SEC: Android Programming

Introduction

Mahesh Kumar
(maheshkumar@andc.du.ac.in)

Course Web Page
(www.mkbhandari.com/mkwiki)

Outline

History of Android

- From its inaugural release to today, Android has transformed ***visually, conceptually*** *and* ***functionally.***

- Let's understand the android history in a sequence.

  → Initially, *Andy Rubin* founded *Android Incorporation* in *Palo Alto, California, United States* in October, 2003.

  → In 17th August 2005, Google acquired *Android Incorporation*. Since then, it is in the subsidiary of *Google Incorporation*.

  → The key employees of *Android Incorporation* are *Andy Rubin, Rich Miner, Chris White* and *Nick Sears*.

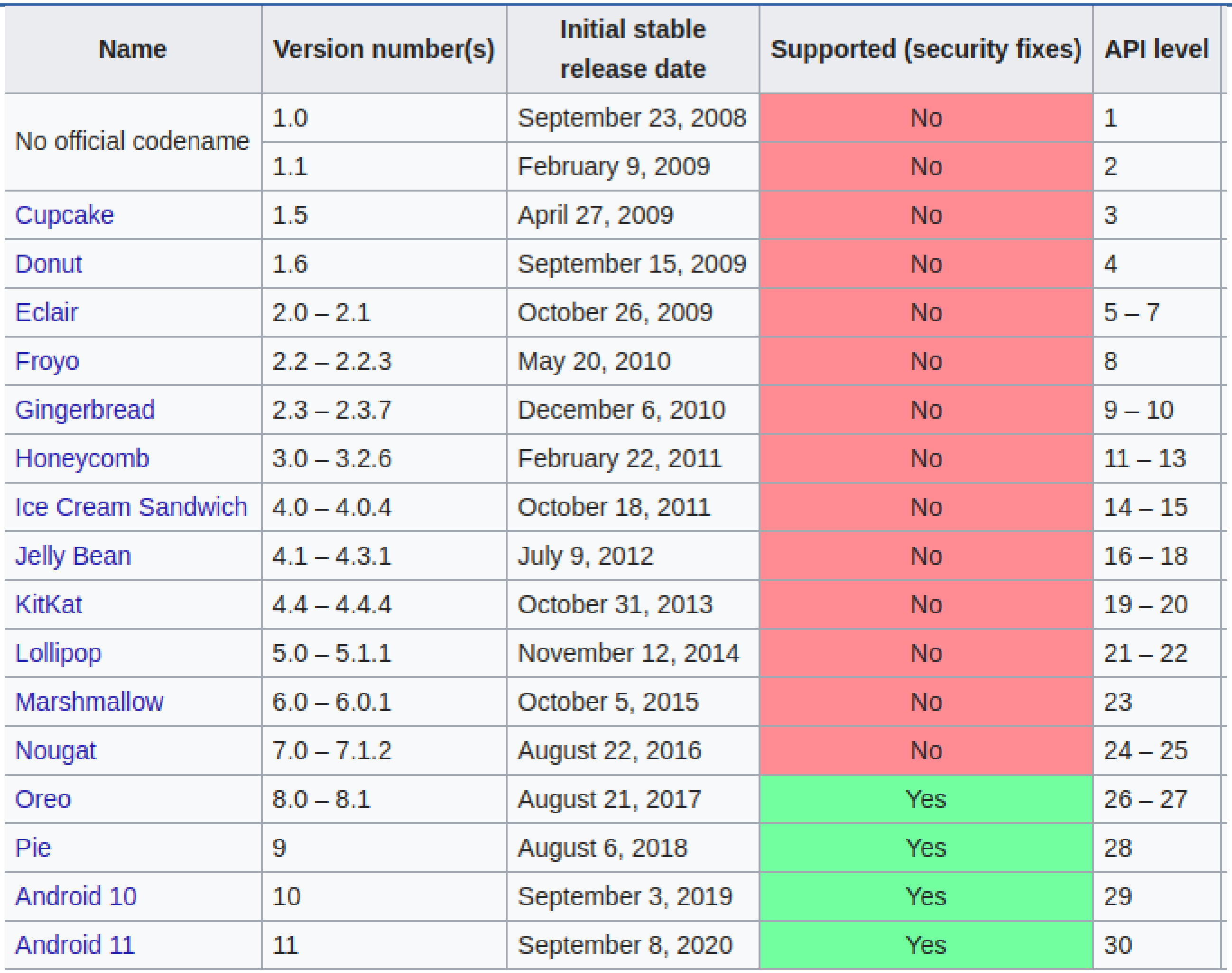
  → Originally intended for *camera* but shifted to *smart phones* later because of low market for camera only.

History of Android

- → Android is the nick name of Andy Rubin given by coworkers because of his love to robots.
- → In 2007, Google announces the development of *Android OS*.
- → In 2008, HTC launched the first *Android mobile* (T-Mobile G1).

- Open Handset Alliance(OHA)
  - → It's a consortium of 84 companies such as google, samsung, AKM, synaptics, KDDI, Garmin, Teleca, Ebay, Intel etc.
  - → It was established on 5th November, 2007, led by Google. It is committed to advance open standards, provide services and deploy handsets using the Android Platform.

History of Android – Versions, Codename, API Level



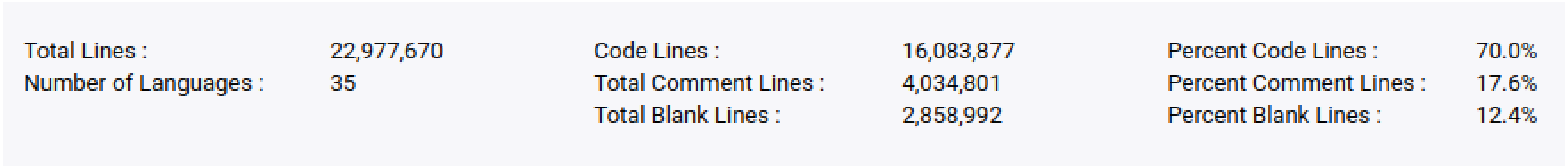
Name	Version number(s)	Initial stable release date	Supported (security fixes)	API level
No official codename	1.0	September 23, 2008	No	1
	1.1	February 9, 2009	No	2
Cupcake	1.5	April 27, 2009	No	3
Donut	1.6	September 15, 2009	No	4
Eclair	2.0 – 2.1	October 26, 2009	No	5 – 7
Froyo	2.2 – 2.2.3	May 20, 2010	No	8
Gingerbread	2.3 – 2.3.7	December 6, 2010	No	9 – 10
Honeycomb	3.0 – 3.2.6	February 22, 2011	No	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011	No	14 – 15
Jelly Bean	4.1 – 4.3.1	July 9, 2012	No	16 – 18
KitKat	4.4 – 4.4.4	October 31, 2013	No	19 – 20
Lollipop	5.0 – 5.1.1	November 12, 2014	No	21 – 22
Marshmallow	6.0 – 6.0.1	October 5, 2015	No	23
Nougat	7.0 – 7.1.2	August 22, 2016	No	24 – 25
Oreo	8.0 – 8.1	August 21, 2017	Yes	26 – 27
Pie	9	August 6, 2018	Yes	28
Android 10	10	September 3, 2019	Yes	29
Android 11	11	September 8, 2020	Yes	30

[2]

Introduction to Android Operating System

- **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 written in Java (UI), C (core), C++ and others.



Total Lines :	22,977,670	Code Lines :	16,083,877	Percent Code Lines :	70.0%
Number of Languages :	35	Total Comment Lines :	4,034,801	Percent Comment Lines :	17.6%
		Total Blank Lines :	2,858,992	Percent Blank Lines :	12.4%

- Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google.

Introduction to Android Operating System

- It was unveiled in November 2007, with the first commercial Android device launched in September 2008.

- It is free and open source software; its source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License.

- However most Android devices ship with additional proprietary software pre-installed, most notably Google Mobile Services (GMS) which includes core apps such as Google Chrome, the digital distribution platform Google Play and associated Google Play Services development platform.

Introduction to Android Operating System

- About 70 percent of Android smartphones run Google's ecosystem

- The "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use Android branding.

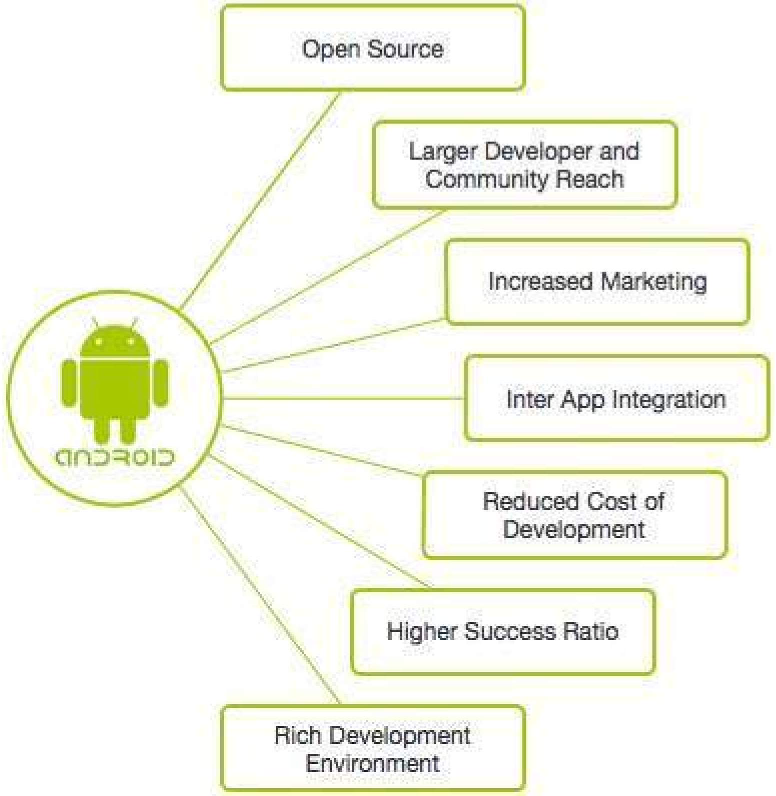
- The source code has been used to develop variants of Android on a range of other electronics, such as game consoles, digital cameras, portable media players, PCs and others, each with a specialized user interface.

- Some well known derivatives include Android TV for televisions and Wear OS for wearables, both developed by Google.

Introduction to Android Operating System

- Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013.
- As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system
- As of August 2020, the Google Play Store features over 3 million apps.
- The current stable version is Android 11, released on September 8, 2020.
- *The goal of android project is to create a successful real-world product that improves the mobile experience for end users.*

Why Android?


Open Source
Larger Developer and Community Reach
Increased Marketing
Inter App Integration
Reduced Cost of Development
Higher Success Ratio
Rich Development Environment
android

[3]

Features of Android

1. *Beautiful UI :*
   Android OS basic screen provides a beautiful and intuitive user interface.

2. *Connectivity :*
   GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

3. *Storage :*
   SQLite, a lightweight relational database, is used for data storage purposes.

4. *Media support :*
   H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.

5. *Messaging :*
   SMS and MMS.

Features of Android

6. *Web Browser :*

   Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3

7. *Multi-touch :*

   Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.

8. *Multi-tasking :*

   User can jump from one task to another and same time various application can run simultaneously.

9. *Resizable widgets :*

   Widgets are resizable, so users can expand them to show more content or shrink them to save space.

Features of Android

10 *Multi-Language :*

Supports single direction and bi-directional text.

11 *GCM :*

Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.

12 *Wi-Fi Direct :*

A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.

13 *Android Beam :*

A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.

Android Applications

- Many, to almost all, Android devices come with preinstalled Google apps including Gmail, Google Maps, Google Chrome, YouTube, Google Play Music, Google Play Movies & TV, and many more.
- Applications ("apps"), are written using the Android software development kit (SDK).
- Kotlin programming language was originally announced in May 2017, which replaced Java as Google's preferred language for Android app development in May 2019.
- Java is still supported (originally the only option for user-space programs, and is often mixed with Kotlin), as is C++.
- Java and/or other JVM languages, such as Kotlin, may be combined with C/ C++.

Android Applications

- Once developed, Android applications can be packaged easily and sold out either through a store such as:
  - → Google Play
  - → SlideME
  - → Opera Mobile Store
  - → Mobango
  - → F-droid
  - → Amazon Appstore

Categories of Android Applications

- There are many android applications in the market. The top categories are

Music	News	Multimedia
Sports	Lifestyle	Food & Drink
Travel	Weather	Books
Business	Reference	Navigation
Social Media	Utilities	Finance

[3]

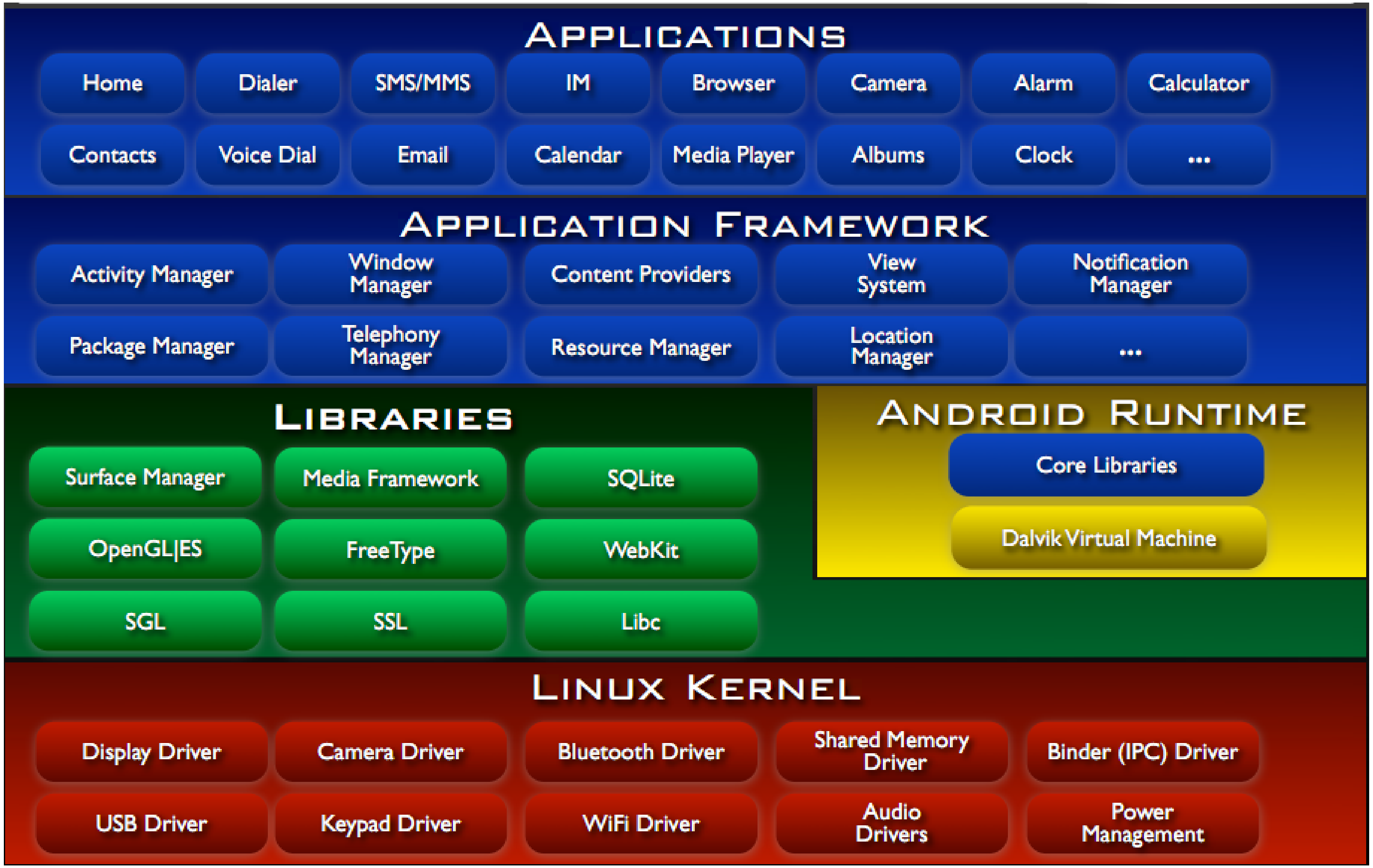
Android Architecture

- Android is an open source, Linux-based software stack created for a wide array of devices and form factors.

- Android architecture or Android software stack is categorized into five parts:

  1. *Linux Kernel*
  2. *Native Libraries (middleware)*
  3. *Android Runtime*
  4. *Application Framework (Java API Framework)*
  5. *Applications (System Apps)*

Android Architecture



[3]

Linux Kernel

- The foundation of the Android platform is the Linux kernel.
- It is the heart of android architecture that exists at the root of android architecture.
- Linux kernel is responsible for
  - → *Device drivers*
  - → *Power Management*
  - → *Memory Management*
  - → *Device Management*
  - → *Resource Access*

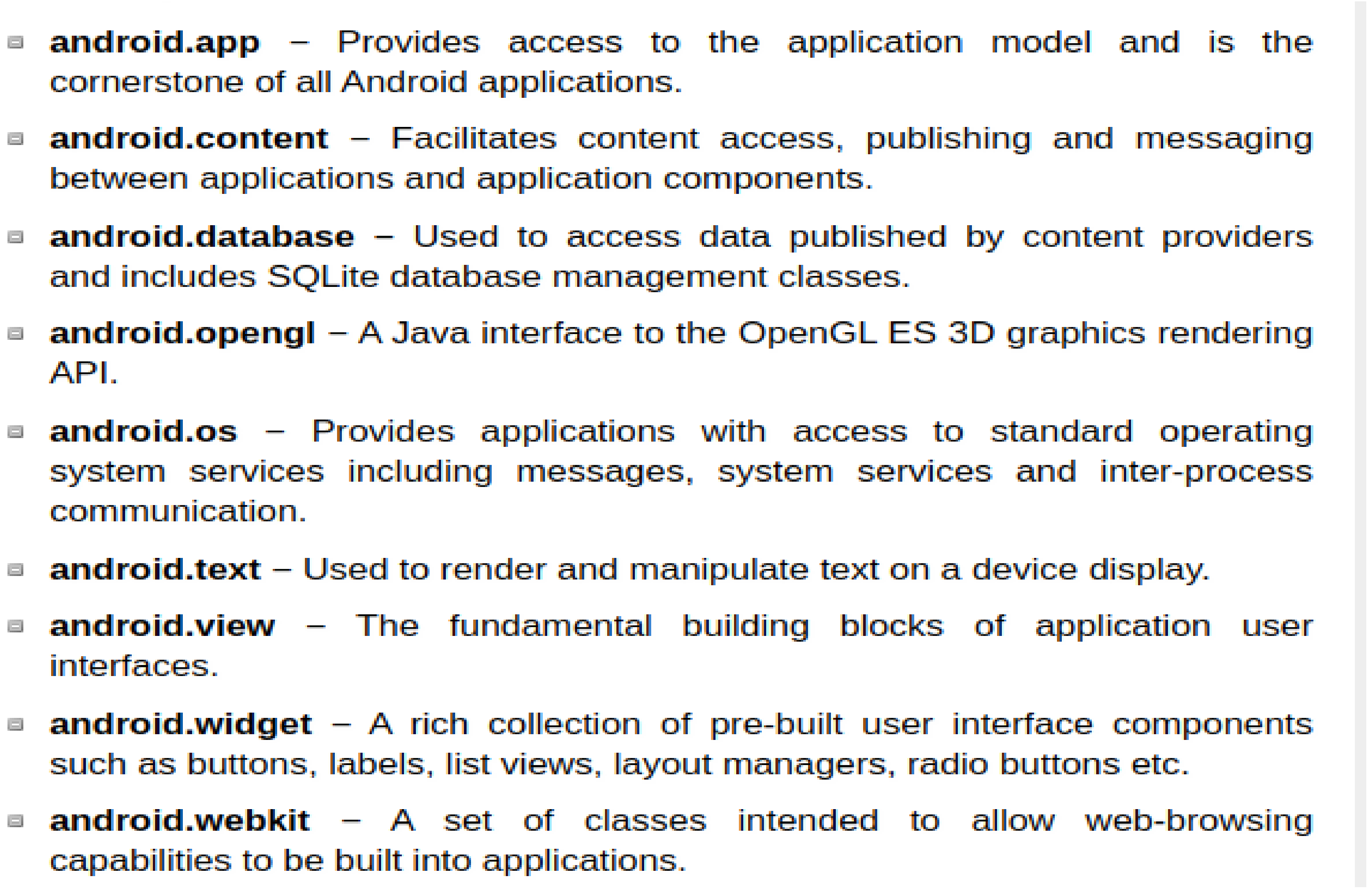
Native C/C++ Libraries

- On the top of Linux kernel, their are Native libraries such as :
  - → *Webkit - for browser support (open source web browser)*
  - → *OpenGL – graphics library used to produce 2D/3D computer graphics (cross-language, cross-platform)*
  - → *FreeType – for font support*
  - → *SQLite – for database*
  - → *Media – for playing and recording audio and video formats.*
  - → *Libc - C runtime library*
  - → *SSL – for internet security*

Android Runtime (ART)

- The third section of the architecture - *which is responsible to run Android application*.
- Available on the second layer from the bottom.
- ART provides a key component called Dalvik Virtual Machine (DVM), a kind of JVM specially designed and optimized for Android.
- DVM make use of Linux core features like memory management and multithreading which are integral part of Java language.
- DVM enables every Android application to run in its own process, with its own instance of DVM.
- In addition to DVM, ART also provides a set of core libraries that enables Android app developers to write/develop Android apps using standard Java Language

Android Runtime (ART) – Core Libraries



- **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.
- **android.webkit** – A set of classes intended to allow web-browsing capabilities to be built into applications.

Application (Java API) Framework

- On the top of Native libraries and Android runtime, there is Application framework

- Application framework includes Android API's such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers.

- Application framework provides many higher-level services to applications in the form of Java classes.

- App developers are allowed to make use of these services in their applications.

Application (Java API) Framework

- Key services are:

1. **Activity Manager:** Controls all aspects of the application life-cycle & activity stack.
2. **Content Providers:** Allows apps to publish & share data with other apps.
3. **Resource Manager:** Provides access to non-code embedded resources such as strings,color settings, and user interface layouts.
4. **Notification Manager:** Allows apps to display alerts & notifications to the users.
5. **View System:** An extensible set of views used to create application UI’s
6. **Location Manager:** Finds the device’s geographic locations.

Applications

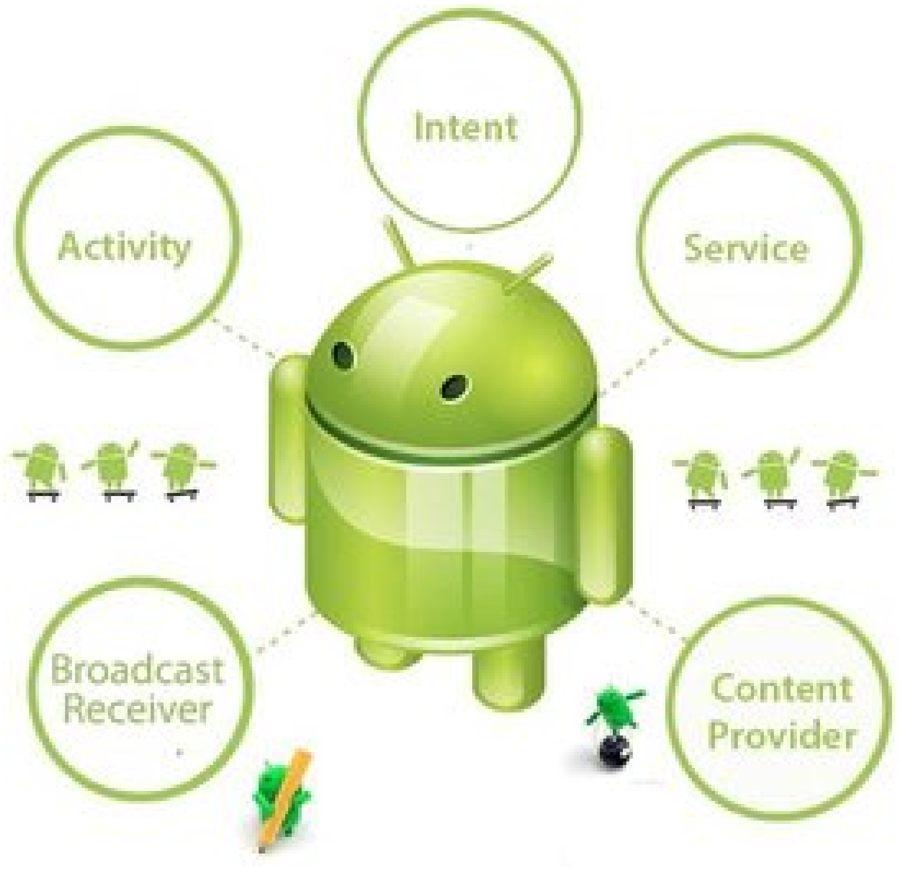
- On the top of android framework, there are applications.
- App developers can only develop applications for this layer.
- The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only.
- All applications such as home, contact, settings, games, browsers are using Application Framework that uses Android Runtime and Native Libraries.
- Android Runtime and Native Libraries are using Linux Kernel.

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 :

1. **Activities**
2. Views
3. Intents and **Broadcast Receivers**
4. **Services**
5. **Content Providers**
6. Fragments
7. AndroidManifest.xml

Android Core Building Blocks



[4]

Activity

- An activity is the entry point for interacting with the user. It represents a single screen with a user interface.
  - → ***For example:*** An email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails.
- Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others.
  - → As such, a different app can start any one of these activities if the email app allows it.
  - → ***For example***, a camera app can start the activity in the email app that composes new mail to allow the user to share a picture.

Activity

- An activity facilitates the following key interactions between system and app:
  - → Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.
  - → Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.
  - → Helping the app handle having its process killed so the user can return to activities with their previous state restored.
  - → Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows. (The most classic example here being share.)

Activity

- The first stepping stone in building an Android user app.
- *An activity is a class that represents a single screen. It is like a Frame in AWT*
- Generally, an Android app has more than one activity.
  - → *There is one "main" activity, and all other activities are "child" activities.*
- An activity is implemented as a subclass of class **Activity**.

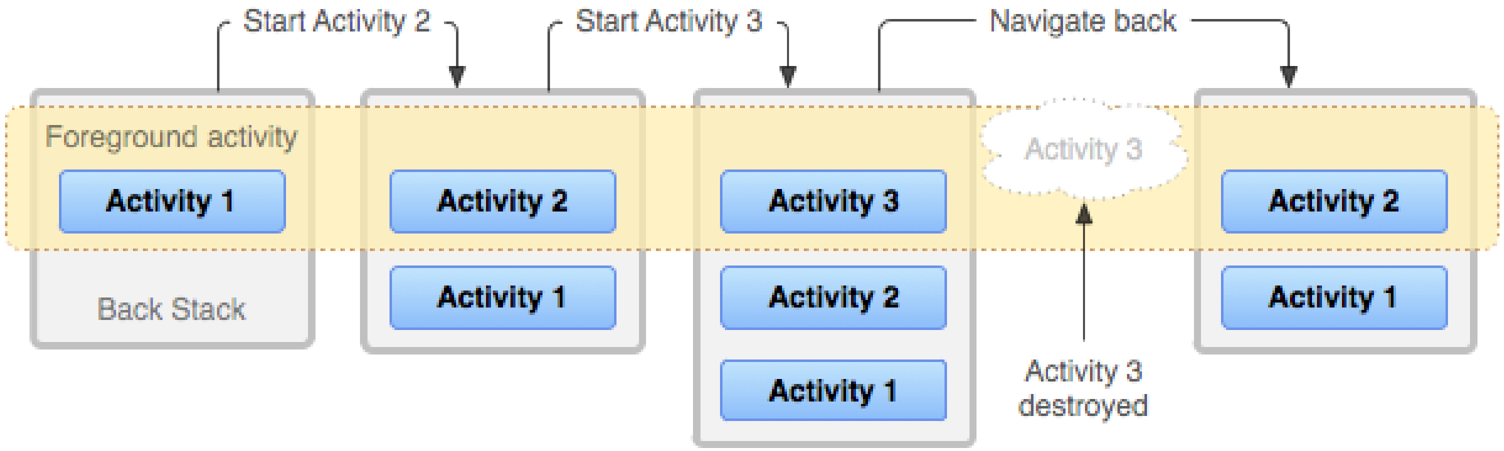
```
public class MainActivity extends Activity {

//do something

}
```

Activity

- A task is a collection of activities that users interact with when performing a certain job. *For example: Buying a product (mobile phone) from Amazon app.*
- The activities are arranged in a stack (the Back Stack) in the order in which each activity is opened.



A representation of how each new activity in a task adds an item to the back stack. When the user presses the **Back** button, the current activity is destroyed and the previous activity resumes. **[5]**

Activity Lifecycle

- As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle.

- The Activity class provides a number of callbacks that allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.

- Within the lifecycle callback methods, you can declare how your activity behaves when the user leaves and re-enters the activity.
  - → *For example, if you're building a streaming video player, you might pause the video and terminate the network connection when the user switches to another app. When the user returns, you can reconnect to the network and allow the user to resume the video from the same spot.*

Activity Lifecycle

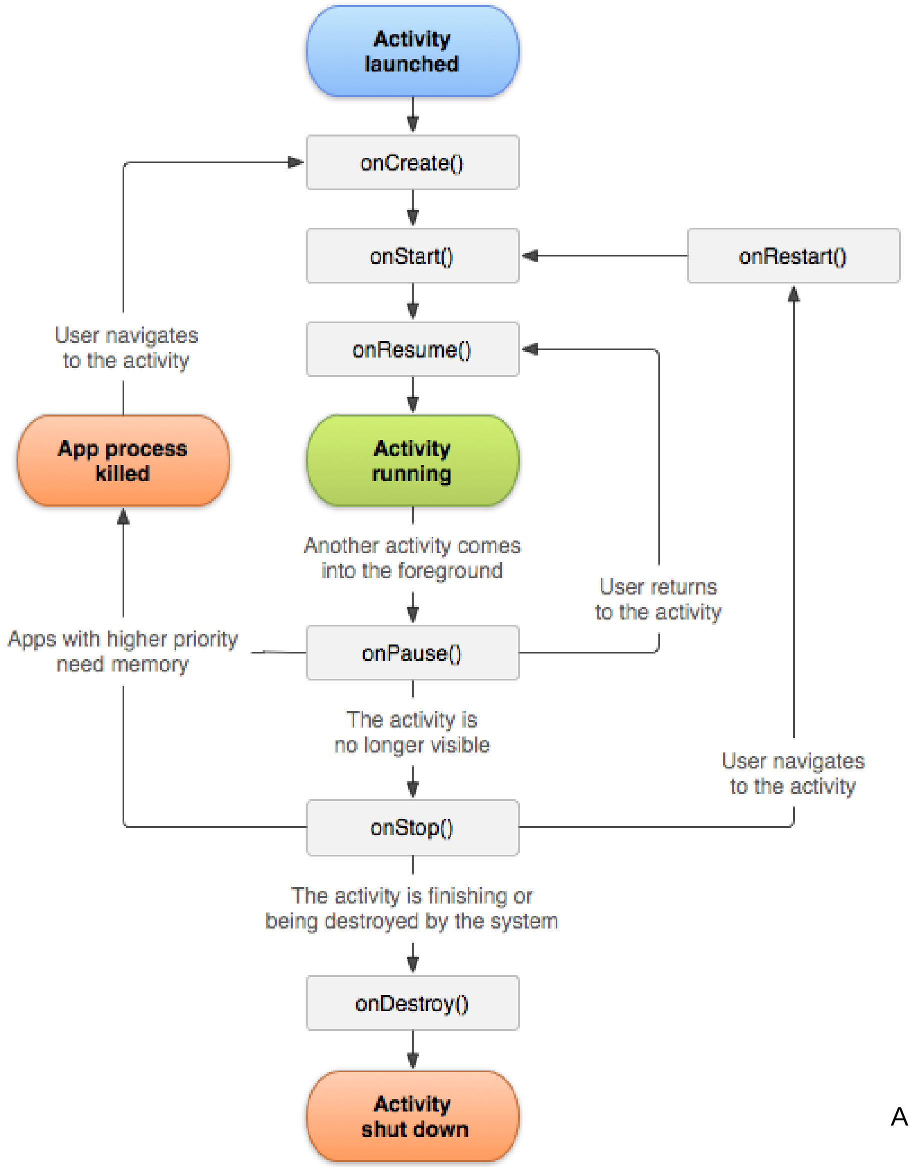
- Each callback allows you to perform specific work that's appropriate to a given change of state.
- Doing the right work at the right time and handling transitions properly make your app more robust and performant.
- For example, good implementation of the lifecycle callbacks can help ensure that your app avoids:
  - → *Crashing if the user receives a phone call or switches to another app while using your app.*
  - → *Consuming valuable system resources when the user is not actively using it.*
  - → *Losing the user's progress if they leave your app and return to it at a later time.*
  - → *Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.*

Activity Lifecycle

- In C, C++, or Java program starts from main( ) function.

- Very similar way, Android system initiates its program with in an Activity starting with a call on *onCreate( )* callback method.

- There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the Activity life cycle diagram (in next slide):


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

A simplified illustration of the activity lifecycle. **[5]**

Activity Lifecycle

- To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks:

  1. onCreate( ) **:** This is the first callback and called when the activity is first created.
  2. onStart( ) **:** This callback is called when the activity becomes visible to the user.
  3. onResume( ) **:** This is called when the user starts interacting with the application.
  4. onPause( ) **:** The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
  5. onStop( ) **:** This callback is called when the activity is no longer visible.
  6. onDestroy( ) **:** This callback is called before the activity is destroyed by the system.

- The system invokes each of these callbacks as an activity enters a new state.

Activity Lifecycle - Summary

- Open/Launch app – onCreate( ), onStart( ), onResume( )
- Minimize app – onPause( ), onStop( )
- Recent apps (Go back to app after minimize) – onRestart( ), onStart( ), onResume( )
- Close app - onPause( ), onStop( ), onDestroy( )

References

R Reference for this topic

1. **Book:** Android application development for java programmers. By James C. Sheusi. Publisher: Cengage Learning, 2013.
2. **Web:** https://en.wikipedia.org/wiki/Android_version_history
3. **Web:** https://www.tutorialspoint.com/android/android_overview.htm
4. **Web:** https://www.javatpoint.com/android-tutorial
5. **Web:** https://developer.android.com/guide/components/activities