Laboratorio di Applicazioni Mobili
Bachelor in Computer Science &
Computer Science for Management

University of Bologna

Android Architecture

Federico Montori
federico.montori2@unibo.it

Table of Contents

- Android History
- Android Data
- Android Architecture
    - Kernel
    - HAL
    - NDK
    - ART
    - APIs
    - System Apps
- Overview on App Components
- XML
- App Distribution

Android

Android is a Linux-based platform for mobile touchscreen devices …

- Operating System
- Middleware
- Applications
- Software Development Kit (SDK)

Which mobile devices?

SMARTPHONES

TABLETS

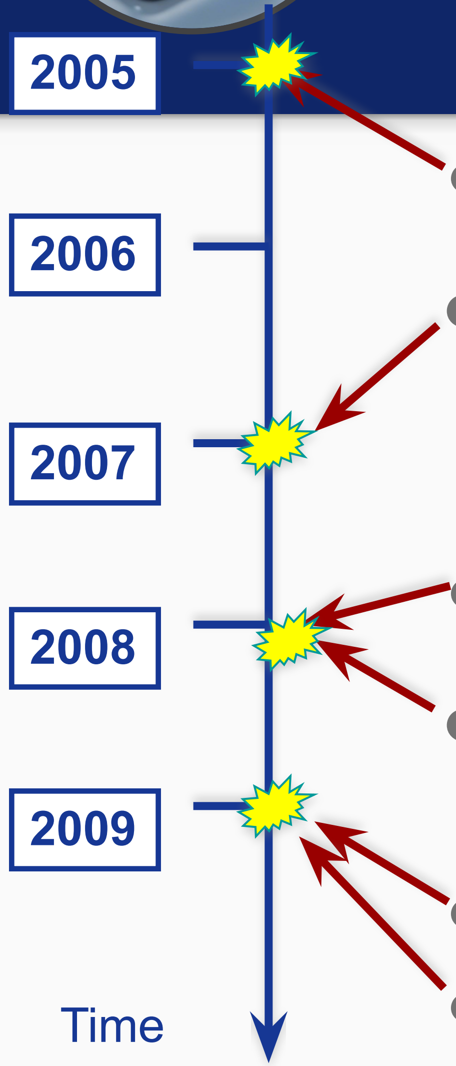
EREADERS

ANDROID TV

GOOGLE GLASSES

AUTOMOTIVE

Android



2005

2006

2007

2008

2009

Time

- **Google** buys Android from the **Android Inc.**
- Open Handset Alliance (**OHA**) created for open standards for mobile devices. Partners of OHA: Google, Motorola, Samsung, Vodafone, T-Mobile, etc.
- Android 1.0 Released
- The first Android smartphone: G1 HTC-Dream

  Few Google apps integrated directly with the OS
- Android **1.1** Released
- Android **1.5 (CupCake)** Released
  - External Apps
  - On-screen keyboard

Android

2008

2009

2010

2011

2012

Time

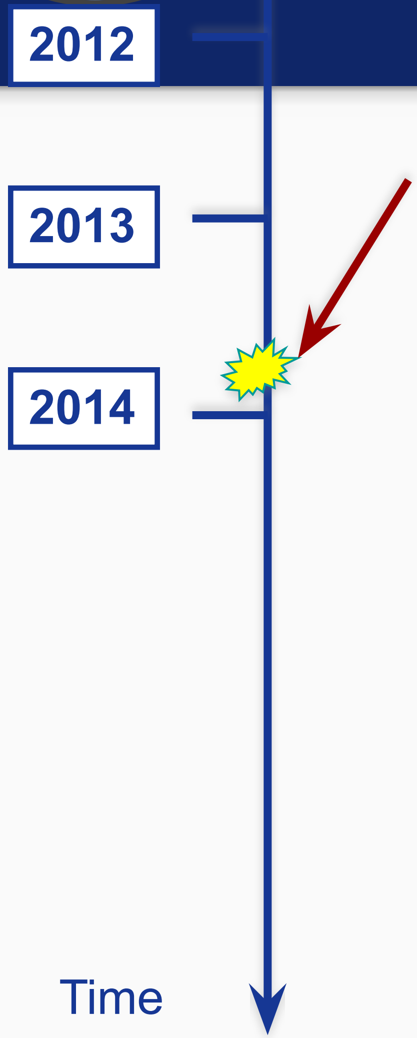
- Android **1.6 (Donut)** Released
  - Different screen sizes
  - CDMA
- Android **2.0 (Eclair)** Released
  - speech-to-text
  - pinch-to-zoom

- Android **2.2 (Froyo)** Released
  - bottom dock
  - voice actions
- Android **2.3 (Gingerbread)** Released
  - The “design release”
- Android **3.0 (Honeycomb)** Released

(First version for devices with larger screens such as tablets)

- Android **4.0 (Ice-Cream Sandwich)** Released. (It merges the 3.x tab centric design and the v2.x phone based design into a single version.)
  - swiping for dismissing
  - card appearance

Android



- Android **4.1** (**Jelly Bean**) Released
- Android **4.4** (**Kitkat**) Released

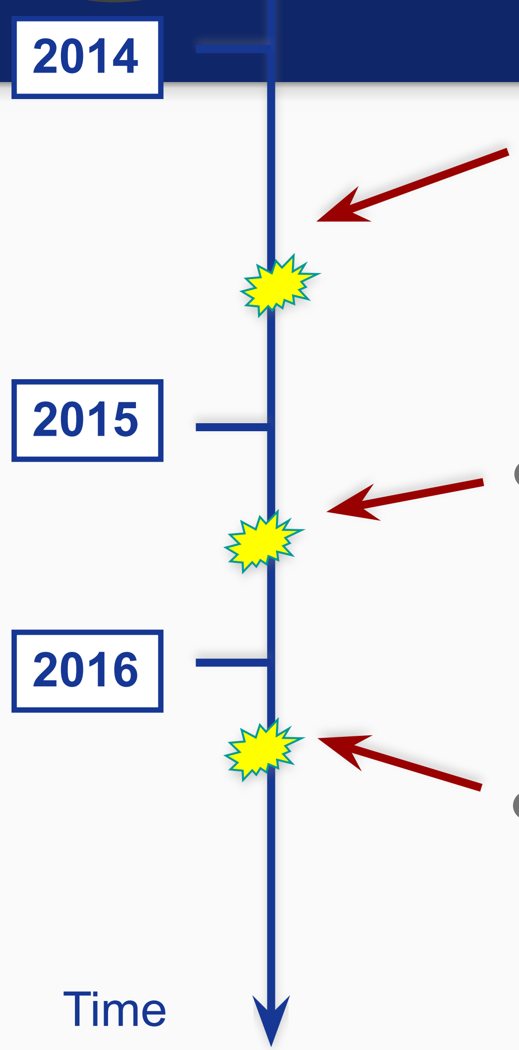
OK GOOGLE!
(but only when the screen is on…)

  - Wireless printing capability
  - Ability for applications to use "immersive mode”
  - Performance optimization
  - New experimental runtime virtual machine, ART…

API Level 19 (Android 4.4):

  - Support to new embedded sensors (e.g. STEP_DETECTOR)
  - Adaptive video playback functionalities
  - Read and write SMS and MMS messages
  - (managing default text messaging client)

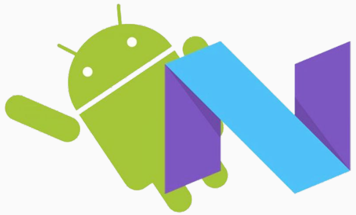
Android



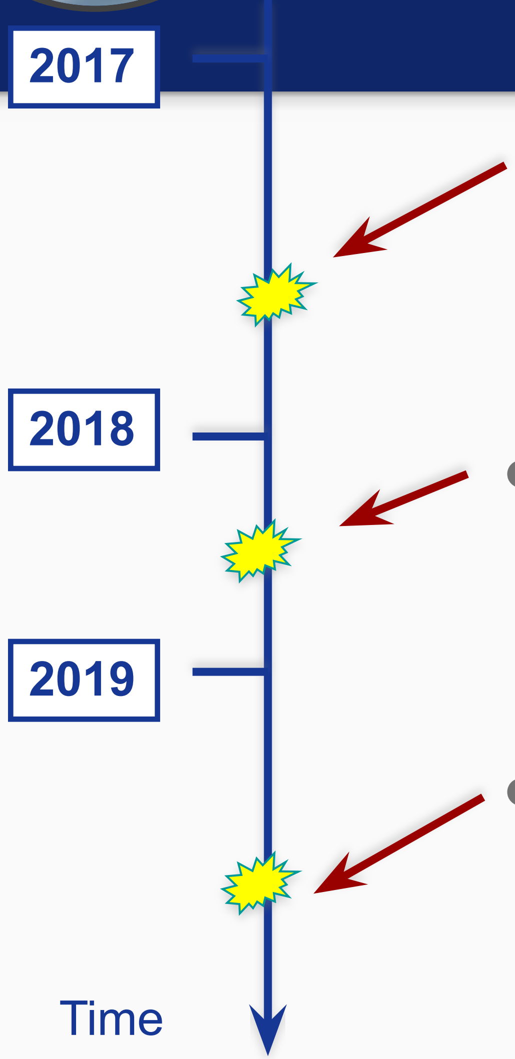
- Android **5.0** (**Lollipop**) Released
  - Material Design!
  - OK Google (the true one)
  - Lots of bugs...

- Android **6.0** (**Marshmallow**) Released
  - Fingerprint
  - USB-C
  - Runtime Permissions!!!

- Android **7.0** (**Nougat**) Released … pixel!
  - Google Assistant
  - Split screen, data saver...



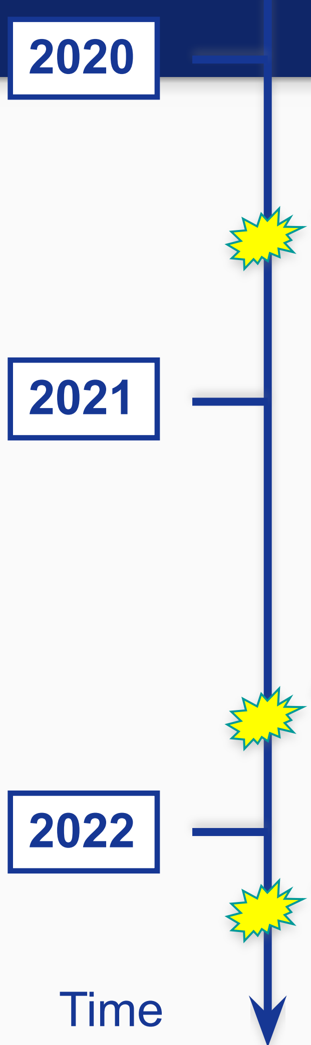
Android



- Android **8.0** (**Oreo**) Released
  - Notification Channels and Snooze
  - picture-in-picture
  - Android Apps on Chromebooks
- Android **9.0** (**Pie**) Released
  - Brightness & Battery Management
  - Hybrid gestures system
  - Privacy & Security
- Android **10.0** (**Q**) Released
  - Expanded permission
  - swipe-driven
  - ...No more sweets!!!

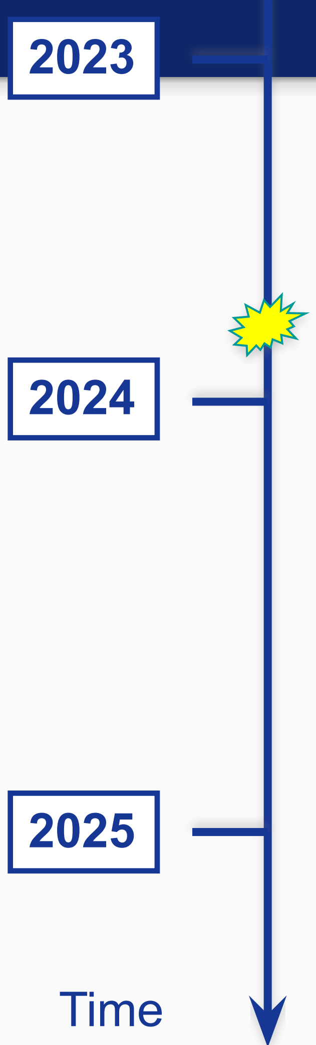
android

Android



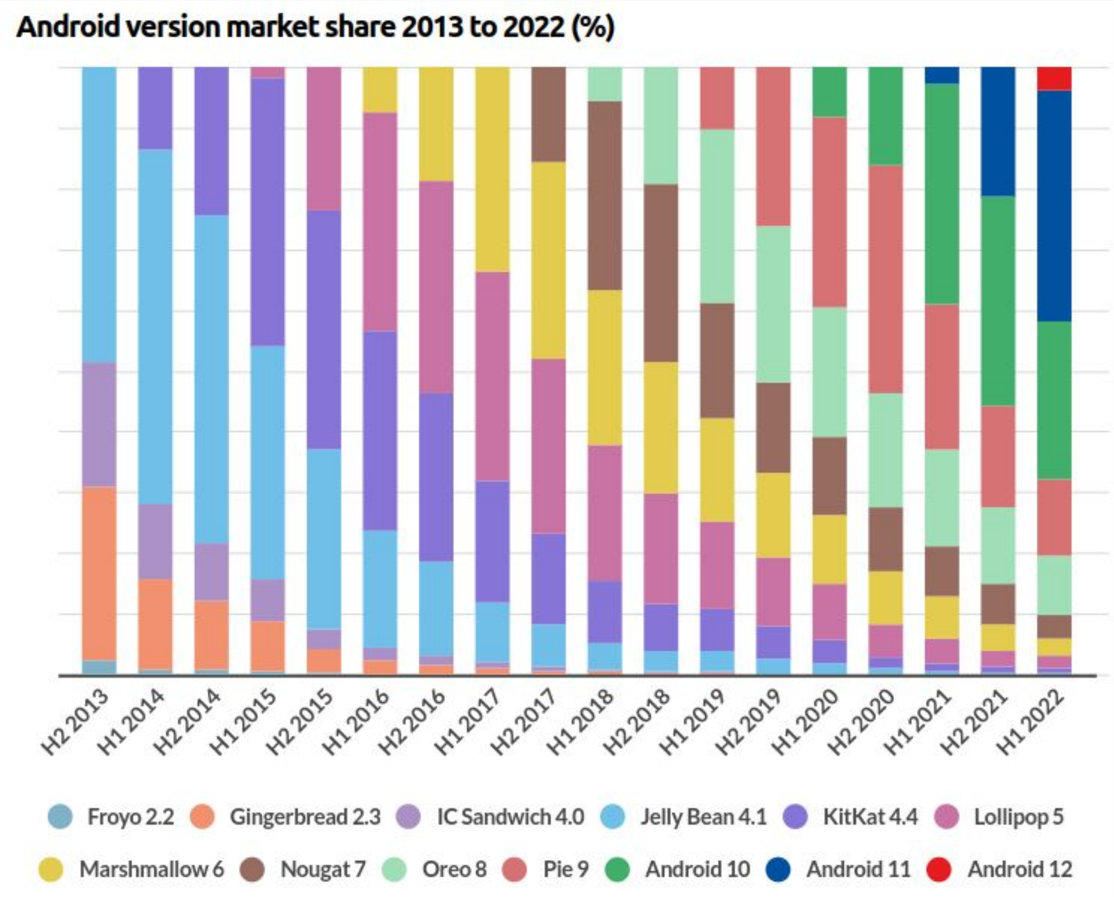
- In February Android **11.0** (**R**) Released
  - One-time permissions for temporary features (location, microphone and camera)
  - Exposure notification and privacy fixes
- In October Android **12.0** (**S**) Released
  - Location can be blurred even if required
  - Mostly optimizations and
  - graphical improvements
- Android **13.0** (**Tiramisu**) Released
  - per-app language personalization
  - permission for notification
  - gallery restricted access to apps

Android



- In October 2023 Android **14.0** (**Upside Down Cake**) Released
  - Mostly a UI improvement
  - Battery optimization

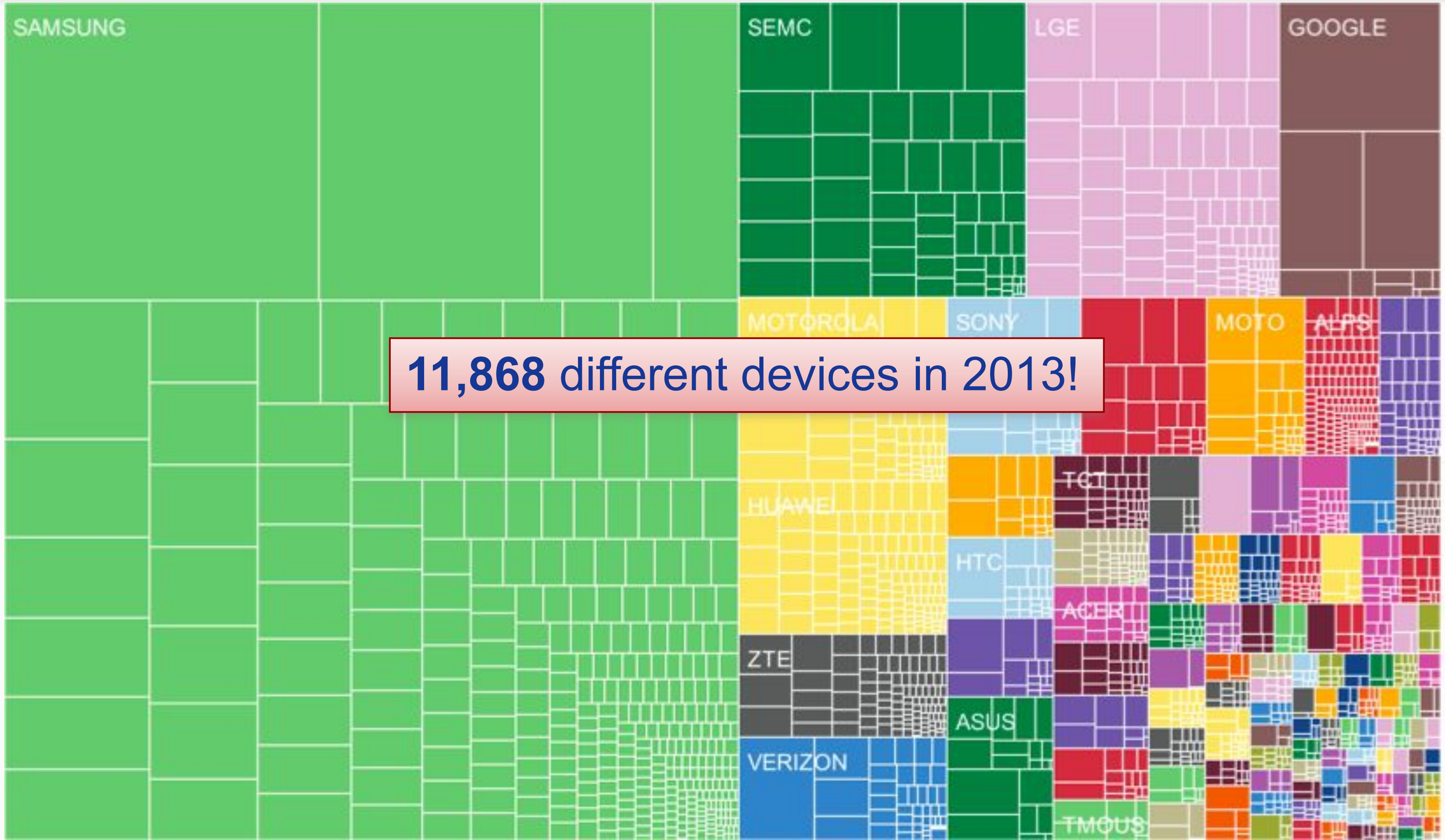
Android Data



Takeaway message:

There is a lot of versioning to deal with…

Android Data

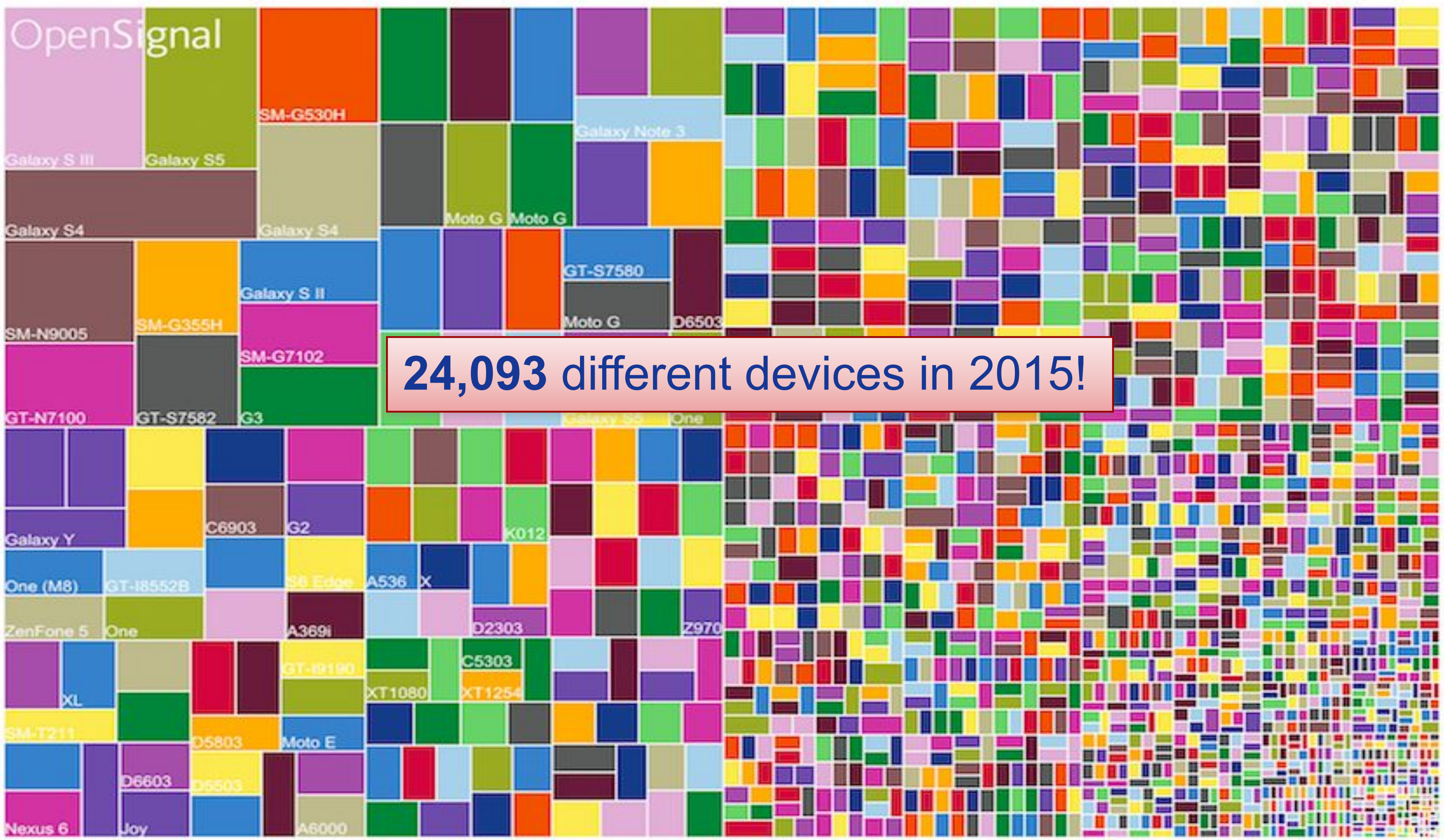


11,868 different devices in 2013!

Takeaway message:

There is a lot of different devices to take into account

Android Data

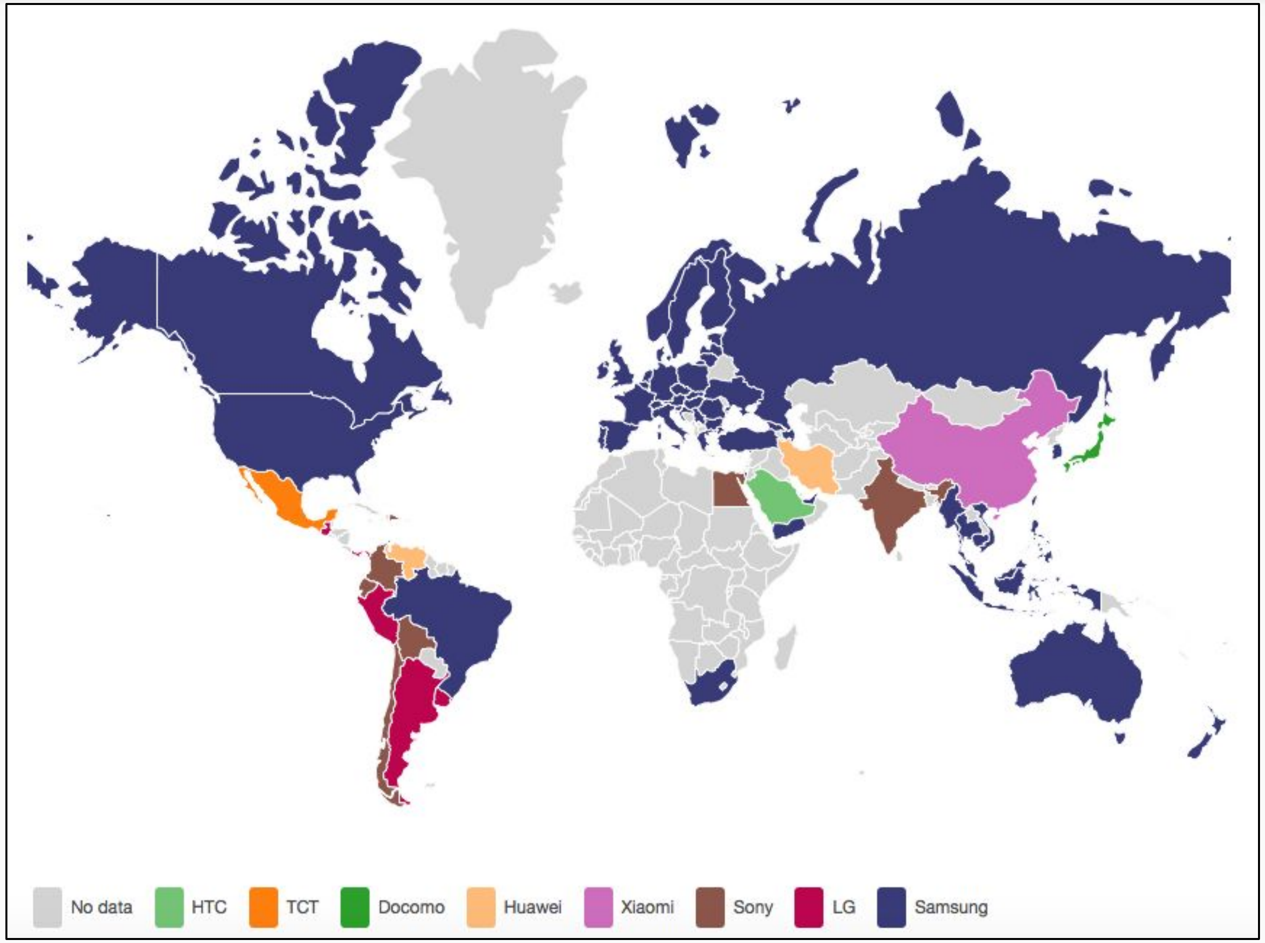


24,093 different devices in 2015!

Takeaway message:

There is a lot of different devices to take into account

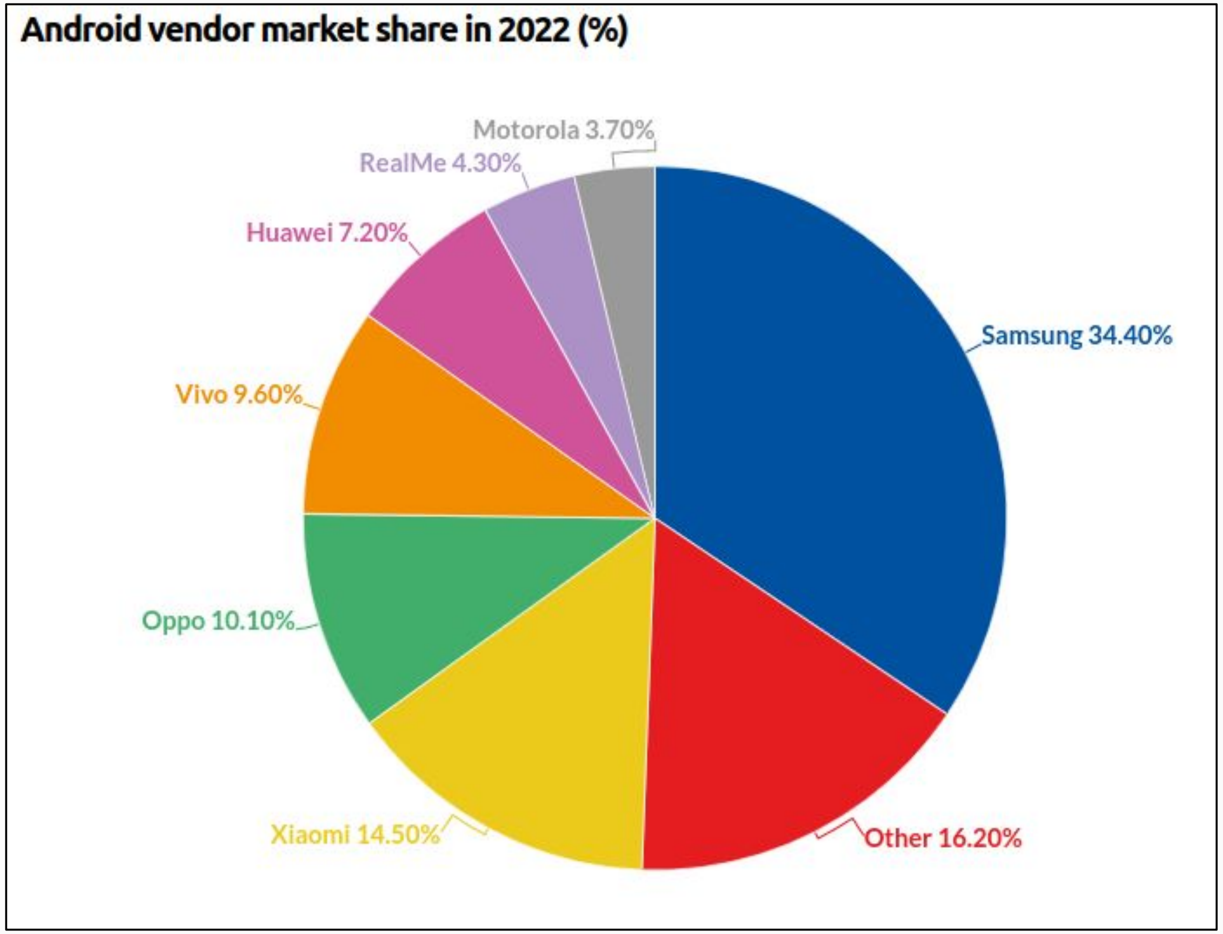
Android Data



Takeaway message:

There is a lot of different devices to take into account

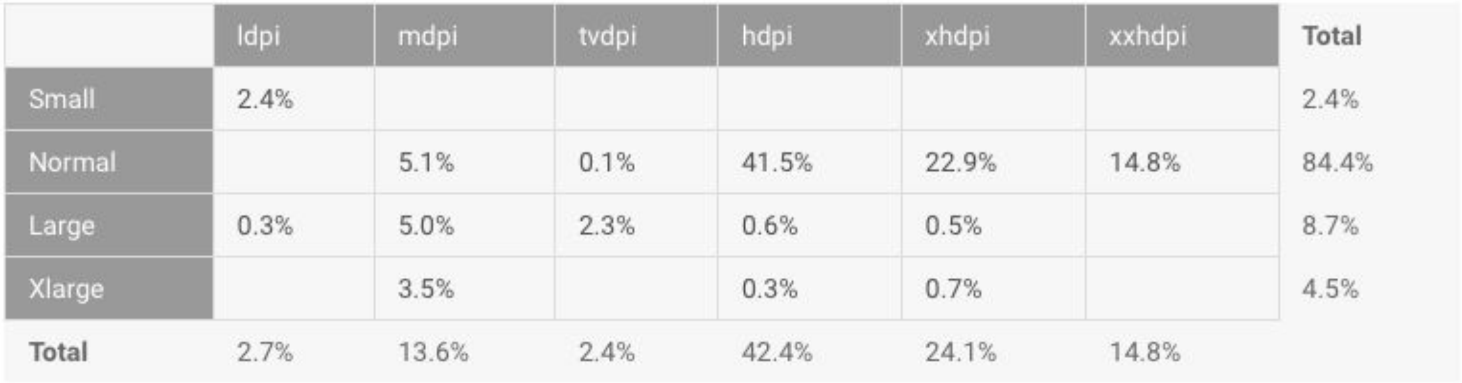
Android Data



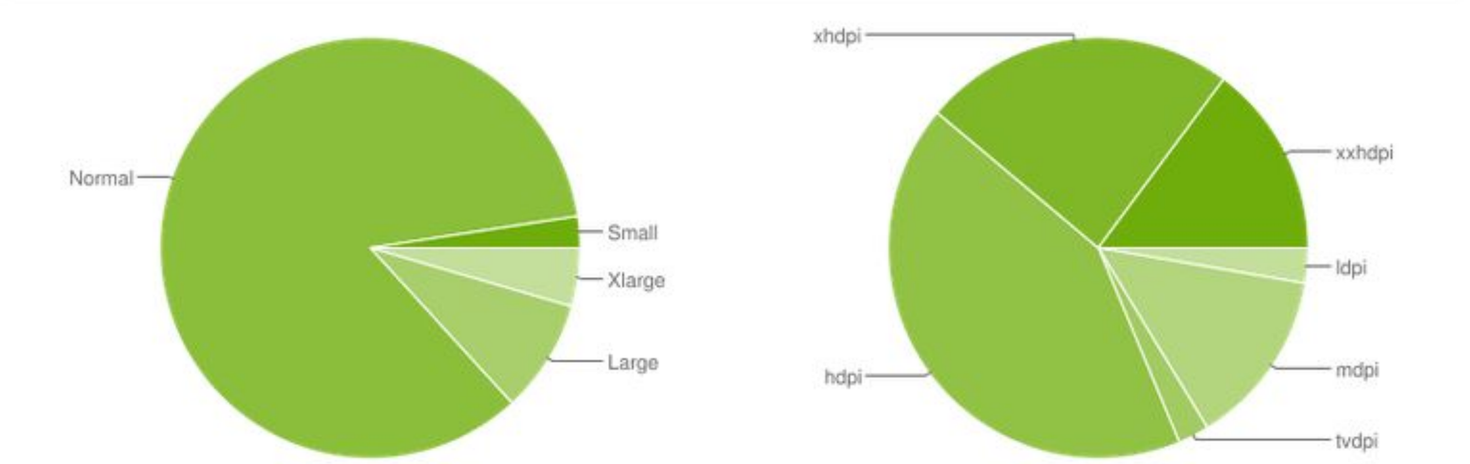
Takeaway message:

There is a lot of different devices to take into account

Android Data

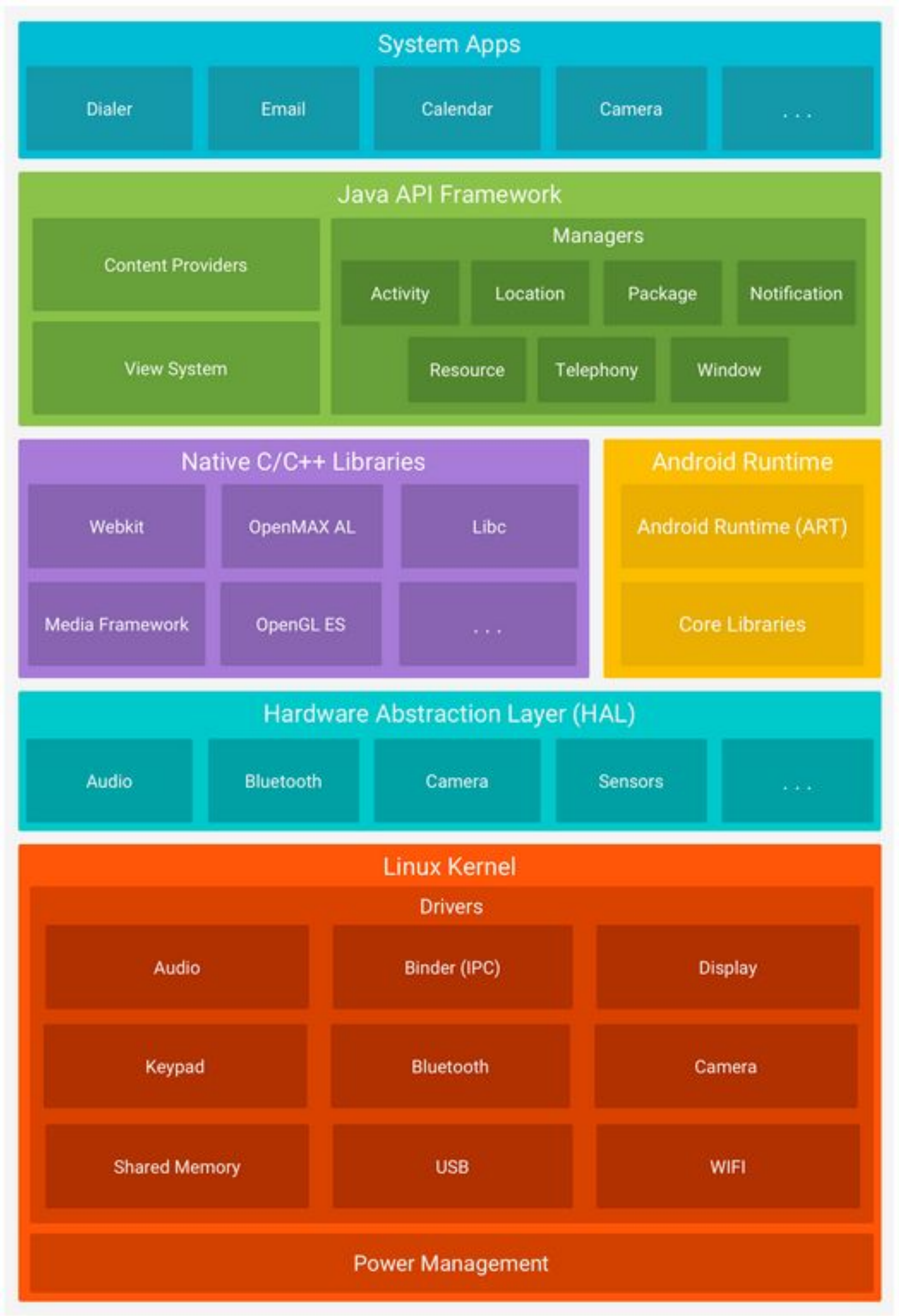


	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	2.4%						2.4%
Normal		5.1%	0.1%	41.5%	22.9%	14.8%	84.4%
Large	0.3%	5.0%	2.3%	0.6%	0.5%		8.7%
Xlarge		3.5%		0.3%	0.7%		4.5%
Total	2.7%	13.6%	2.4%	42.4%	24.1%	14.8%	



Takeaway message:

There is a lot of different conditions that we need to take into account



System Apps

Dialer | Email | Calendar | Camera | . . .

Java API Framework

Content Providers

View System

Managers

Activity | Location | Package | Notification

Resource | Telephony | Window

Native C/C++ Libraries

Webkit | OpenMAX AL | Libc

Media Framework | OpenGL ES | . . .

Android Runtime

Android Runtime (ART)

Core Libraries

Hardware Abstraction Layer (HAL)

Audio | Bluetooth | Camera | Sensors | . . .

Linux Kernel

Drivers

Audio | Binder (IPC) | Display

Keypad | Bluetooth | Camera

Shared Memory | USB | WIFI

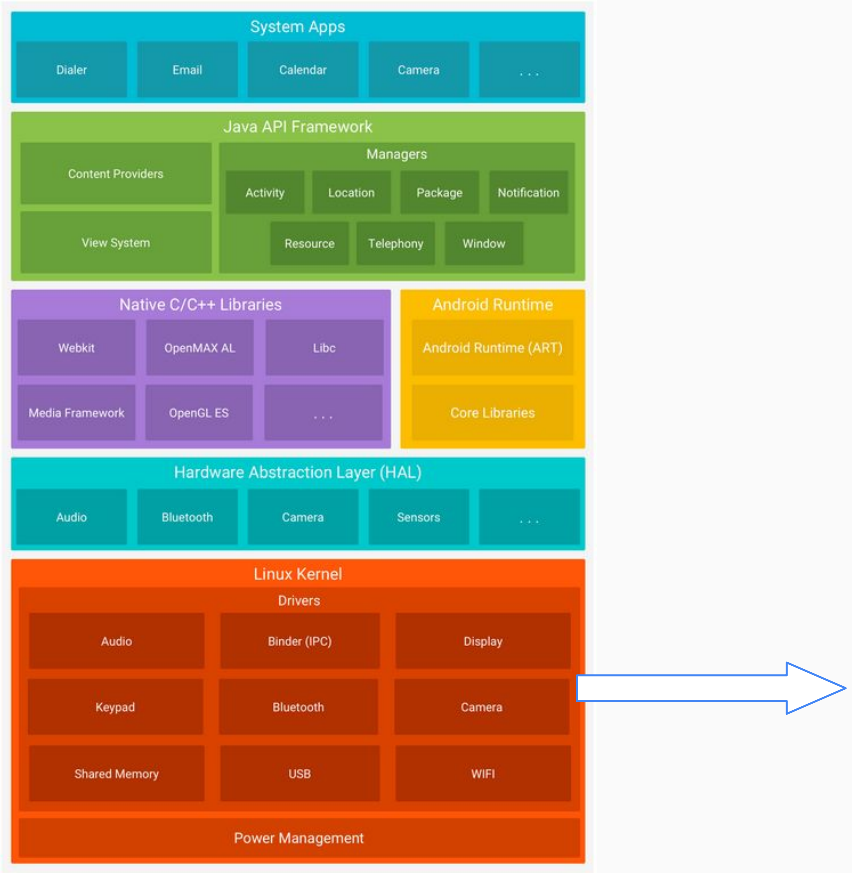
Power Management

Android Architecture

The rest of the course will be dedicated on how to build Apps…

… but what is underneath?

Android Architecture



OS is Built on top of Linux kernel

Advantages:

- Portability (i.e. easy to compile on different hardware architectures)
- Security (e.g. secure multi-process environment)
- Power Management
- Android Runtime (ART) relies on the kernel for threads and memory management
- Manufacturers build drivers on top of a reliable kernel

Android Architecture

- User based permission model
- Processes are isolated
- Inter-process communication (IPC)
- Resources are protected from other processes
- Each application has its own User ID (UID)
  - This means that in Android, each App is a different Linux User
- Application Sandbox (process isolation)
- Verified boot

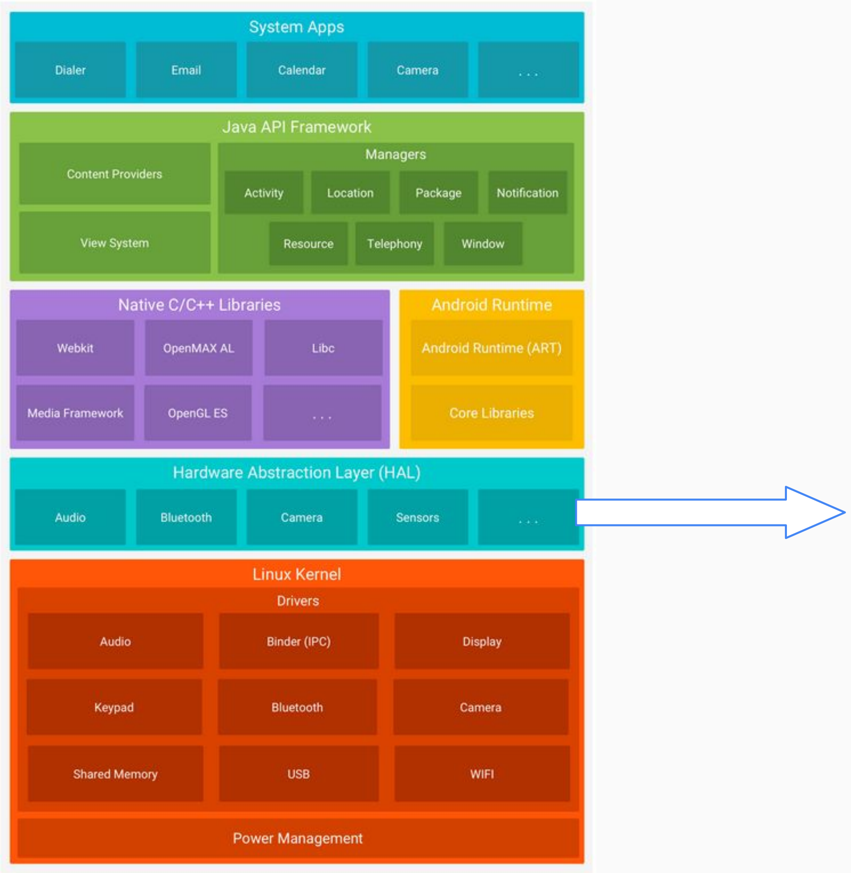
Android Architecture

- Android 5.0:
  - Mandatory Access Control (MAC) between system and apps, all third-party apps ran within the same SELinux context so inter-app isolation was primarily enforced by UID-based sandbox.
- Android 8.0:
  - limited system calls available to user-level apps

Android Architecture

- Android 9.0:
  - all non-privileged apps with SDK version >= 28 must run in individual SELinux sandboxes, providing MAC on a per-app basis
- Android 10:
  - apps have a limited raw view of the filesystem, with no direct access to paths like **/sdcard/DCIM**. However, apps retain full raw access to their package-specific paths

Android Architecture



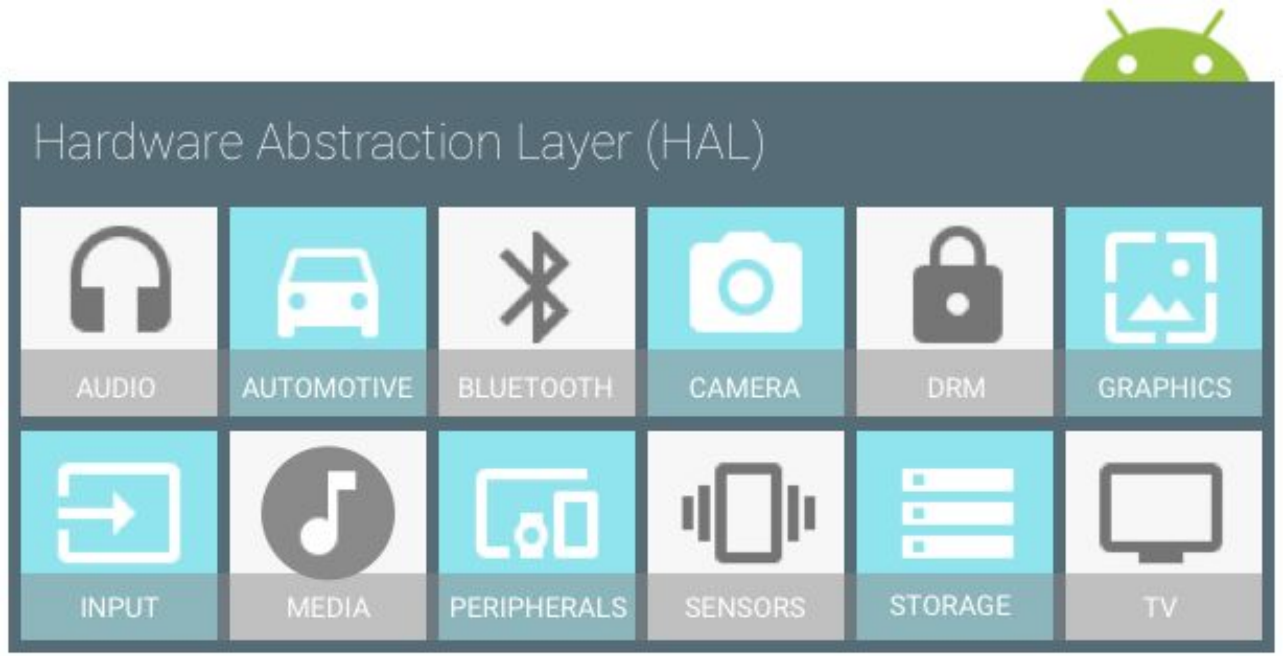
Hardware Abstraction Layer (HAL)

Advantages:

- Shadows the real device
- Manages different devices of the same type
- Standard interface to expose lower level capabilities to higher level APIs

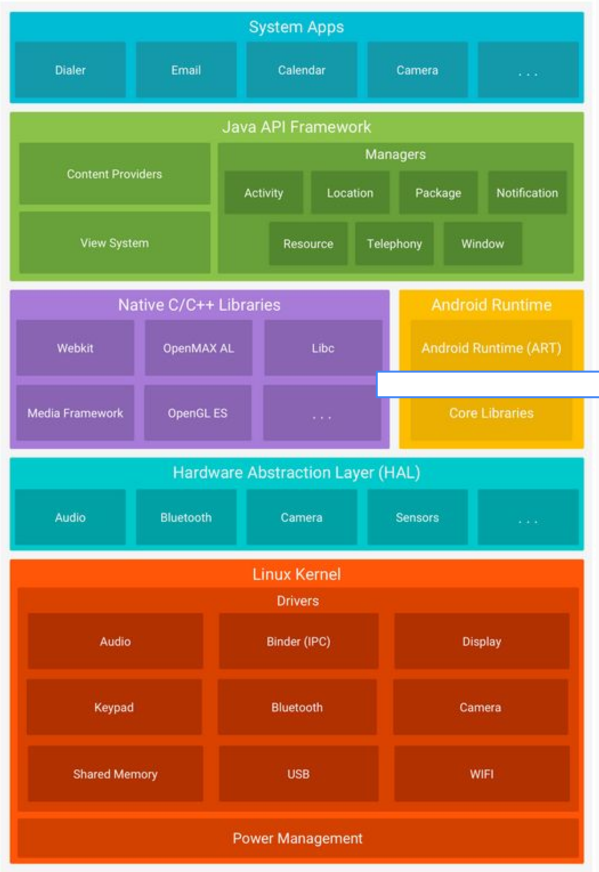
Android Architecture

HAL: Standard interface defined by Android that manufacturers have to implement – Android is agnostic about lower level driver implementations.



- Application developers rely on common APIs
- Depending on the hardware, appropriate libraries are loaded

Android Architecture



Native C/C++ Libraries

- Graphics (Surface Manager)
- Multimedia (Media Framework)
- Database DBMS (SQLite)
- Font Management (FreeType)
- WebKit
- C libraries (Bionic)

Android Architecture

The **NDK** Enables C/C++ coding

- Useful if you want to interact/extend with some native libraries
  - ○ Performance
  - ○ Reuse your C/C++ libraries
- JAVA APIs are provided for most used libraries
- NDK can be installed as an Android Studio plugin

```
public class myNDKActivity extends Activity {
        public native void doNothing() { }
}
```

Android Architecture

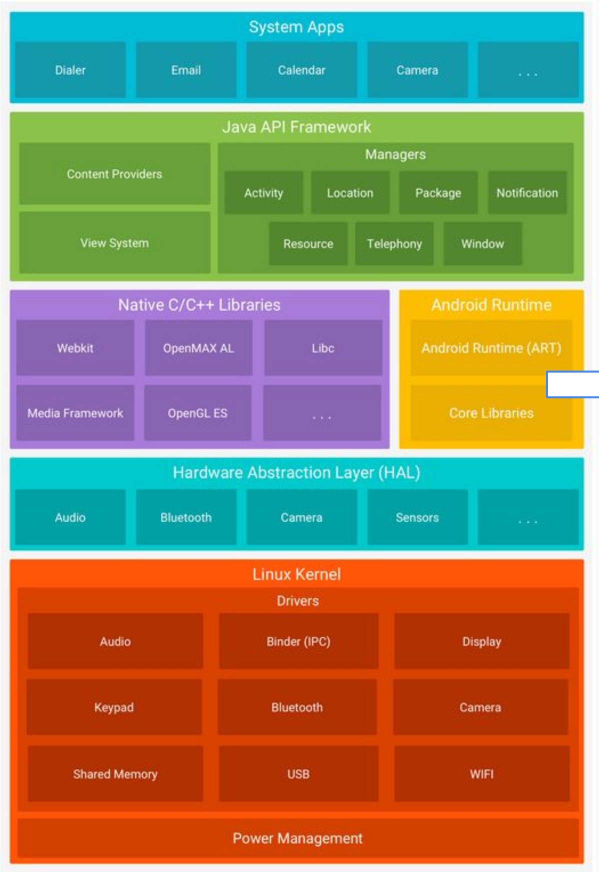
the NDK can be useful for cases in which you need to do one or more of the following:

- Squeeze extra performance out of a device to achieve low latency or run computationally intensive applications, such as games or physics simulations: https://developer.android.com/ndk/guides
- Reuse your own or other developers' C or C++ libraries.

Usage:

- Use the NDK to compile C and C++ code into a native library and package it into your APK using Gradle.
- Your Java code can then call functions in your native library through the Java Native Interface (JNI) framework.

Android Architecture



Android Runtime (ART)

It is a Java Virtual Machine (JVM) implementation (not using the Oracle JVM) optimized for memory-constrained devices

It has everything that an ordinary JVM has:
- **compiles Java code into bytecode**
- **interprets bytecode**

a little differently though…

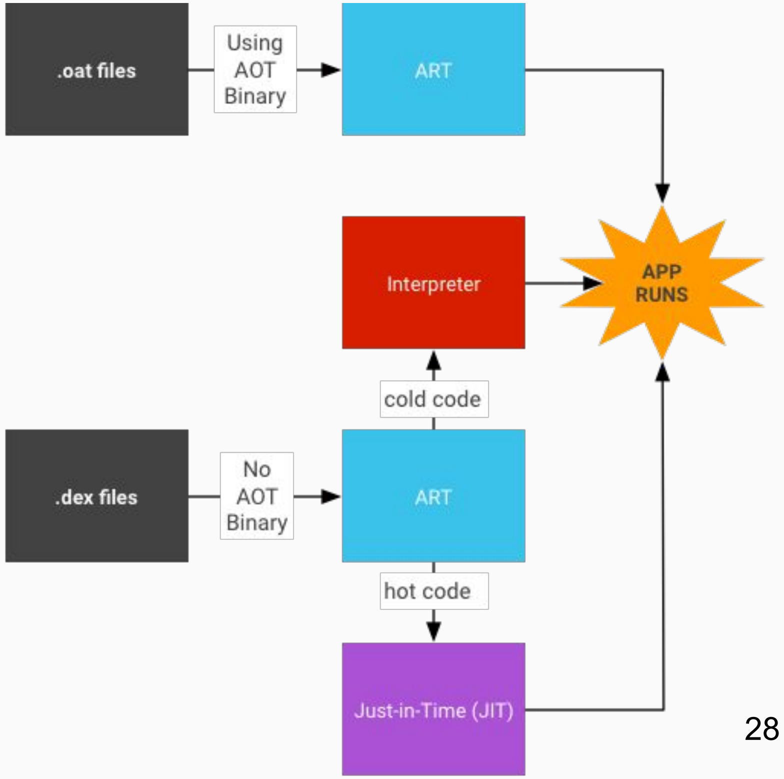
Android Architecture

Starting from Android 5.0, ART is used instead of Dalvik
Several enhancements such as stack size, error handling, Optimized Garbage collection, AOT...

- Designed to run multiple VM on low end devices
- Runs DEX bytecode

Ahead-of-time (AOT) and **Just-in-time** (JIT) compilation

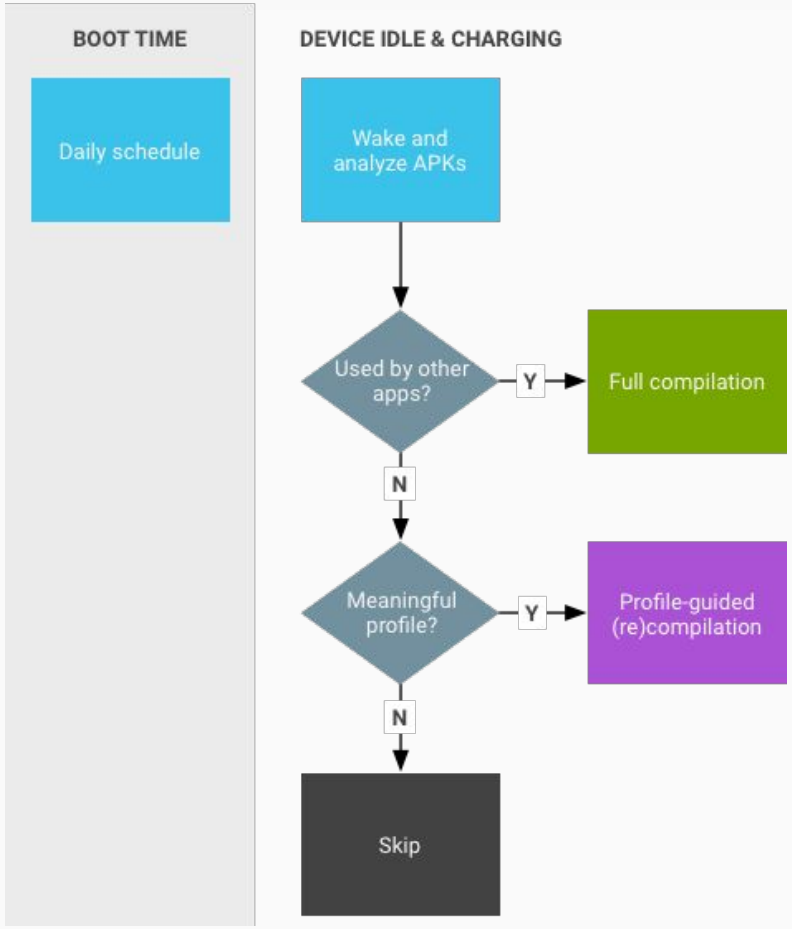
- AOT: At install time, ART compiles APPs using an on-device tool called dex2oat
  - Code compiled at installation
- JIT: code profiling
  - Code partially interpreted when compiled is not available



Android Architecture

There is a lot more to it nowadays.

- DEX files need to be interpreted by the VM (or JIT compiled).
- OAT files are already “machine level” code, so more similar to pure compilation.
- We have a daemon that looks for uncompiled apps when the device is idle and compiles them through.
- Compiled apps may be recompiled sometimes by JIT if the conditions have changed…

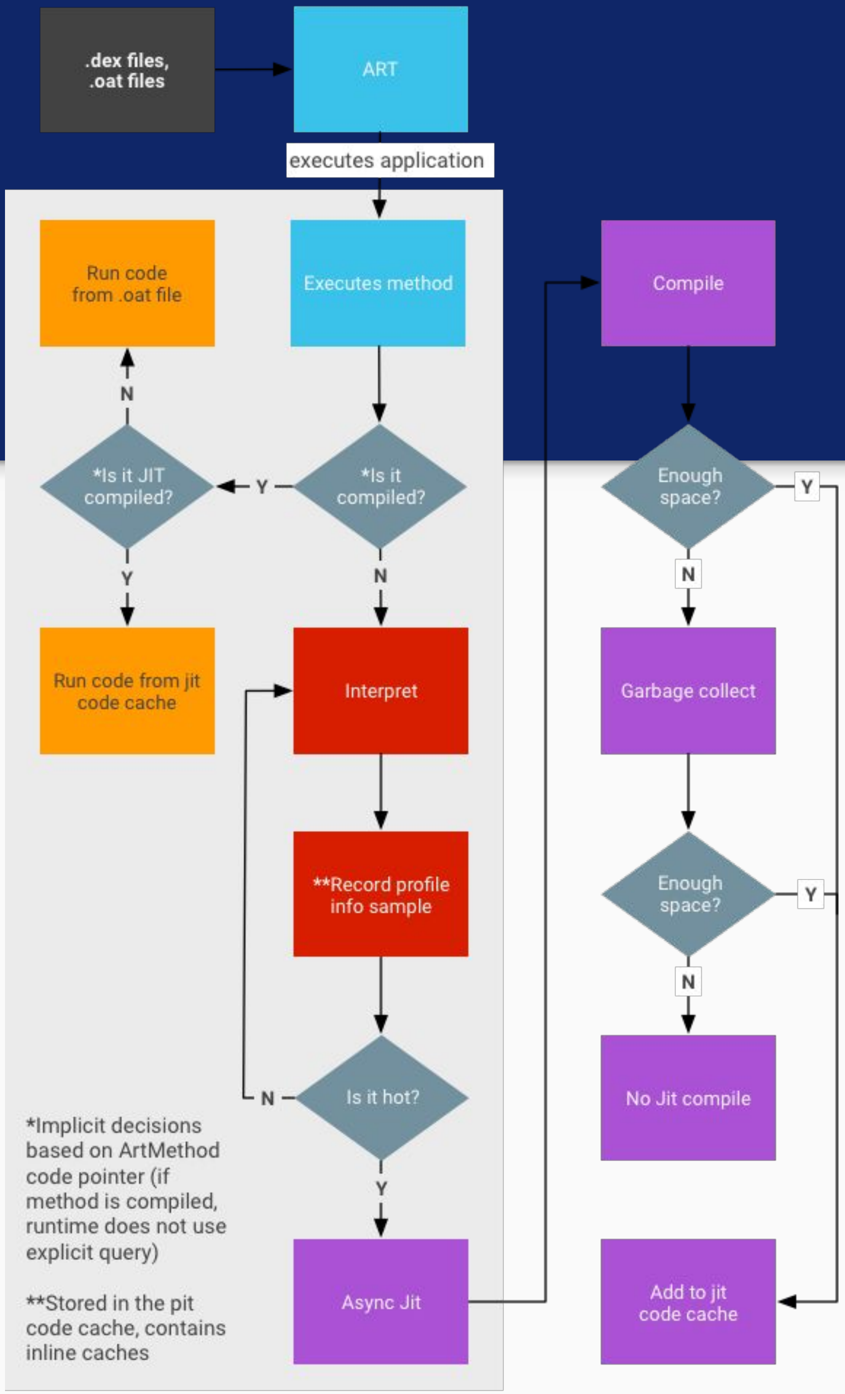


Android Architecture

- AOT and JIT replace the code interpretation that was classic for Java.

- However, their management is complex (see aside).

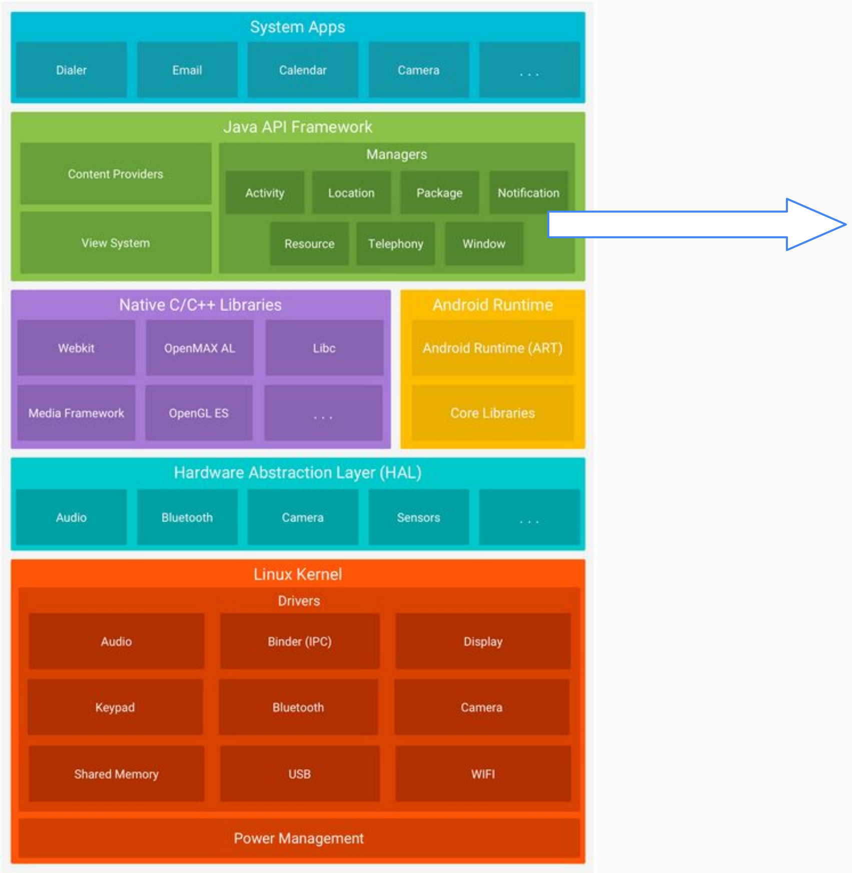
- Do not confuse AOT and JIT with the “compilation” that takes place when developing the app and outputs an APK…
    - The latter outputs bytecode, which still is not machine code.



*Implicit decisions based on ArtMethod code pointer (if method is compiled, runtime does not use explicit query)

**Stored in the pit code cache, contains inline caches

Android Architecture



System Apps

Dialer | Email | Calendar | Camera | . . .

Java API Framework

Content Providers

View System

Managers

Activity | Location | Package | Notification

Resource | Telephony | Window

Native C/C++ Libraries

Webkit | OpenMAX AL | Libc

Media Framework | OpenGL ES | . . .

Android Runtime

Android Runtime (ART)

Core Libraries

Hardware Abstraction Layer (HAL)

Audio | Bluetooth | Camera | Sensors | . . .

Linux Kernel

Drivers

Audio | Binder (IPC) | Display

Keypad | Bluetooth | Camera

Shared Memory | USB | WIFI

Power Management

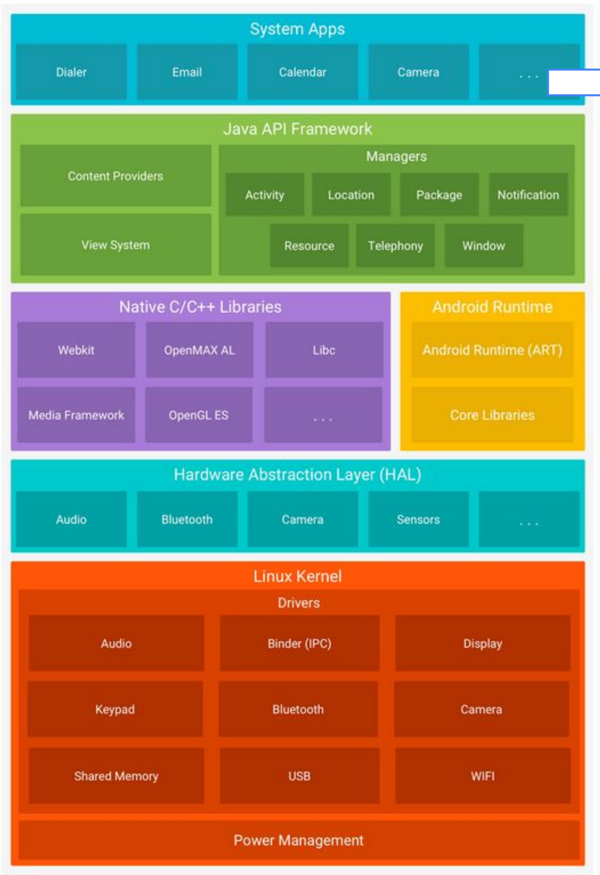
APIs (Core Components of Android)

- Activity Manager
- Packet Manager
- Telephony Manager
- Location Manager
- Contents Provider
- Notification Manager
- …

Android Architecture

- **View System**
  - Through which you build the APP UI
- **Resource Manager**
  - Through which you handle resources
- **Notification Manager**
  - Through which you can access to different kind of notifications
- **Activity Manager**
  - Which handles the Activity lifecycle and provides a back stack
- **Content Providers**
  - To share data among APPs

Android Architecture



System Apps

Dialer | Email | Calendar | Camera | . . .

Java API Framework

Content Providers | View System

Managers

Activity | Location | Package | Notification

Resource | Telephony | Window

Native C/C++ Libraries

Webkit | OpenMAX AL | Libc

Media Framework | OpenGL ES | . . .

Android Runtime

Android Runtime (ART)

Core Libraries

Hardware Abstraction Layer (HAL)

Audio | Bluetooth | Camera | Sensors | . . .

Linux Kernel

Drivers

Audio | Binder (IPC) | Display

Keypad | Bluetooth | Camera

Shared Memory | USB | WIFI

Power Management

System Apps

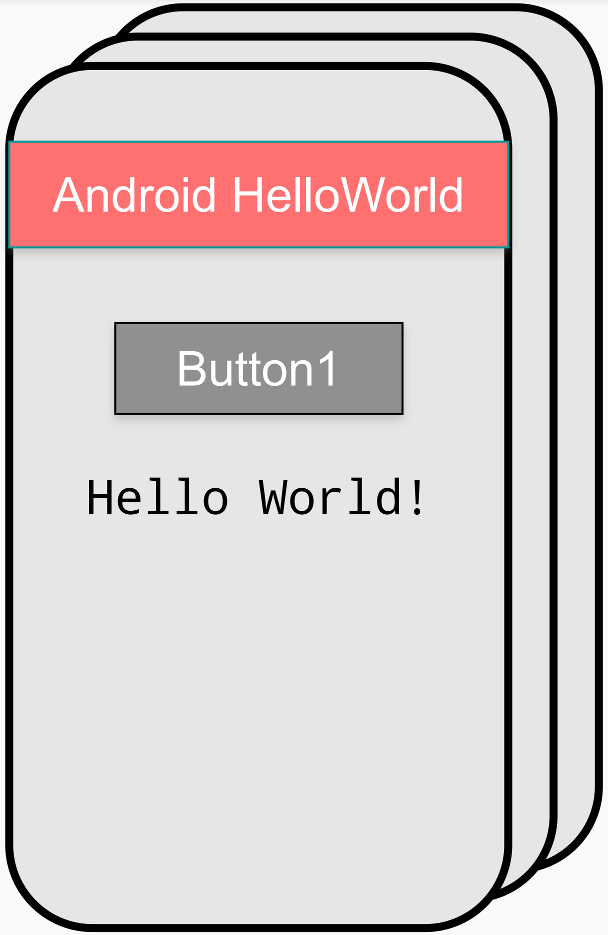
Applications that come with the system by default and have enhanced privileges.

e.g. the alarm clock can wake up the phone even if it is turned off. This is unimplementable by a developer.

App Components

- Activities
- Views
- Resources
- Intents
- Services
- Persistence

App Components



- An **Activity** corresponds to a single screen of the Application.
- An **Application** can be composed of multiples screens (Activities).
- The Home Activity is shown when the user launches an application.
- Different activities can exchange information one with each other.

App Components

- Each activity is composed by a list of graphics components.
- Some of these components (called **Views**) can interact with the user by handling events (e.g. Buttons).
- Two ways to build the graphic interface:

Programmatic Approach:

```
/* Java Code */
Button button = new Button (this);
TextView text = new TextView();
text.setText(“Hello world”);
```

App Components

- Each activity is composed by a list of graphics components.
- Some of these components (called **Views**) can interact with the user by handling events (e.g. Buttons).
- Two ways to build the graphic interface:

Declarative Approach:

```
/* XML Code */
<TextView android.text=@string/hello" android:textcolor=@color/blue
android:layout_width="fill_parent" android:layout_height="wrap_content" />
<Button android.id="@+id/Button01" android:textcolor="@color/blue"
android:layout_width="fill_parent" android:layout_height="wrap_content" />
```

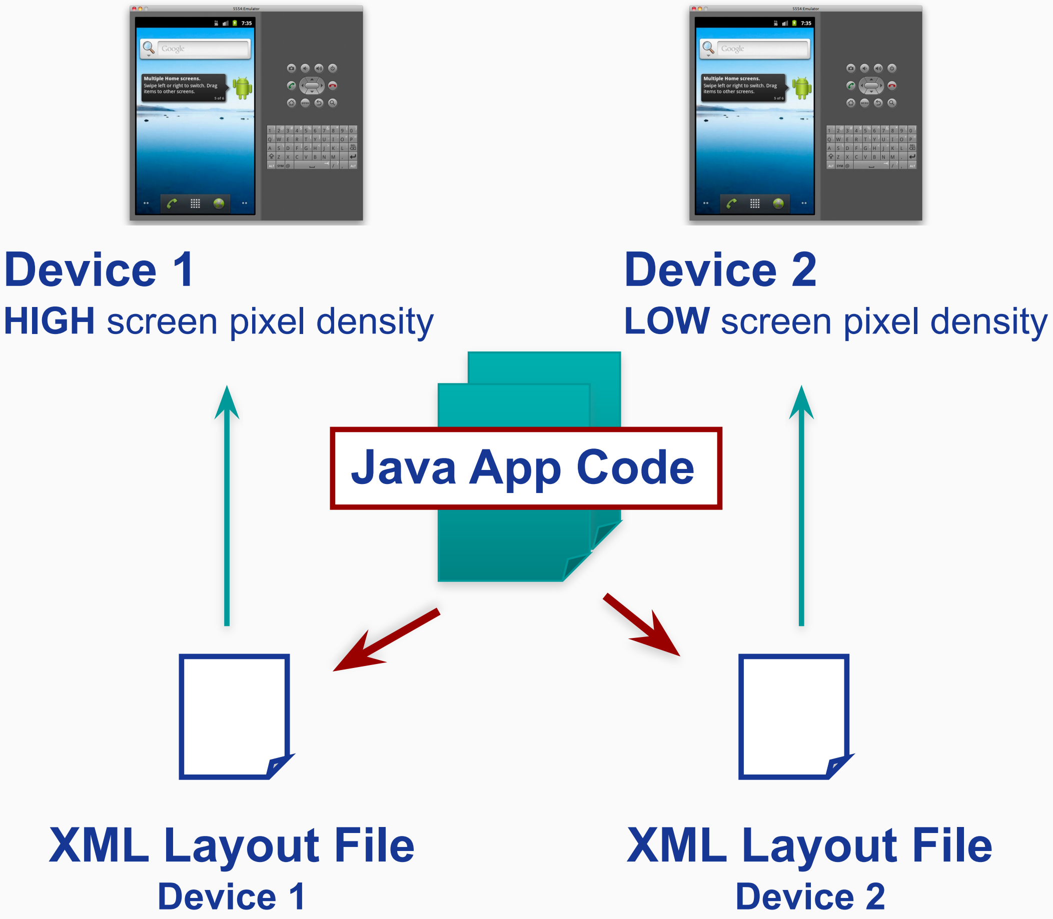
XML

Extensible Markup Language (XML) lets you define and store data in a shareable manner.

Data is organized in a tree and each element contains text and/or children and it is wrapped between a start and an end tag.

```
<root>                    <!-- This is an XML comment -->
    <child1 name="john">
        <exam id="LAM"> 30 </ leaf>
    </ child1>
    <child2 name="jack" />
</ root>
```

App Components



Device 1
HIGH screen pixel density

Device 2
LOW screen pixel density

Java App Code

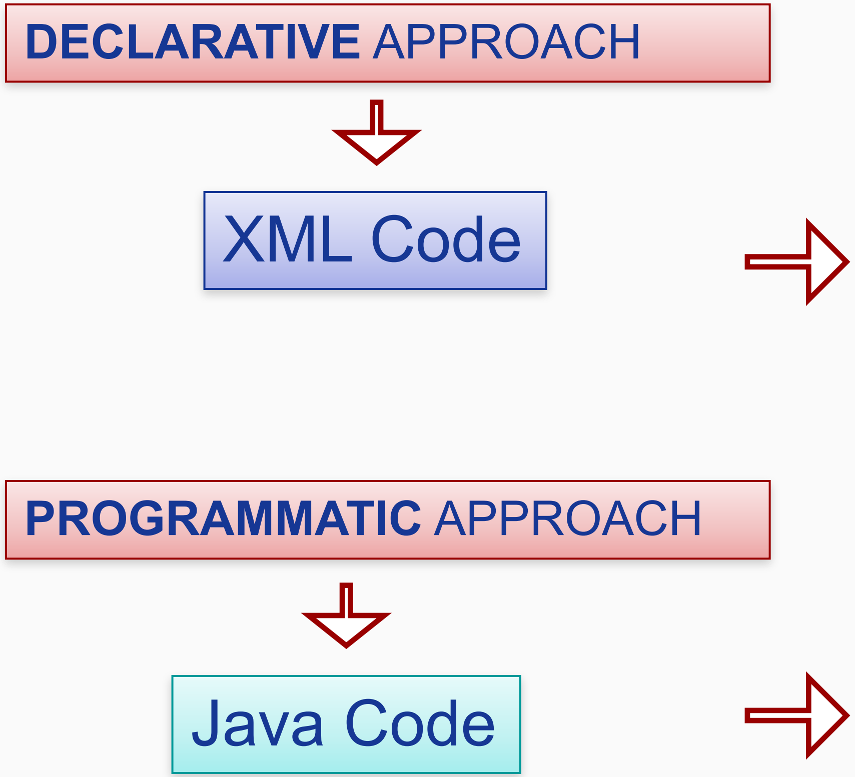
XML Layout File
Device 1

XML Layout File
Device 2

- Build the application layout through XML files.
- Define different XML layouts for different devices
- At runtime, Android detects the current device configuration and loads the appropriate **resources**
- No need to recompile!
- This stands for all other resources

App Components

Android applications typically use both the approaches!



DECLARATIVE APPROACH

XML Code

Define the Application layouts and resources used by the Application (e.g. labels).

PROGRAMMATIC APPROACH

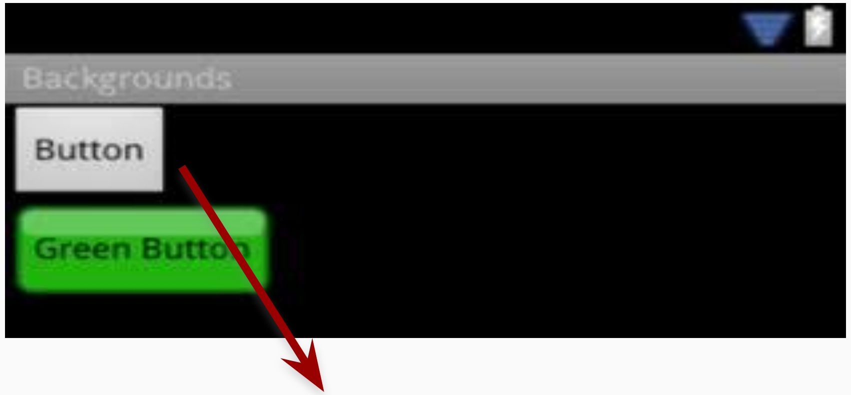
Java Code

Manages the events, and handles the interaction with the user.

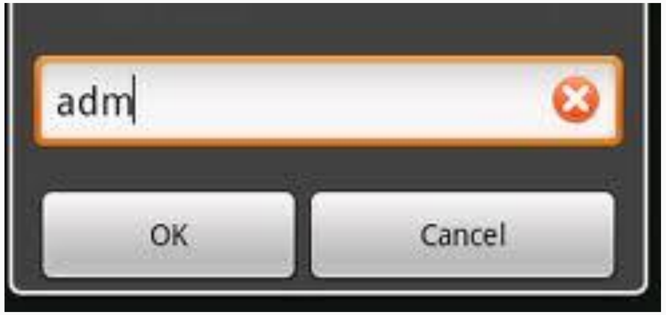
App Components

Views can generate **events** (caused by human interactions) that must be managed by the Android-developer through **CALLBACKS** (from now on you need to know what these are)

Button



TextEdit



```
public void onClick(View arg0) {
    if (arg0 == Button) { /* Manage Button events */ } }
```

App Components

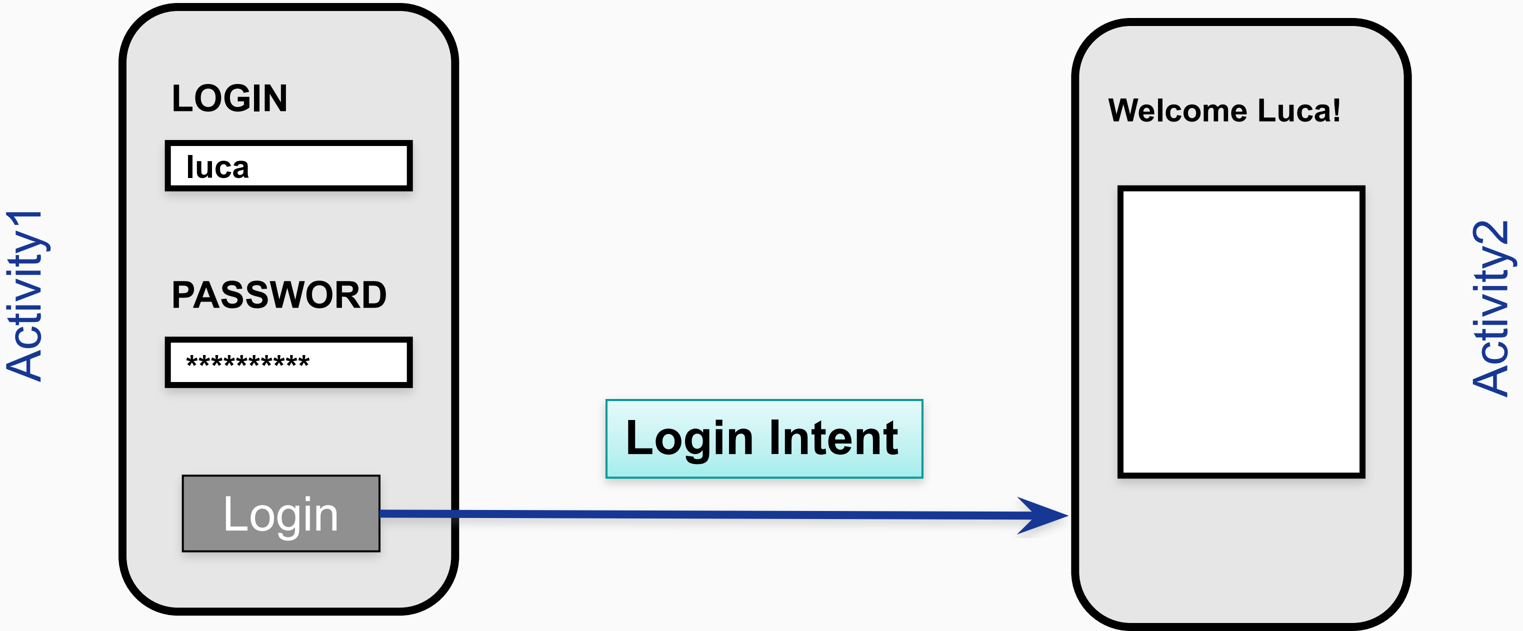
Main difference between Android programming and Java (Oracle) programming:

- Mobile devices have constrained resource capabilities
- Activity lifecycle depends on users’ choice (i.e. change of visibility) as well as on system constraints (i.e. memory shortage).
- Developer must implement lifecycle methods to account for state changes of each Activity … there is no **main** function.

This is a reactive programming style!

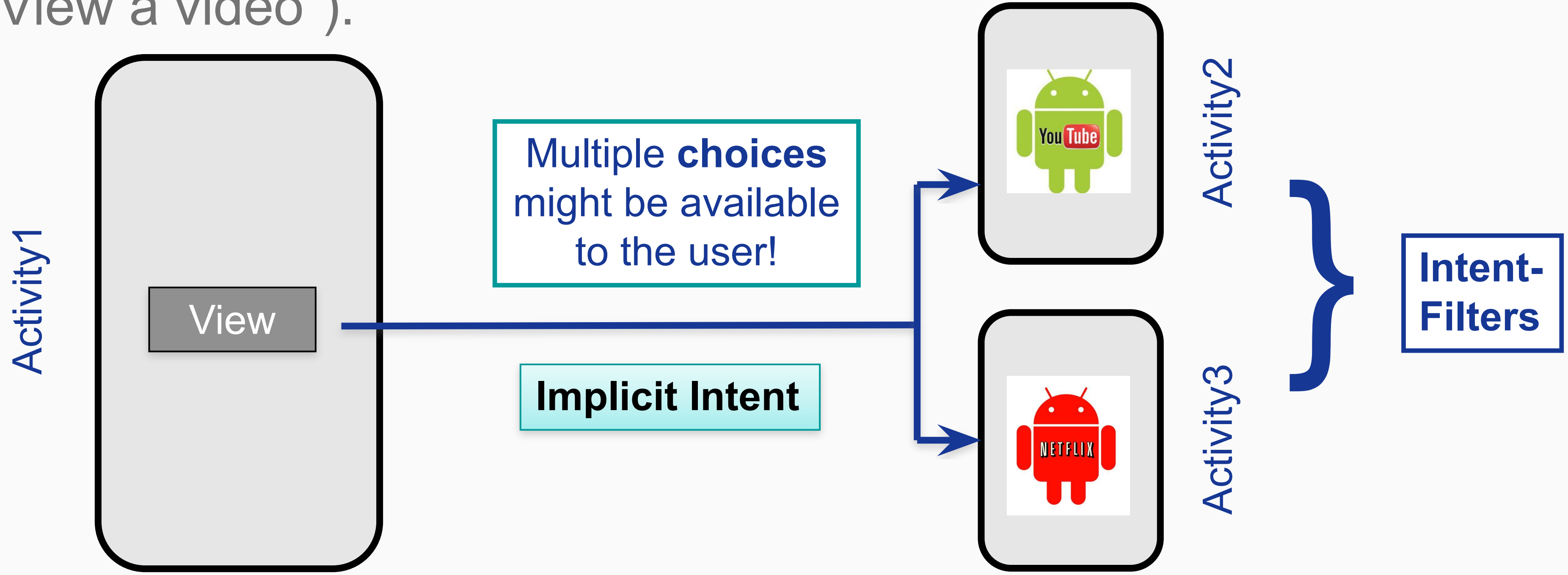
App Components

- **Intents**: asynchronous **messages** to activate core Android components (e.g. Activities).
- **Explicit** Intent □ The component *(e.g. Activity1)* specifies the destination of the intent *(e.g. Activity2)*.



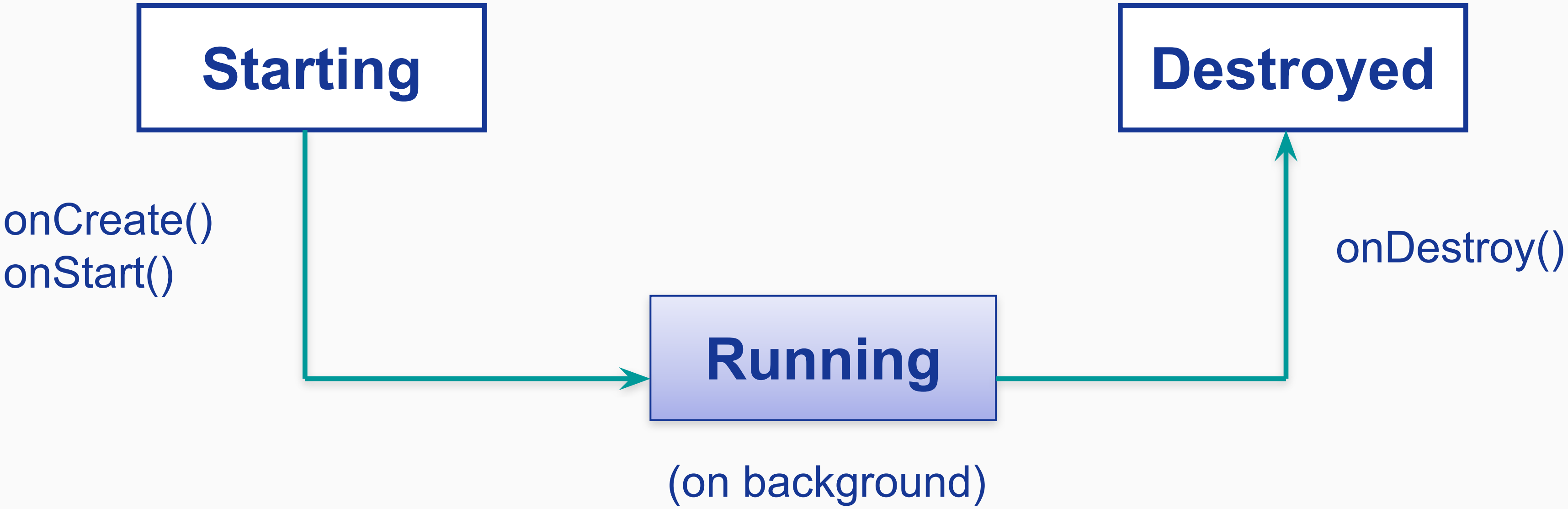
App Components

- **Intents**: asynchronous **messages** to activate core Android components (e.g. Activities).
- **Implicit** Intent →The component (e.g. Activity1) specifies the type of the intent (e.g. "View a video").



App Components

- Services: like Activities, but run in background and do not provide an user interface.
- Used for non-interactive tasks (e.g. networking).



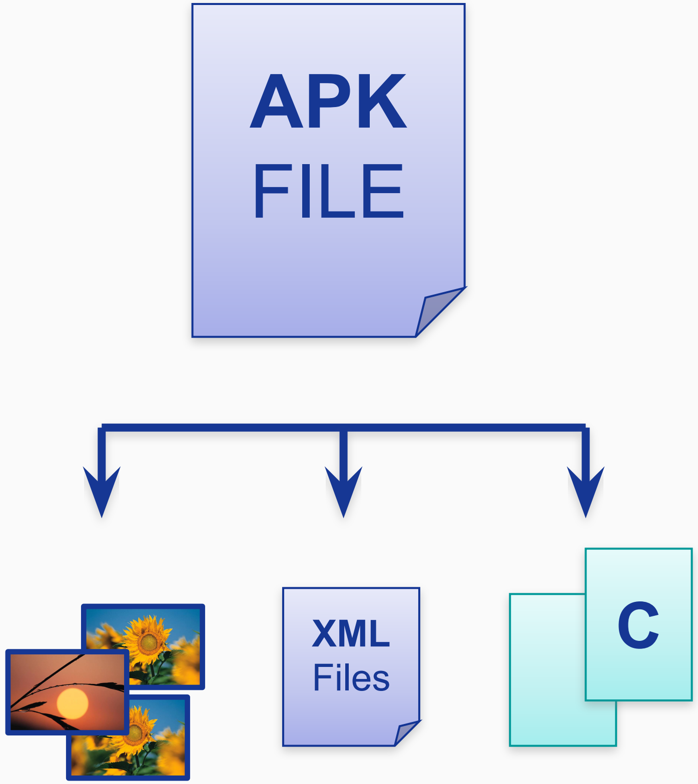
App Components

- Android applications run with a distinct system identity (Linux user ID and group ID), in an isolated way.
- Applications must explicitly share resources and data. They do this by declaring the permissions they need for additional capabilities.
- Applications statically declare the permissions they require.
- User must give his/her consensus upon using the feature.
- Permission must be asked at runtime too.

ANDROIDMANIFEST.XML

```
<uses-permission
android:name=“android.permission.ACCESS_FINE_LOCATION" />
```

App Distribution



- Each Android application is contained in a single APK file.
- Java Bytecode
- Resources (e.g. images. videos, XML layout files)
- Libraries (optimal native C/C++ code)

Questions?

federico.montori2@unibo.it