Mobile App Development

Lec 1: Introduction

Ekarat Rattagan, PhD

Outline

1	Introduction and application fundamental
2	Layout and GUI widget I
3	Layout and GUI widget II
4	Activity
5	Intents + Preference
6	Saving data & files
7.	Saving database

Midterm

8	Concurrent I
9	Concurrent II
10	Multimedia I
11	Multimedia II
12	Networking I
13	Networking II
14	Jason
15.	Case study

Final Project

Outline

- กลางภาค 30%
- ปลายภาค 30%
- Project 30%
- เข้าเรียน 10%
- Quiz (option) 10%

Biography

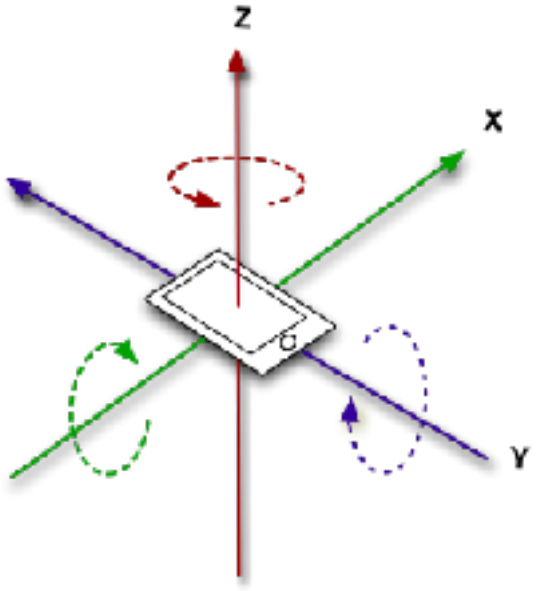
- **Name**: Ekarat Rattagan (เอกรัฐ รัฐกาญจน์)
- **Education**: Ph.D. (Electrical Engineering and Computer Science), NCTU, Taiwan.
- **Research**:
  - Mobile system and app technology
  - Video game technology
- **Published**:
  - "*Calibrating Parameters and Formulas for Process-level Energy Consumption Profiling in Smartphones*", Journal of Network and Computer Application, 2014.
  - "*Semi-online Power Estimation For Smartphone Hardware Components*", IEEE International Symposium on Industrial Embedded System(SIES), Siegen, Germany, June 8-10, 2015.
  - "*Symbolic Regression and Clustering for Power Consumption Estimation on Smartphone Hardware Subsystem*", Taiwan patent, 2015.
  - "*Wi-Fi Usage Monitoring and Power Management Policy for Smartphone Background Applications*", Management and Innovation Technology International Conference (MITicon), Bang-Saen, Thailand, 12-14 October 2016.
- **Tel**: 094-450-4027
- **Line id**: ajpok
- **E-mail**: pokekarat@gmail.com

Biography (Cont.)

- **More channels**
  - Linkedin: https://th.linkedin.com/in/ekarat-rattagan-478210100
  - ResearchGate: https://www.researchgate.net/profile/Ekarat_Rattagan
  - Dblp: http://dblp.uni-trier.de/pers/hd/r/Rattagan:Ekarat

Smartphones

Smartphone HW components



Gyroscope

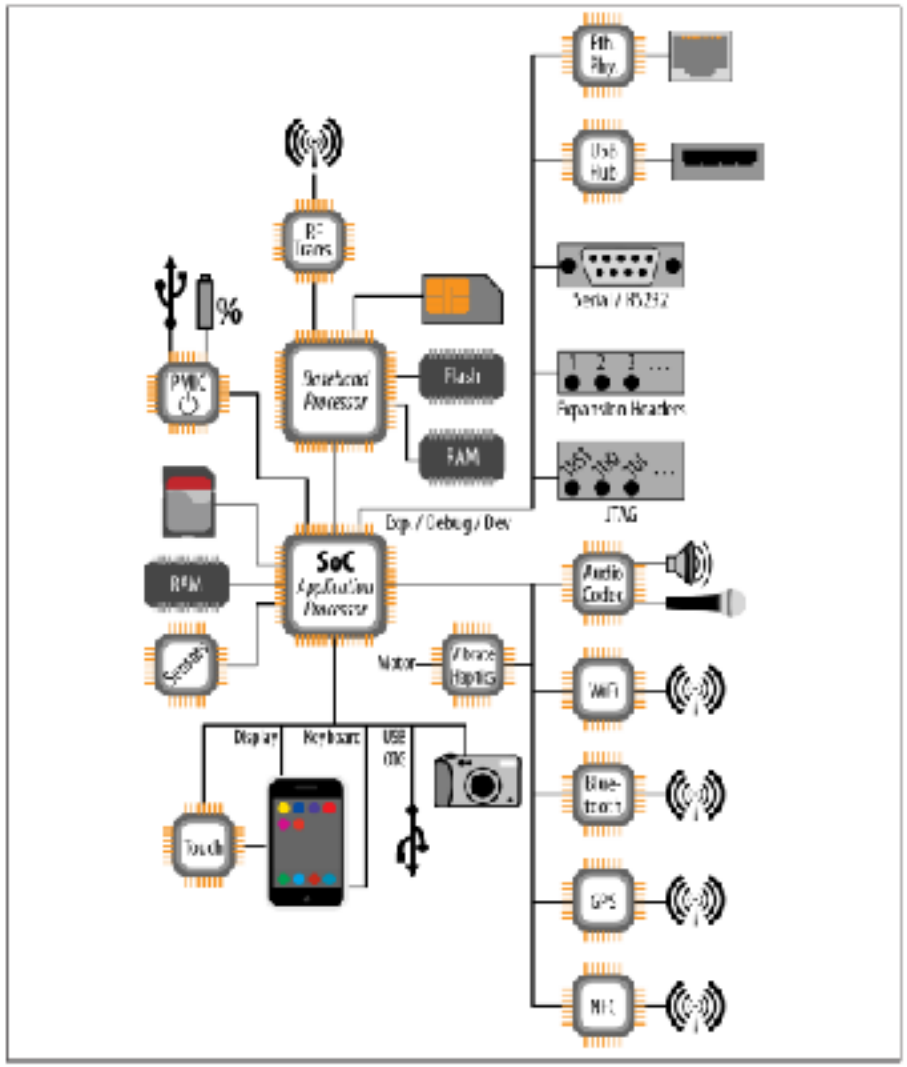


Figure 5-1. Typical system architecture block diagram

Embedded Android book

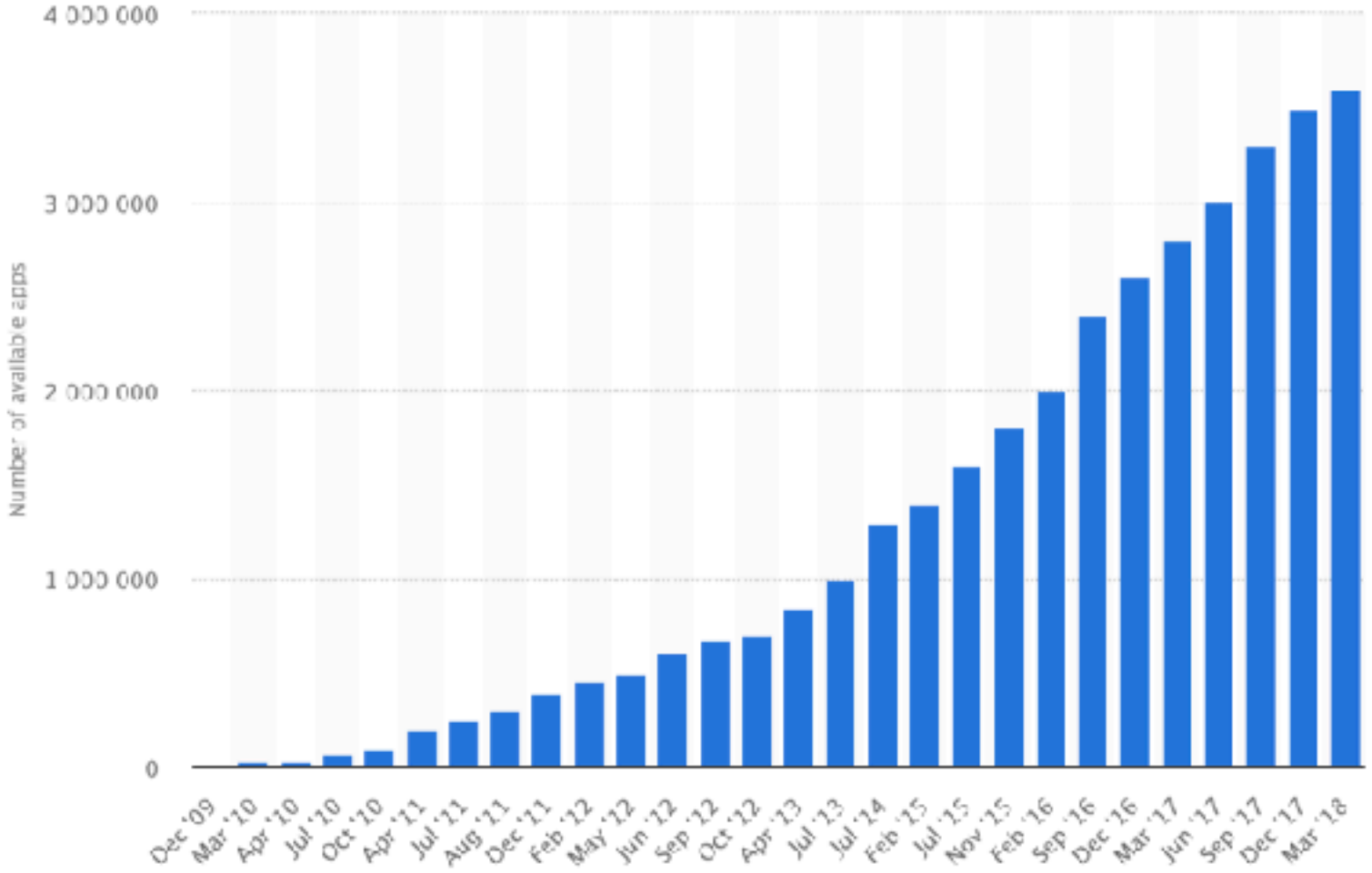
What is Android?

- Mobile operating system
  - Google purchased from Android, Inc. in 2005.
- Runs on phones, tablets, watches, TVs

http://www.thaiware.com/

http://www.mobipicker.com/

What is Android?

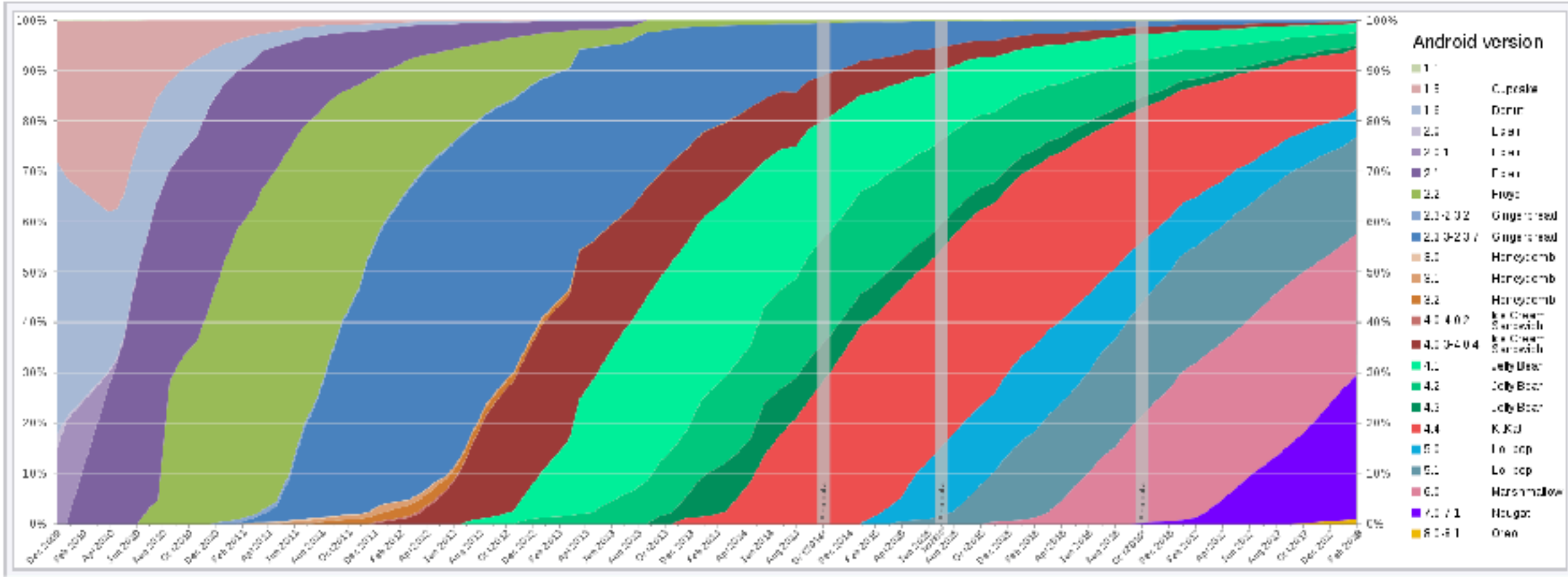


❑ 3.6 million apps published in Play Store (Mar. 2018) [statista].

Android version history & distribution

https://www.mobileappdaily.com/2017/11/15/android-history-things-that-you-have-never-heard

Android version history & distribution



https://en.wikipedia.org/wiki/Android_version_history

Android version history & distribution

Code name	Version number	Initial release date	API level	Security patches[1]
Beta[2]	1.1	February 9, 2009	2	Unsupported
Cupcake	1.5	April 27, 2009	3	Unsupported
Donut[3]	1.6	September 15, 2009	4	Unsupported
Eclair[4]	2.0 – 2.1	October 26, 2009	5 – 7	Unsupported
Froyo[5]	2.2 – 2.2.3	May 20, 2010	8	Unsupported
Gingerbread[6]	2.3 – 2.3.7	December 6, 2010	9 – 10	Unsupported
Honeycomb[7]	3.0 – 3.2.6	February 22, 2011	11 – 13	Unsupported
Ice Cream Sandwich[8]	4.0 – 4.0.4	October 18, 2011	14 – 15	Unsupported
Jelly Bean[9]	4.1 – 4.3.1	July 9, 2012	16 – 18	Unsupported
KitKat[10]	4.4 – 4.4.4	October 31, 2013	19 – 20	Unsupported[11]
Lollipop[12]	5.0 – 5.1.1	November 12, 2014	21 – 22	Unsupported[13]
Marshmallow[14]	6.0 – 6.0.1	October 5, 2015	23	Supported
Nougat[15]	7.0 – 7.1.2	August 22, 2016	24 – 25	Supported
Oreo[16]	8.0 – 8.1	August 21, 2017	26 – 27	Supported
Android P[17]	9	June 6, 2018 (beta 2)	28	In Beta
Android Q	10?			Developer preview; not yet supported
Legend: Old version	Older version, still supported	**Latest version**	Latest preview version	Future release				

https://en.wikipedia.org/wiki/Android_version_history

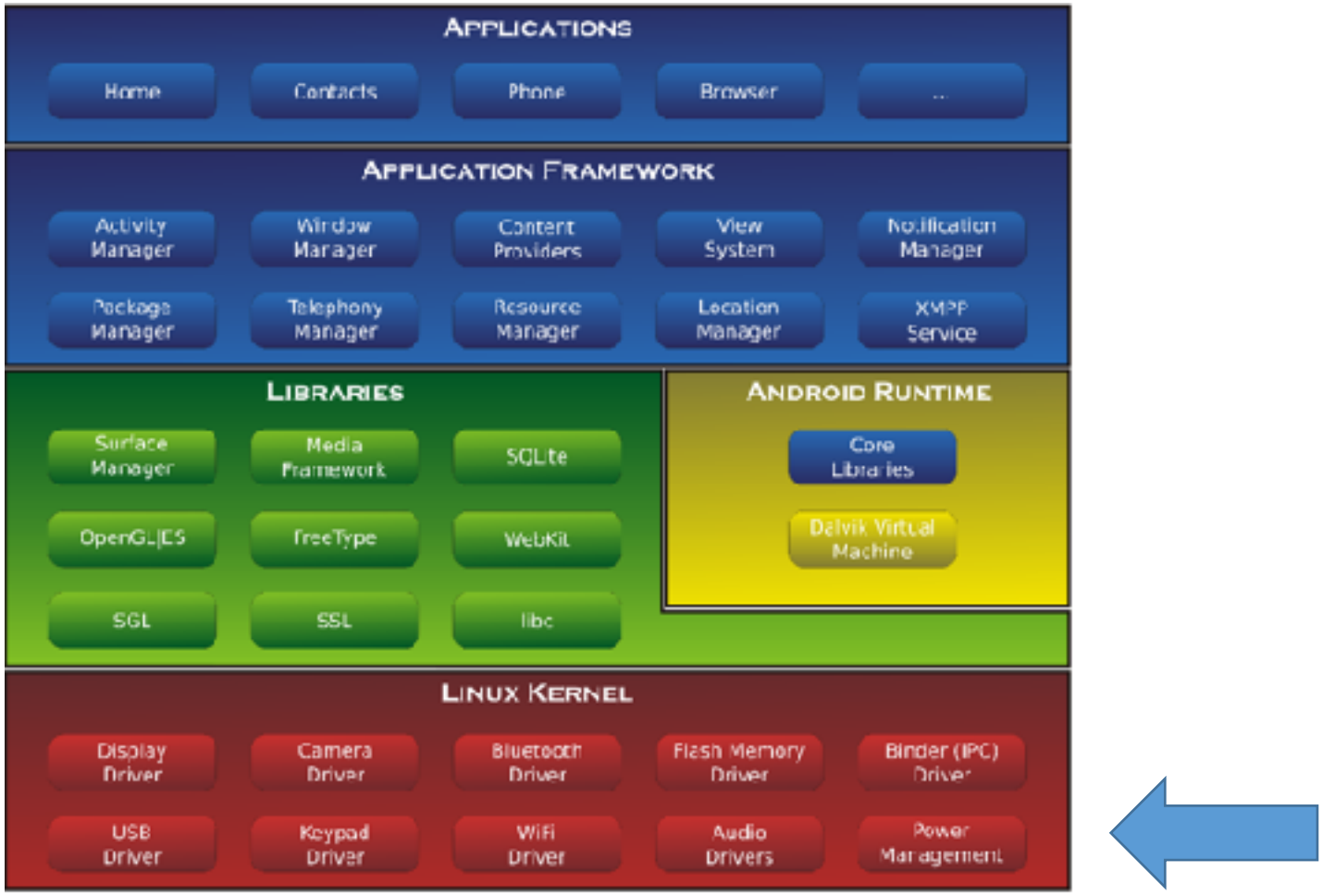
Android platform

- Based on Java and Linux.
- Open source codes
  - Easier to customize, license, etc.
- software stack for mobile devices:
  Operating system, middleware & key applications
- Use Android SDK to create applications
  Libraries & development tools
- Lots of documentation
  http://developer.android.com/

The Android Architecture

More IT, CS

More EE, CE

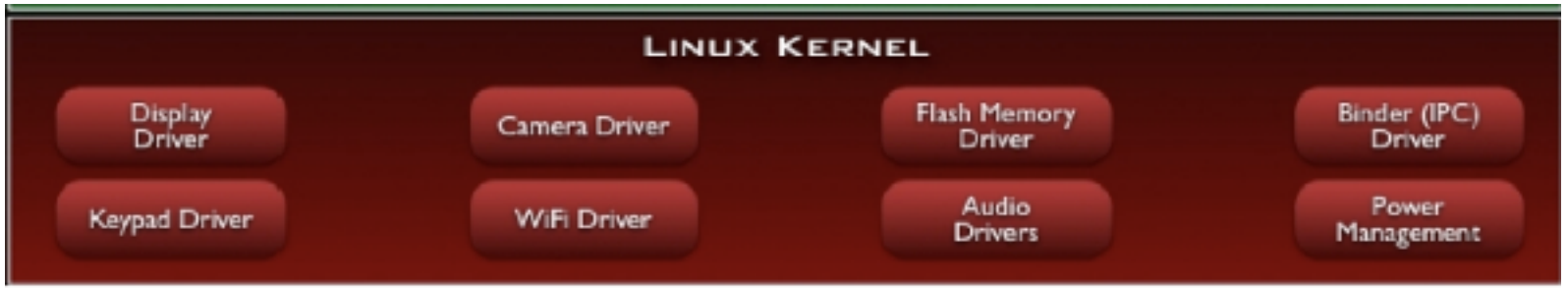


http://developer.android.com/guide/basics/what--is--android.html

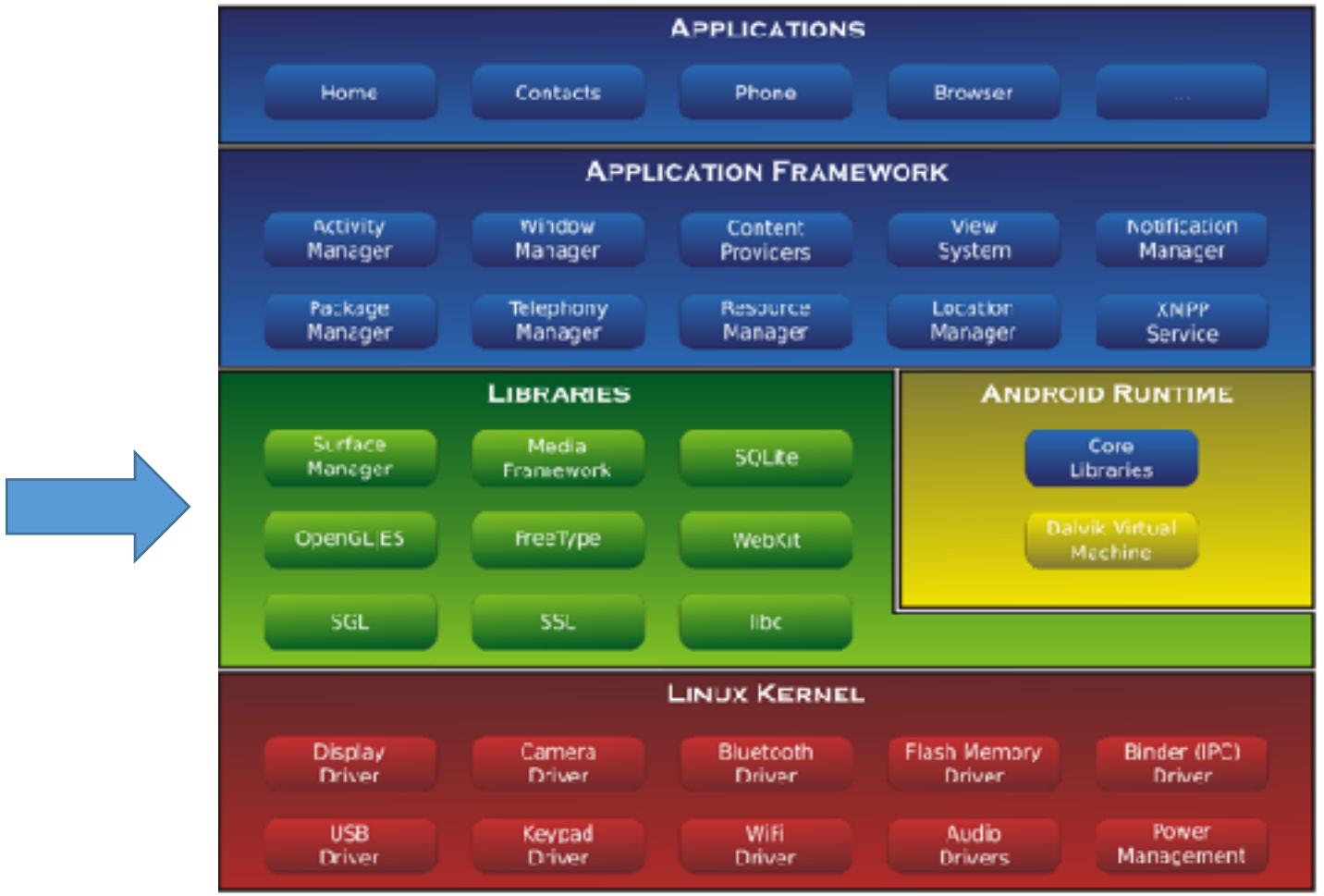
Linux Kernel Layer

Abstraction layer between HW & SW

- Memory & process management
- Network stack
- Device driver model



Library layer



http://developer.android.com/guide/basics/what--is--android.html

Library layer: Native Libraries

C/C++ libraries

- System C library
  bionic libc
- Surface Manager
  Display management
- Media Framework
  Audio/video
- Webkit
  Web browser engine
- OpenGL ES, SGL
  Graphics engines
- SQLite
  Relational database engine
- SSL
  Secure Socket Layer

Library layer: Android Runtime

http://developer.android.com/guide/basics/what--is--android.html

Library layer: Android Runtime

Support services for executing applications

- Core libraries
- Dalvik Virtual Machine (DVM)

Library layer:
Android Runtime - Core Libraries

Core libraries

- Doesn’t include all standard Java SDK classes
- Android.*
- Java.*, javax.*
- Junit.*
- Org.apache.*, org.json.*, org.xml.*

Library layer:
Android Runtime - DVM(1/7)

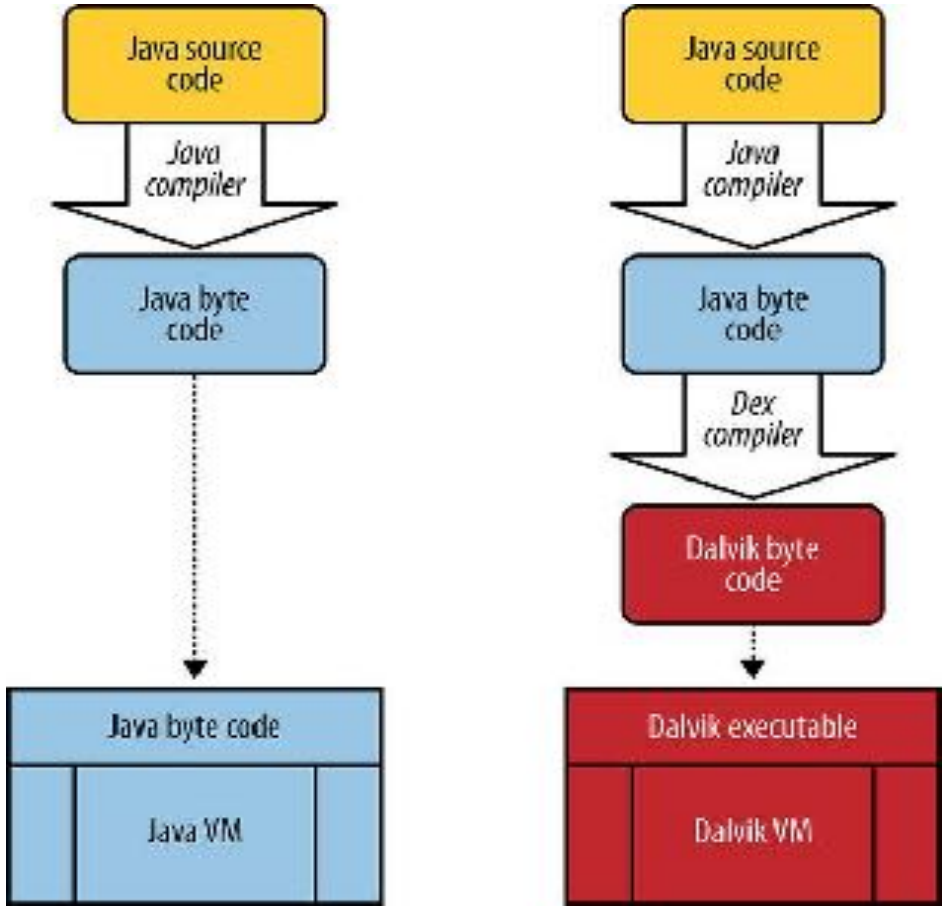
DVM designed to run on a handheld device

- Slow CPU
- Little RAM
- Limited battery life

Library layer: Android Runtime - DVM(2/7)

Apps typically wrote in Java

- Do not run in a standard Java virtual machine
- dx program transforms java classes into .dex formatted bytecodes
- Bytecodes executed in DVM
- Applications typically run in their own processes, inside their own instance of the DVM.



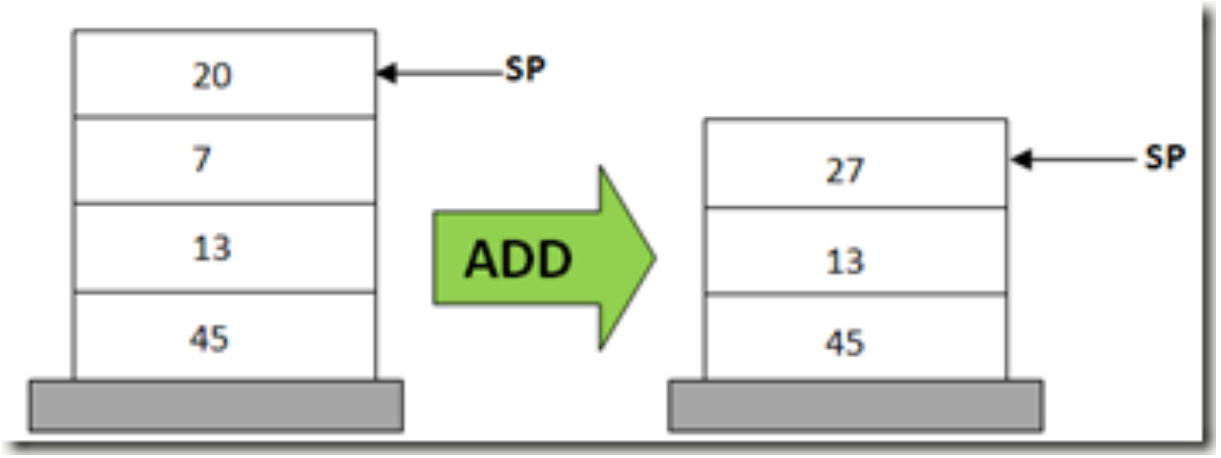
Library layer:
Android Runtime - DVM(3/7)

- Memory
  - One .dex file for multiple classes
  - Modified garbage collection to improve memory sharing

- CPU
  - Optimization applied at installation time
  - Register-based, rather than stack-based

Library layer:
Android Runtime - DVM(4/7)

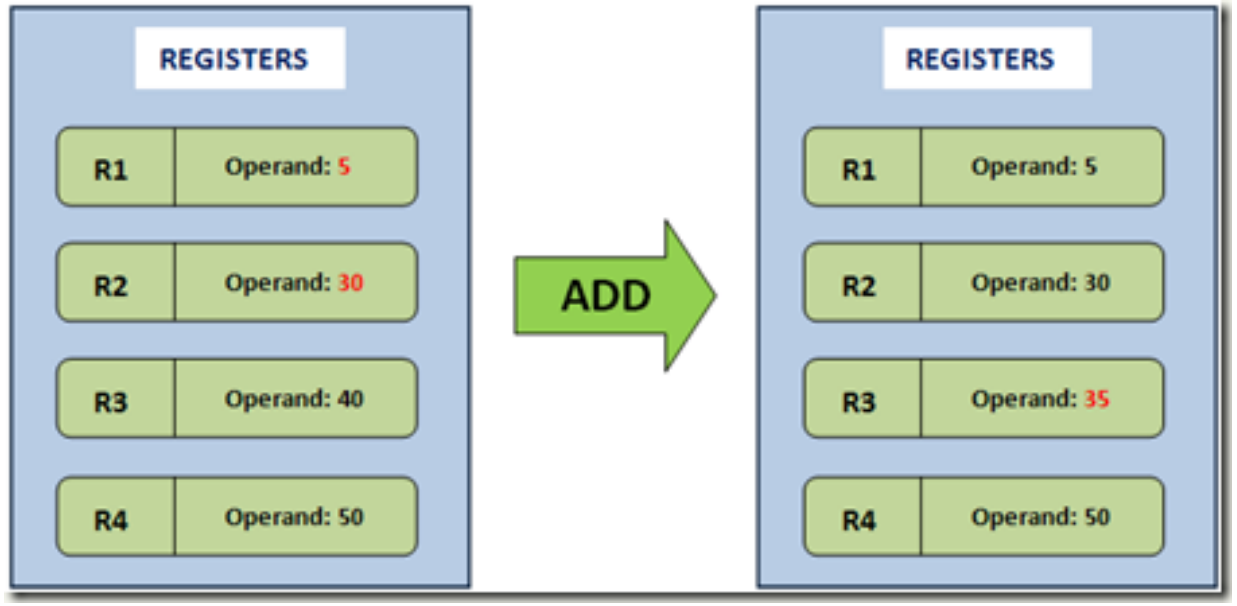
Stack-based



1.POP 20
2.POP 7
3.ADD 20, 7, result
4.PUSH result

Library layer:
Android Runtime - DVM(5/7)

Register-based



1. ADD R1, R2, R3;

Add contents of R1 and R2, store result in R3

Library layer: Android Runtime - DVM(6/7)

Why register-based

❑Expected benefits over stack-based VMs

- Avoids slow instruction dispatch
- Avoids unnecessary memory accesses
- More efficient instruction stream

Library layer: Android Runtime - DVM(7/7)

Example

```
public static long sumArray  (int[ ] arr)
{
    long sum = 0;
    for(int i:arr)
    {
        sum += i;
    }
    return sum;
}
```

Java Bytecode (Stack-based)

```
0: lconst_0
1: lstore_1
2: aload_0
3: astore_3
4: aload_3
5: arraylength
6: istore      4
8: iconst_0
9: istore      5
11:iload       5
13:iload       4
15:if_icmpge   36
18:aload_3
19:iload       5
21:iaload
22:istore      6
24:lload_1
25:iload       6
27:i2l
28:ladd
29:lstore_1
30:iinc        5, 1
33:goto        11
36:lload_1
37:lreturn
```

% javap –c ClassName

Dex Bytecode (Register-based)

```
0000: const-wide/16 v0, #long 0 // #0000
0002: array-length v2, v8
0003: const/4 v3, #int 0 // #0
0004: move v7, v3
0005: move-wide v3, v0
0006: move v0, v7
0007: if-ge v0, v2, 0010 // +0009
0009: aget v1, v8, v0
000b: int-to-long v5, v1
000c: add-long/2addr v3, v5
000d: add-int/lit8 v0, v0, #int 1 // #01
000f: goto 0007 // -0008
0010: return-wide v3
```

% dexdump –d classes.dex

Register-based vs Stack-based VMs

- 30% fewer instructions
- 35% fewer code units
- 35% more bytes in the instruction stream

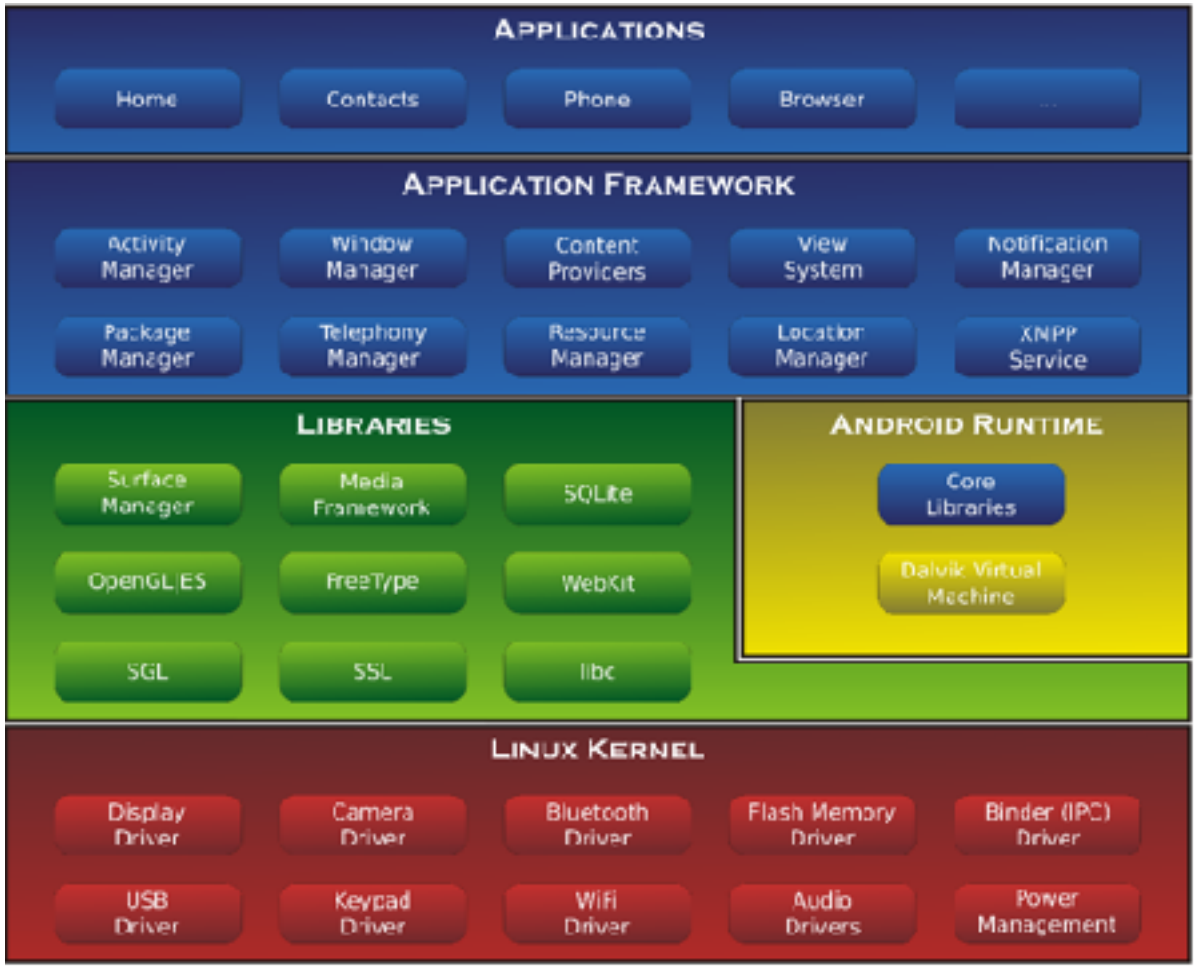
Application Framework Layer

http://developer.android.com/guide/basics/what--is--android.html

Application Framework Layer

- **Window Manager**
  Manages top-level window's look & behavior
- **View system**
  Lists, grids, buttons, etc.
- **Content providers**
  Inter-application data sharing
- **Activity manager**
  Application lifecycle

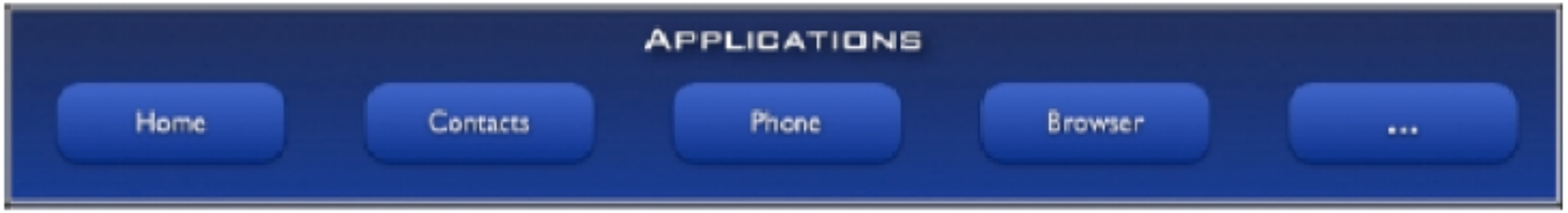
Application Layer



http://developer.android.com/guide/basics/what--is--android.html

Application Layer

- Standard apps
  - Home – main screen
  - Contacts - contacts database
  - Phone-dial phone numbers
  - Browser-view web pages
  - Email
- Installed apps (Google Play)



Building an App

Android requires that all APKs be digitally signed with a certificate, public key, before they can be installed.

See: developer.android.com/guide/developing/building/index.html

Running an App

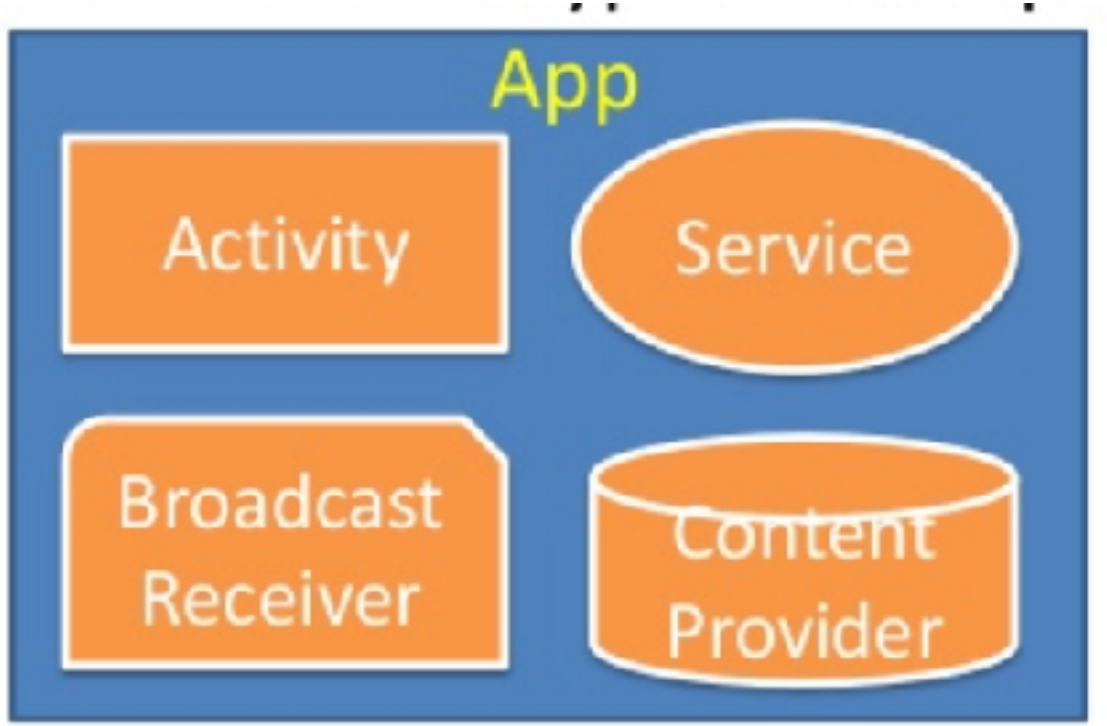
By default, each app

- Assigned a unique Linux user ID
- Executes in its own Linux process
- By default, each process runs its own DVM

Application Components

Main component classes include

1. Activities
2. Services
3. Broadcast receivers
4. Content providers



1. Activity

Primary class for interacting with users

- Usually implements a focused task
- E.g., calculator

2. Service

Runs in the background to perform long running or remote operations

- Does not have a visual user interface
- E.g., Music player

3. Broadcast Receiver

Component that listens for broadcast announcements (events)

- Does not have a visual user interface
- E.g., Messaging (SMS receipt)

4. Content Providers

Store & retrieve data across apps

- Uses database-style interface
- E.g., contacts

Conclusion

- Smartphone HW & SW
- Android architecture
- Android app components