e. classes.dex, compiled classes and resources, etc. The important part is that the R.java is found here under build/generated/source/r/…/R.java
- **libs** : This is a commonly seen folder in eclipse and android studio, which optionally can hold the libraries or .jar files
- **src** : The src folder can have both application code and android unit test script. You will find two folders named “androidTest” and “main” correspond to src folder.
- **Gradle**: This is where the gradle build systems jar wrapper is found. This jar is how Android Studio communicates with gradle installed in Windows/MAC.

External Libraries

- This is not actually a folder but a place where Referenced Libraries and information on targeted platform SDK are shown.

What is R.java?

- 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.
- On creating 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.

Build Process

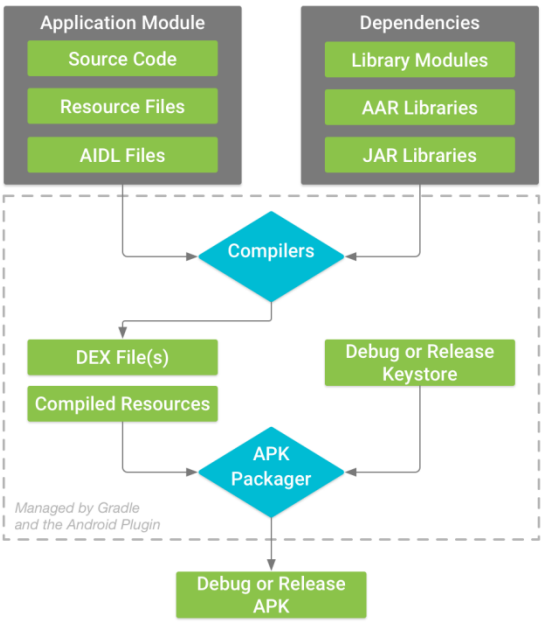
AIDL
Android Interface Defininatoin Language

AAR
Android Archive

JAR
Java Archive

DEX
Dalvik Executable

APK
Android Application Package



The build process for a typical Android app module is follows these general steps:

- The compilers convert your source code into DEX (Dalvik Executable) files, which include the bytecode that runs on Android devices, and everything else into compiled resources.
- The APK Packager combines the DEX files and compiled resources into a single APK. Before your app can be installed and deployed onto an Android device, the APK must be signed.
- The APK Packager signs your APK using either the debug or release keystore:
    - If you are building a debug version of your app, that is, an app you intend only for testing and profiling, the packager signs your app with the debug keystore.
    - Android Studio automatically configures new projects with a debug keystore.
    - If you are building a release version of your app that you intend to release externally, the packager signs your app with the release keystore.
- Before generating your final APK, the packager uses the zipalign tool to optimize your app to use less memory when running on a device.
- At the end of the build process, you have either a debug APK or release APK of your app that you can use to deploy, test, or release to external users.