Programming with Android: System Architecture

Luca Bedogni Marco Di Felice
Dipartimento di Scienze dell'Informazione
Università di Bologna

Outline

Android … What?

- **Android** is a *Linux-based* platform for *mobile devices* …
  - *Operating System*
  - *Applications*
  - *Software Development Kit* (**SDK**)
- Which kind of **mobile devices** … (examples)

SMARTPHONES

TABLETS

EREADERS

ANDROID TV

GOOGLE GLASSES

?

Android … What?

SMART FRIDGE

ANDROID MICROWAVE

SMARTPHONES

TABLETS

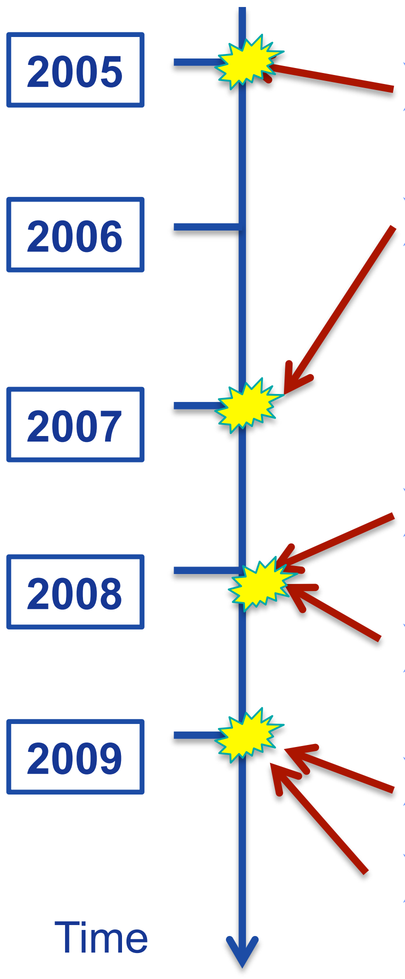
EREADERS

ANDROID TV

GOOGLE GLASSES

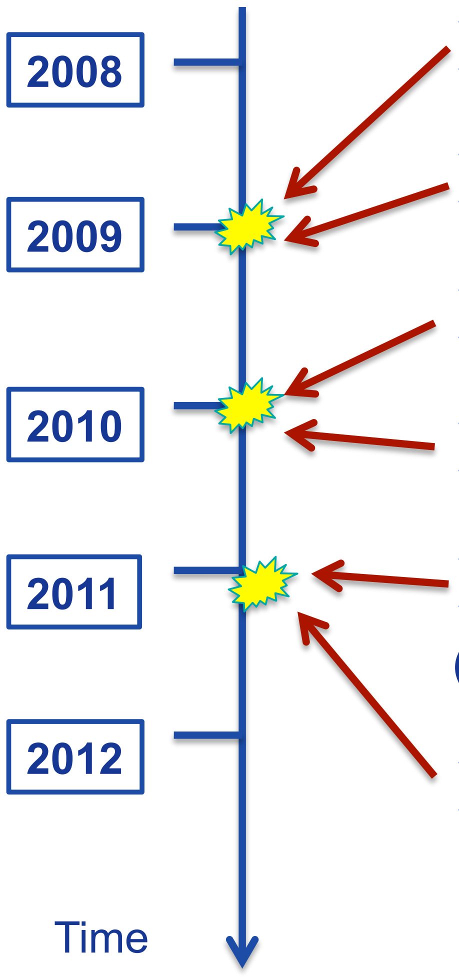
?

Android ... When?



- **Google** buys Android from the **Android Inch**
- 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
- Android **1.1** Released
- Android **1.5 (CupCake)** Released

Android … When?

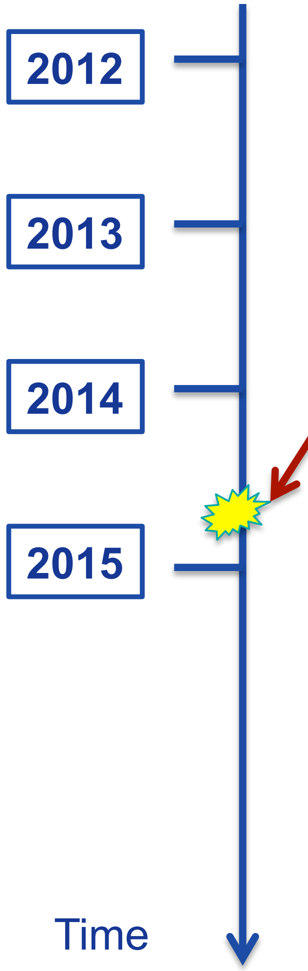


- Android **1.6** (**Donut**) Released
- Android **2.0** (**Eclair**) Released
- Android **2.2** (**Froyo**) Released
- Android **2.3** (**Gingerbread**) Released
- Android **3.0** (**Honeycomb**) Released

(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.)

Android … When?



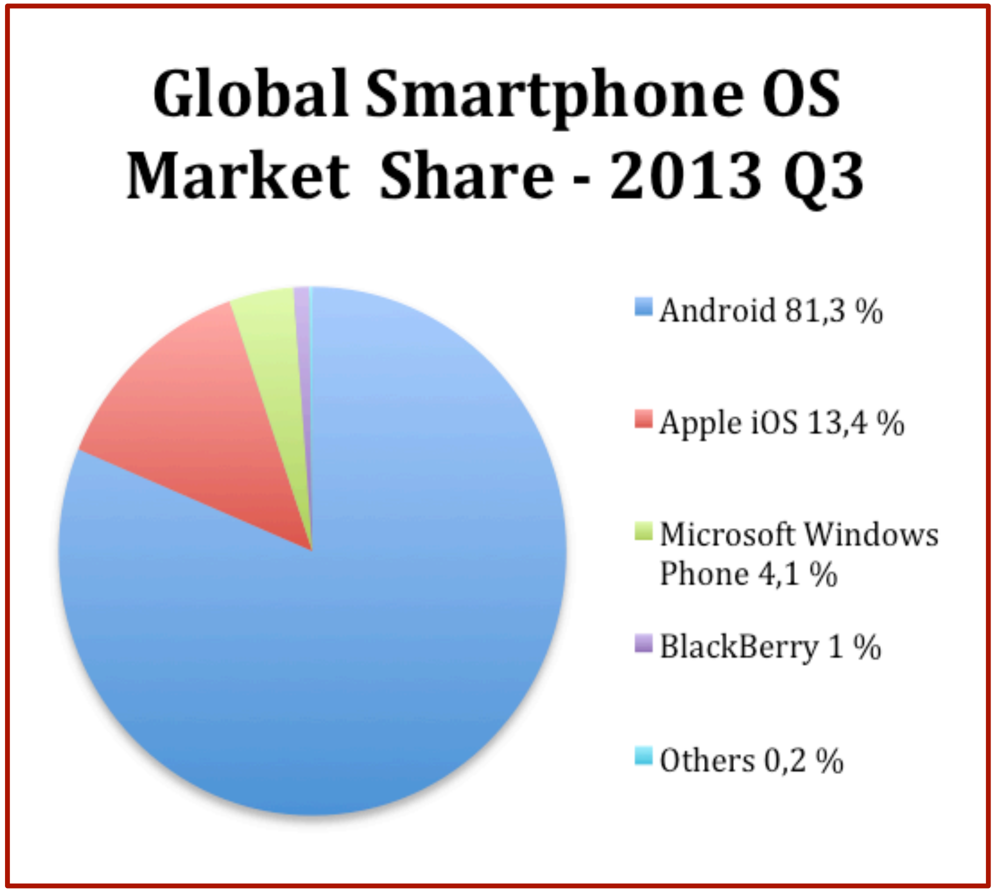
- Android **5.0** (**Lollipop**) Released
  - UI Restyling (Material Design)
  - Personal unlock mechanisms
  - (Battery) Performance optimization
  - New experimental runtime virtual machine, ART…

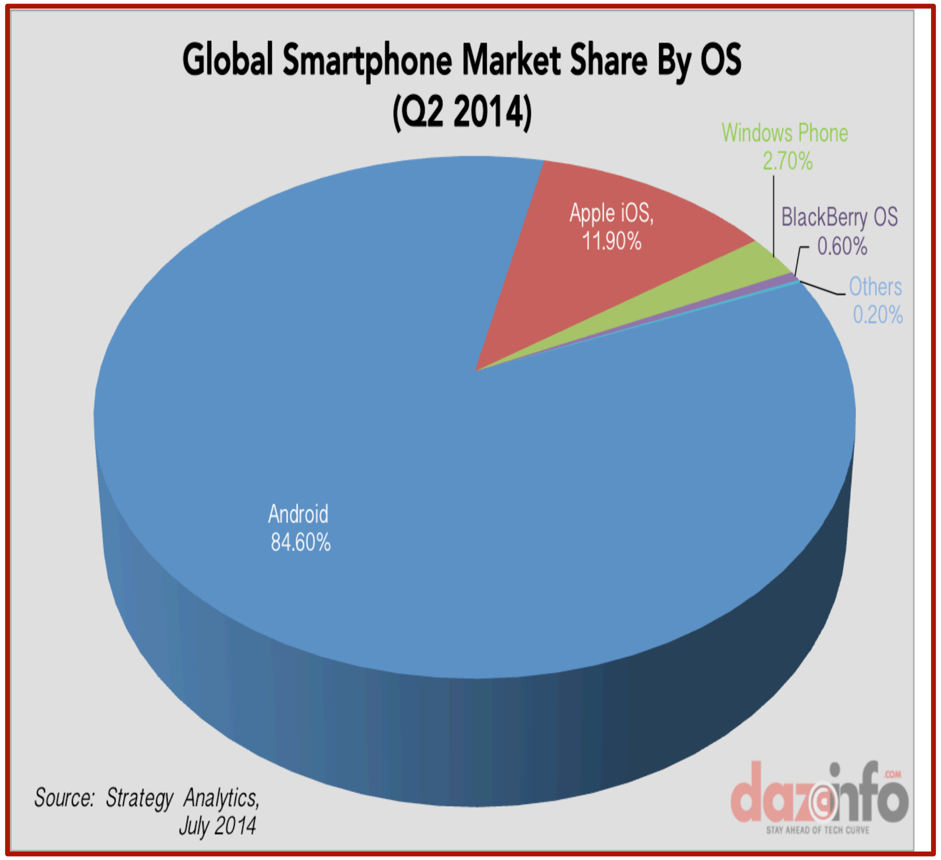
API Level 21 (Android 5.0):

- Material design style
- Lock screen notification
- Support for multiple network connections

  (Scan for network connections with specific capabilities)

Android … Why?



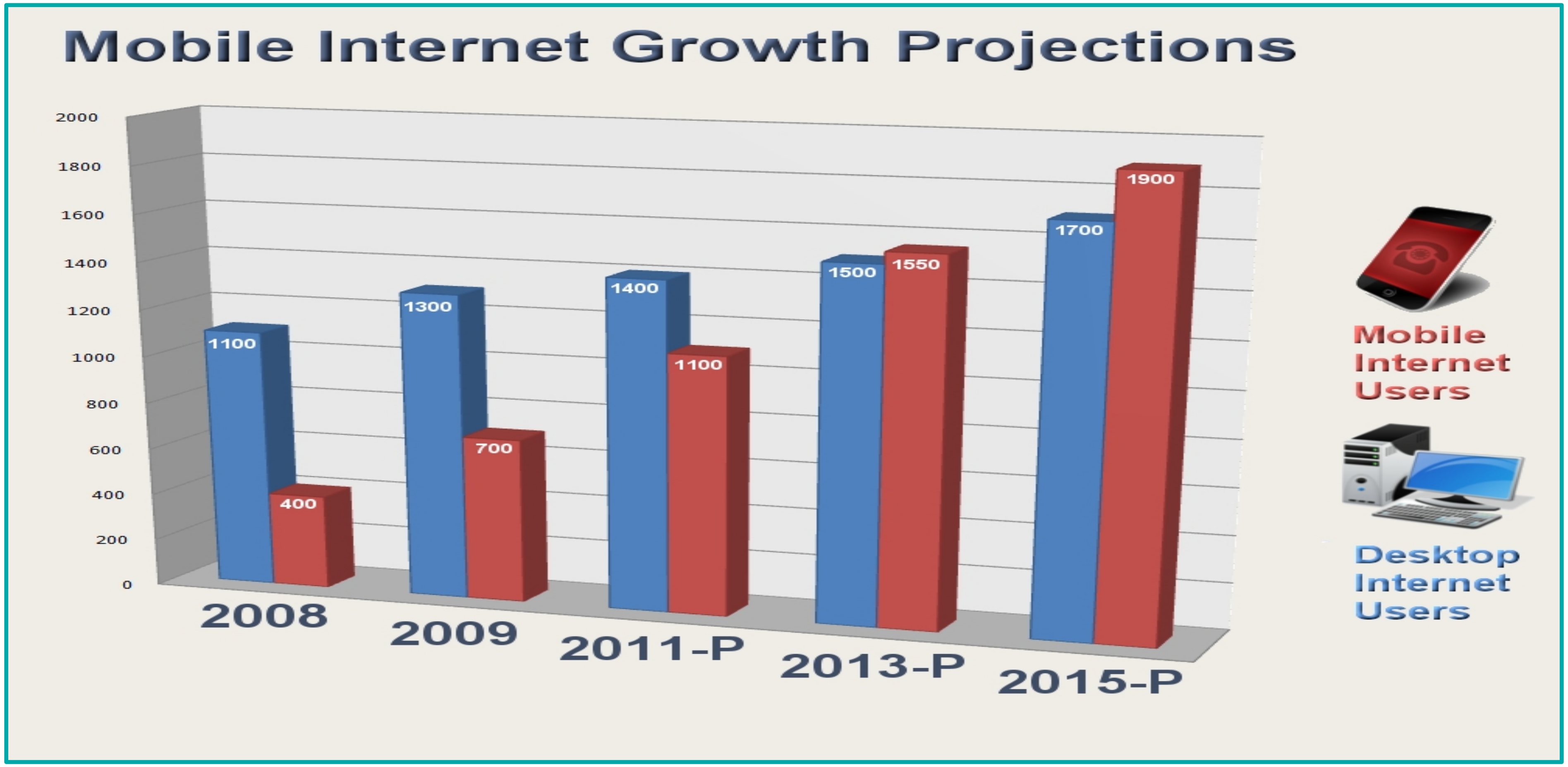


2013 Market Share

www.gartner.com

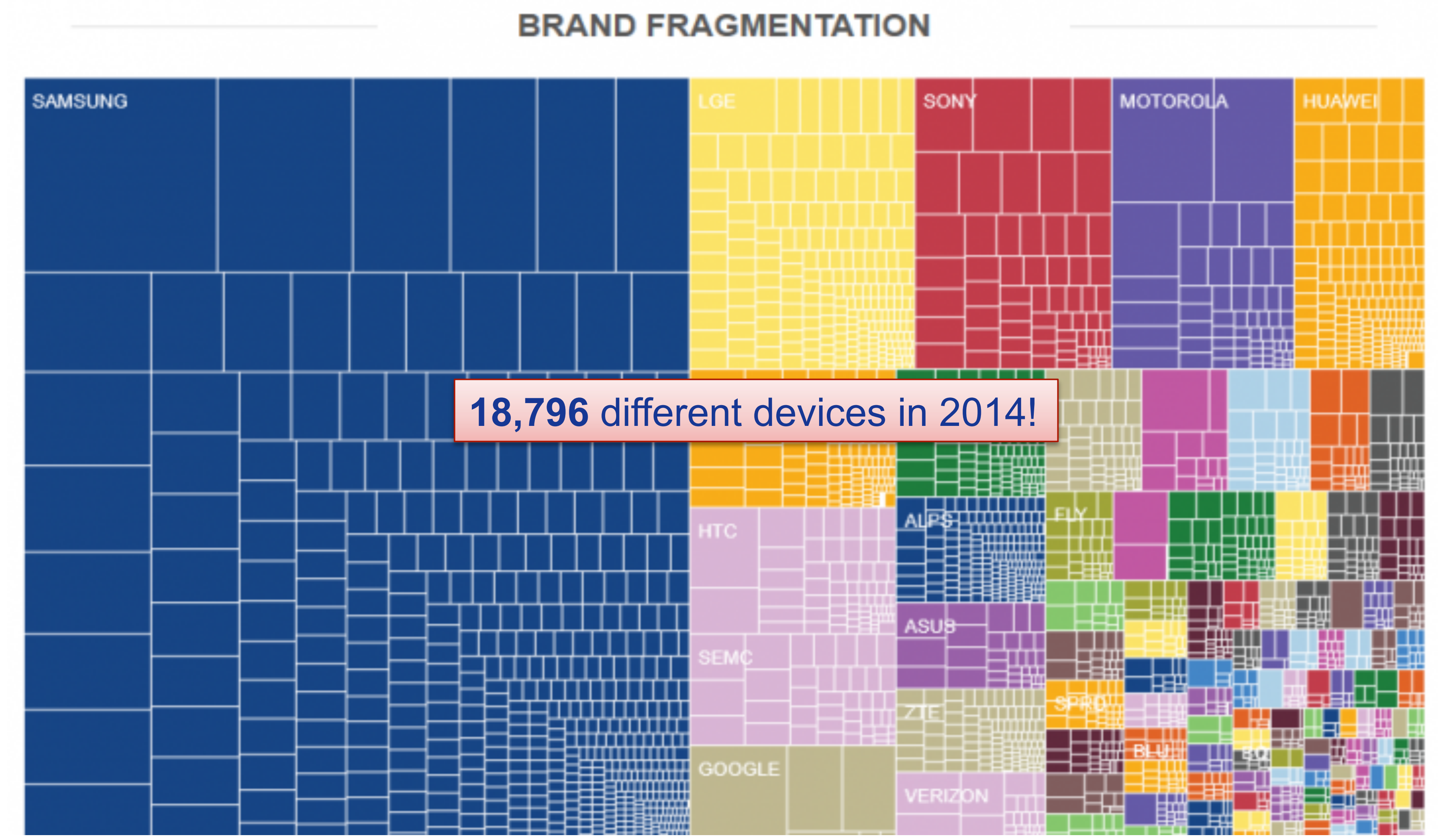
2014 Market Share

Android … Why?



Sorgente: http://jeffalangray.com

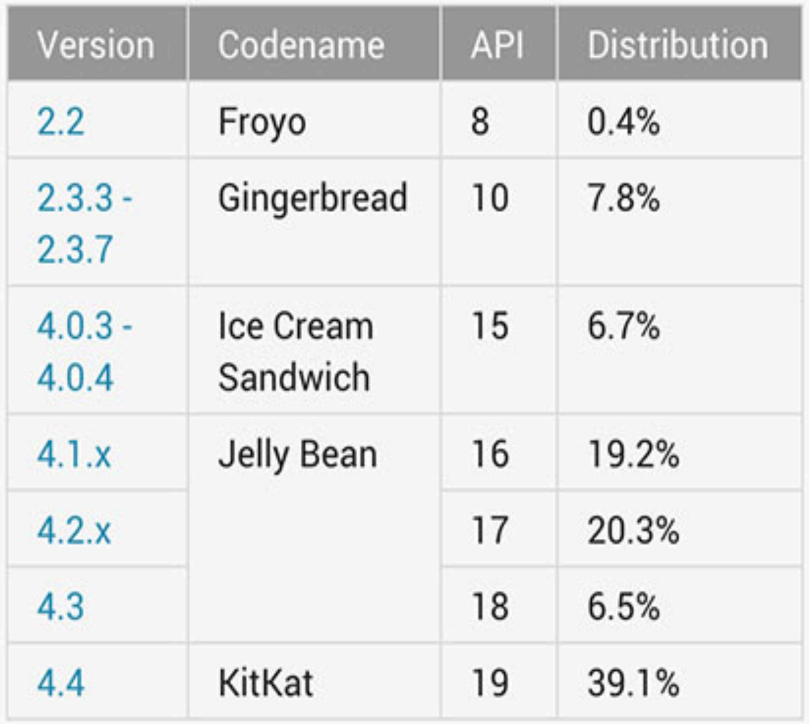
Android … Why?



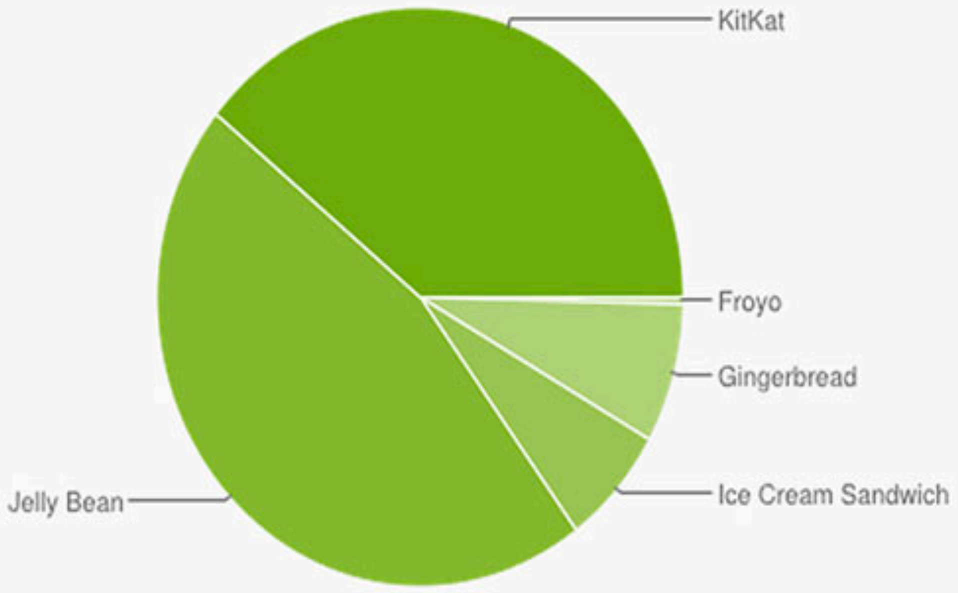
18,796 different devices in 2014!

http://thenextweb.com/mobile/2014/08/21/18796-different-android-devices-according-opensignals-latest-fragmentation-report/

Android … How?



Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.7%
4.1.x	Jelly Bean	16	19.2%
4.2.x		17	20.3%
4.3		18	6.5%
4.4	KitKat	19	39.1%

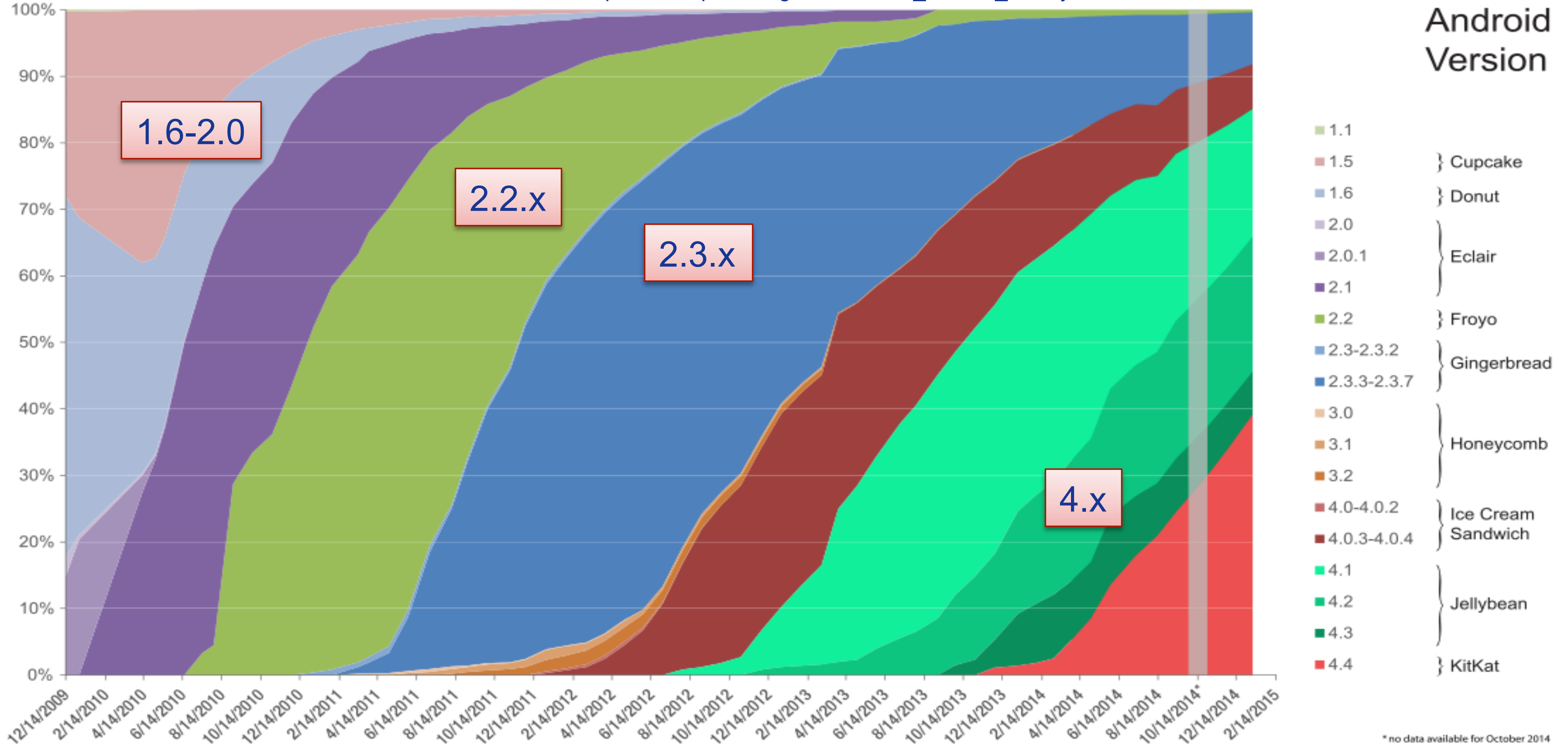


January 2015

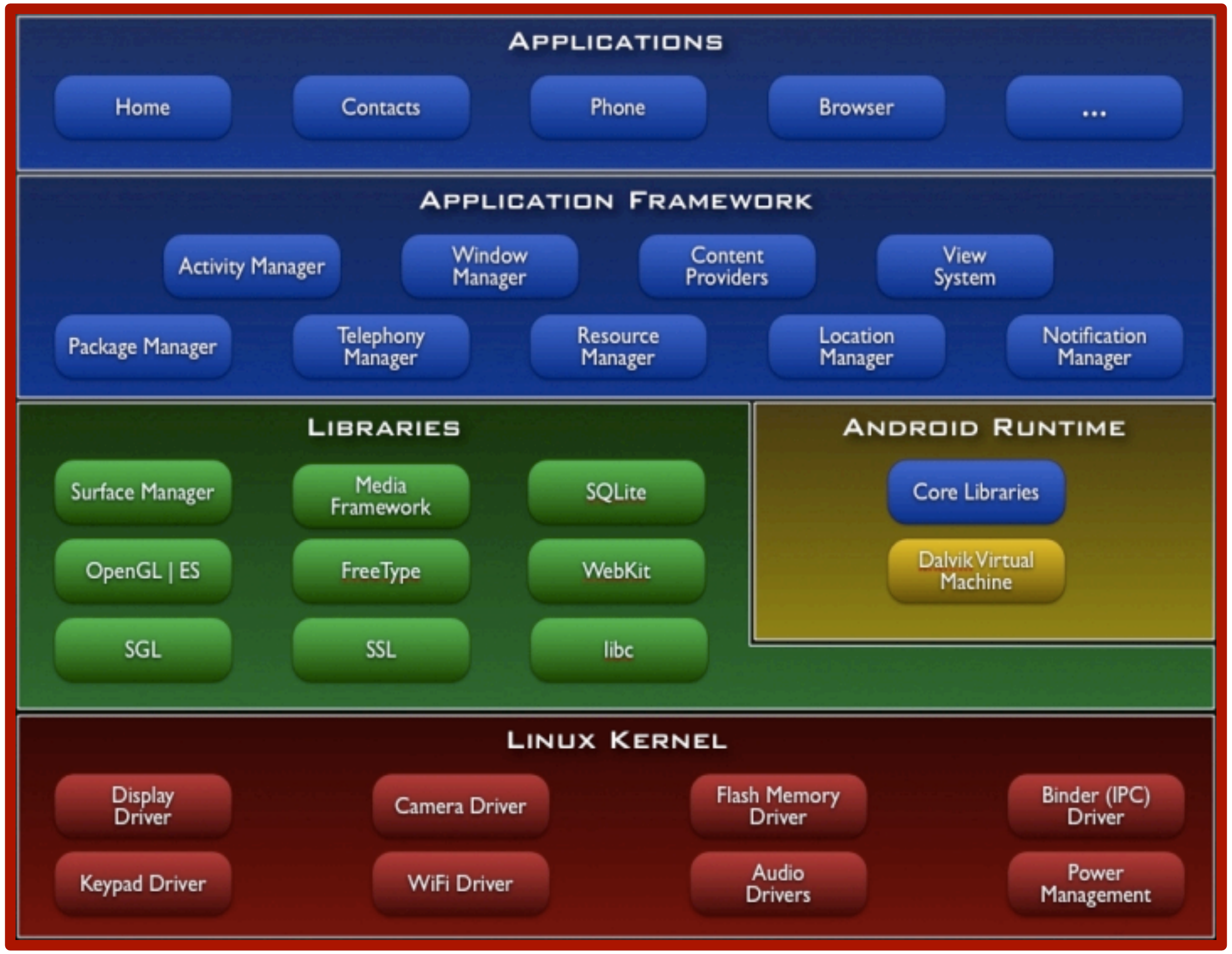
http://www.droid-life.com/

Android … How?

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



The Android Architecture



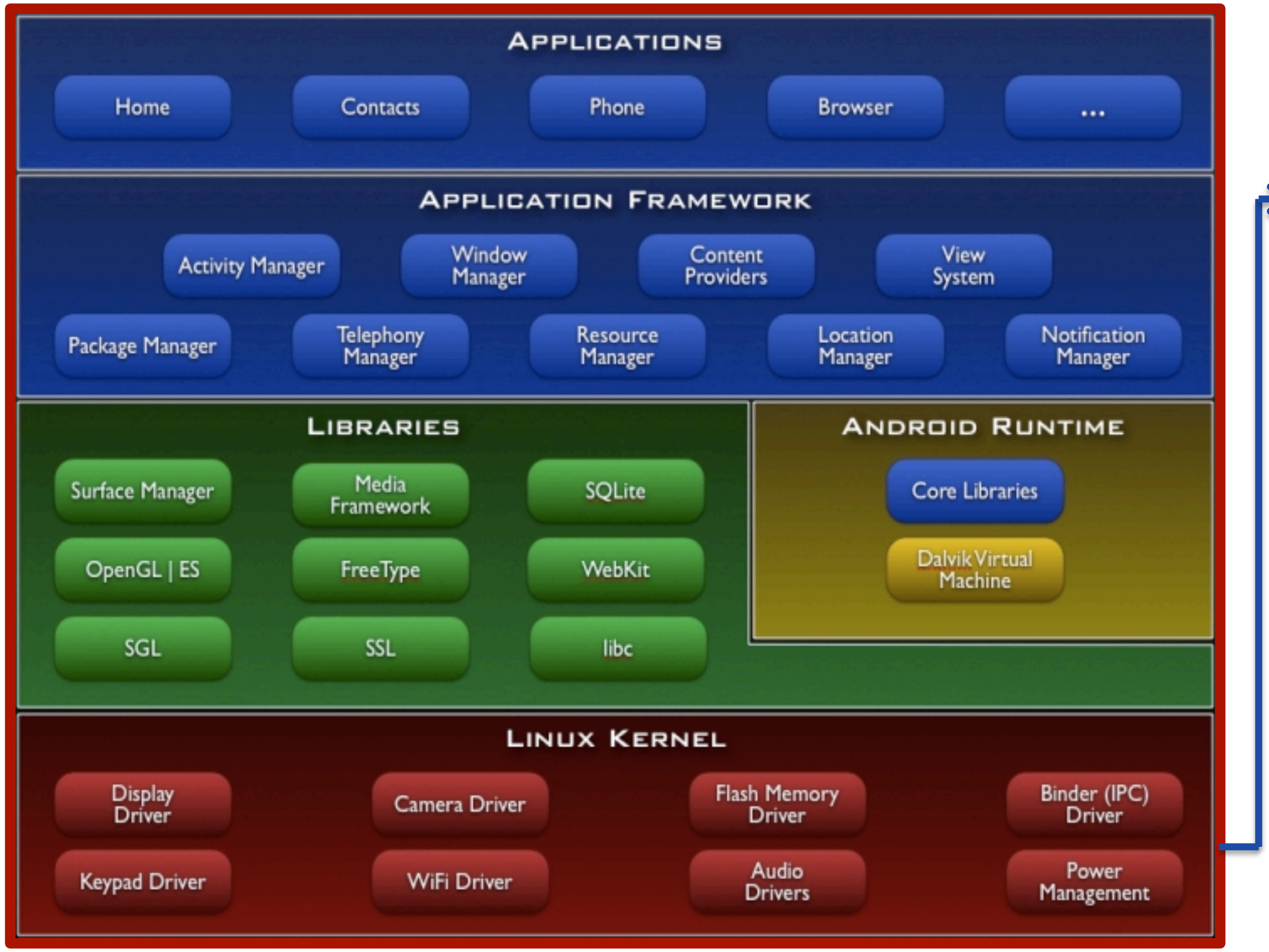
}

Stack
Architecture

Open Source Architecture
(Apache/MIT License v. 2.0)

Business-friendly License

The Android Architecture

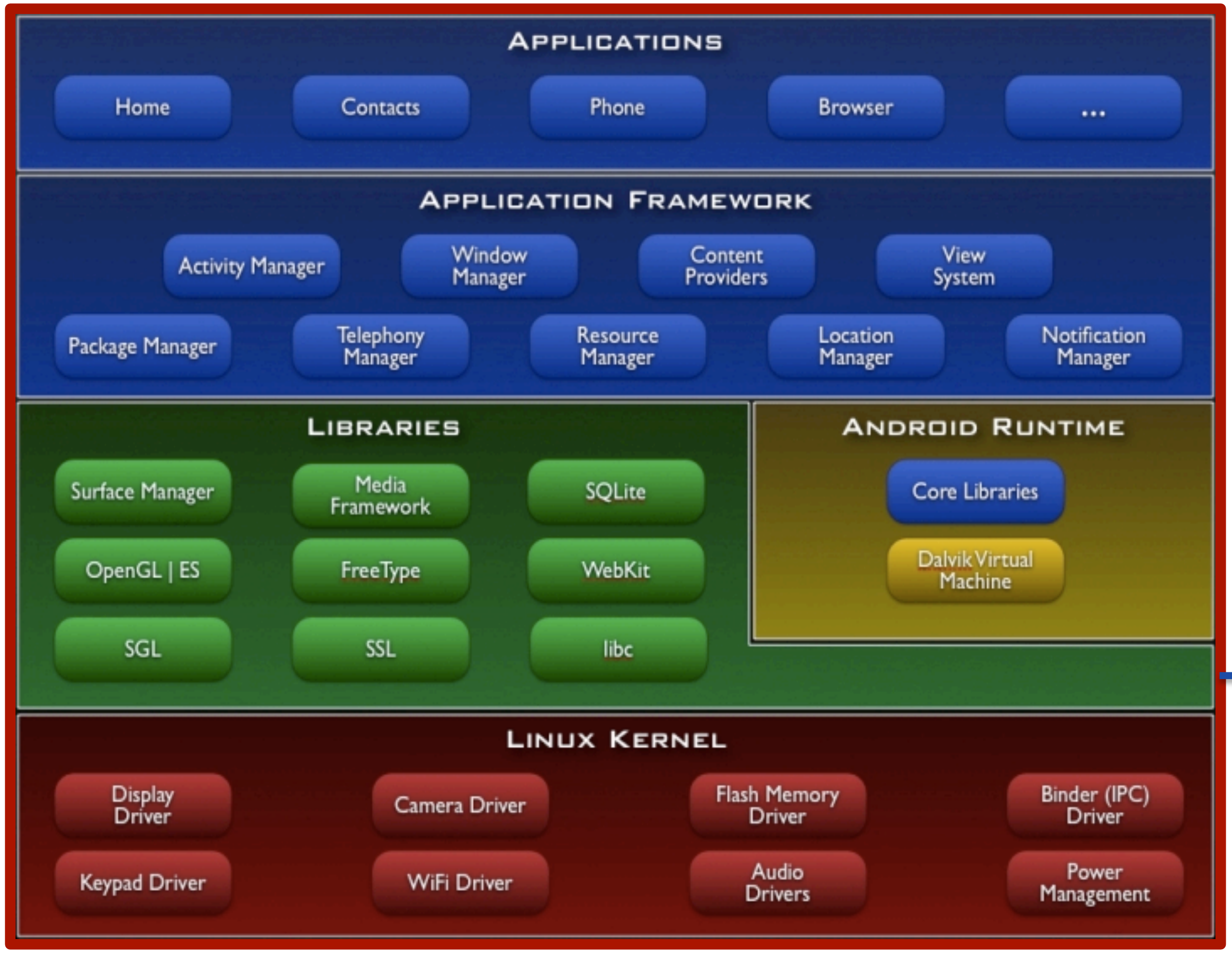


Built on top of **Linux kernel** (v. 2.6-3.14)

Advantages:

- **Portability** (i.e. easy to compile on different hardware architectures)
- **Security** (e.g. secure multi-process environment)
- **Power** Management

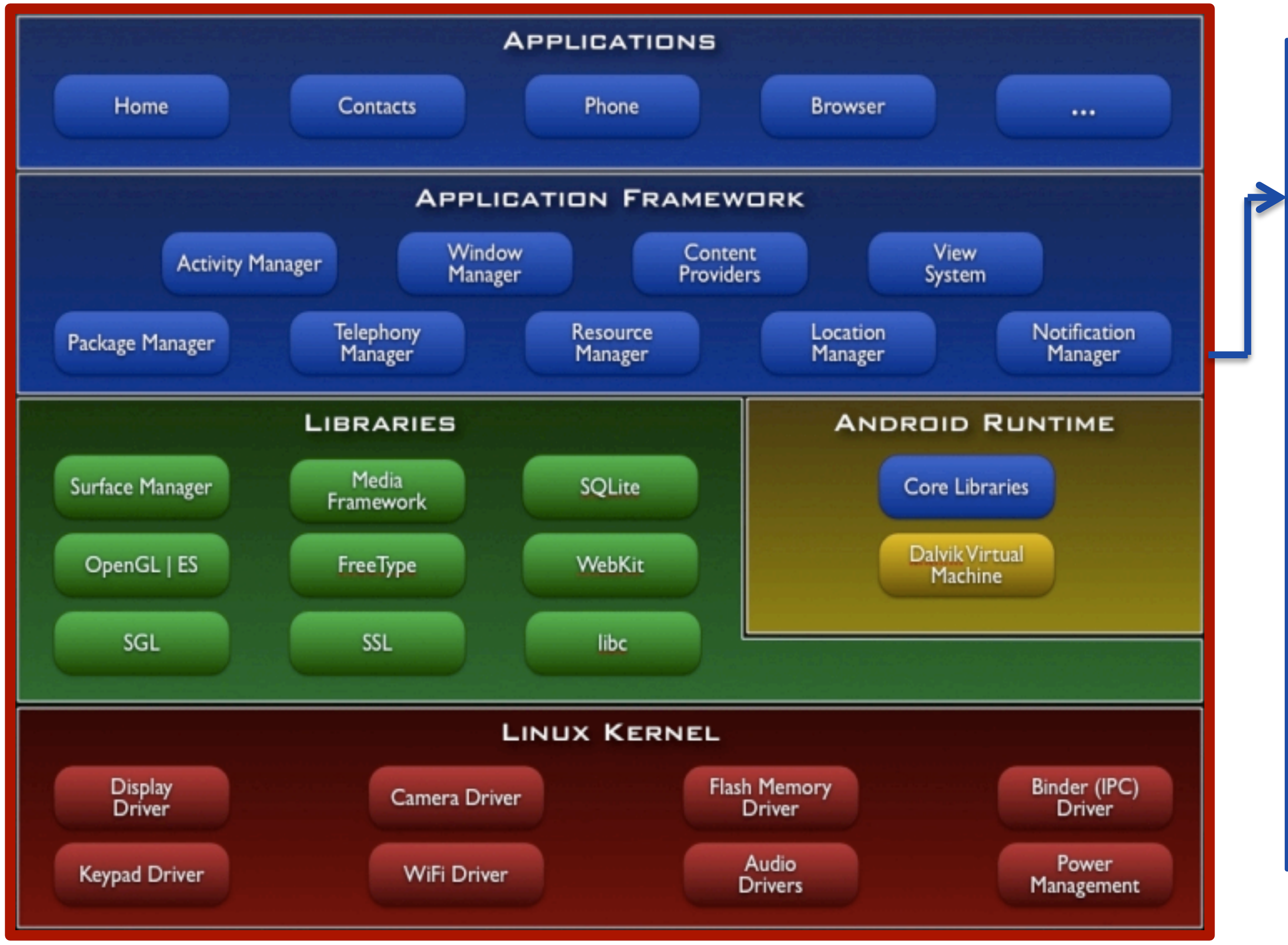
The Android Architecture



Native **Libraries**
(C/C++ code)

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

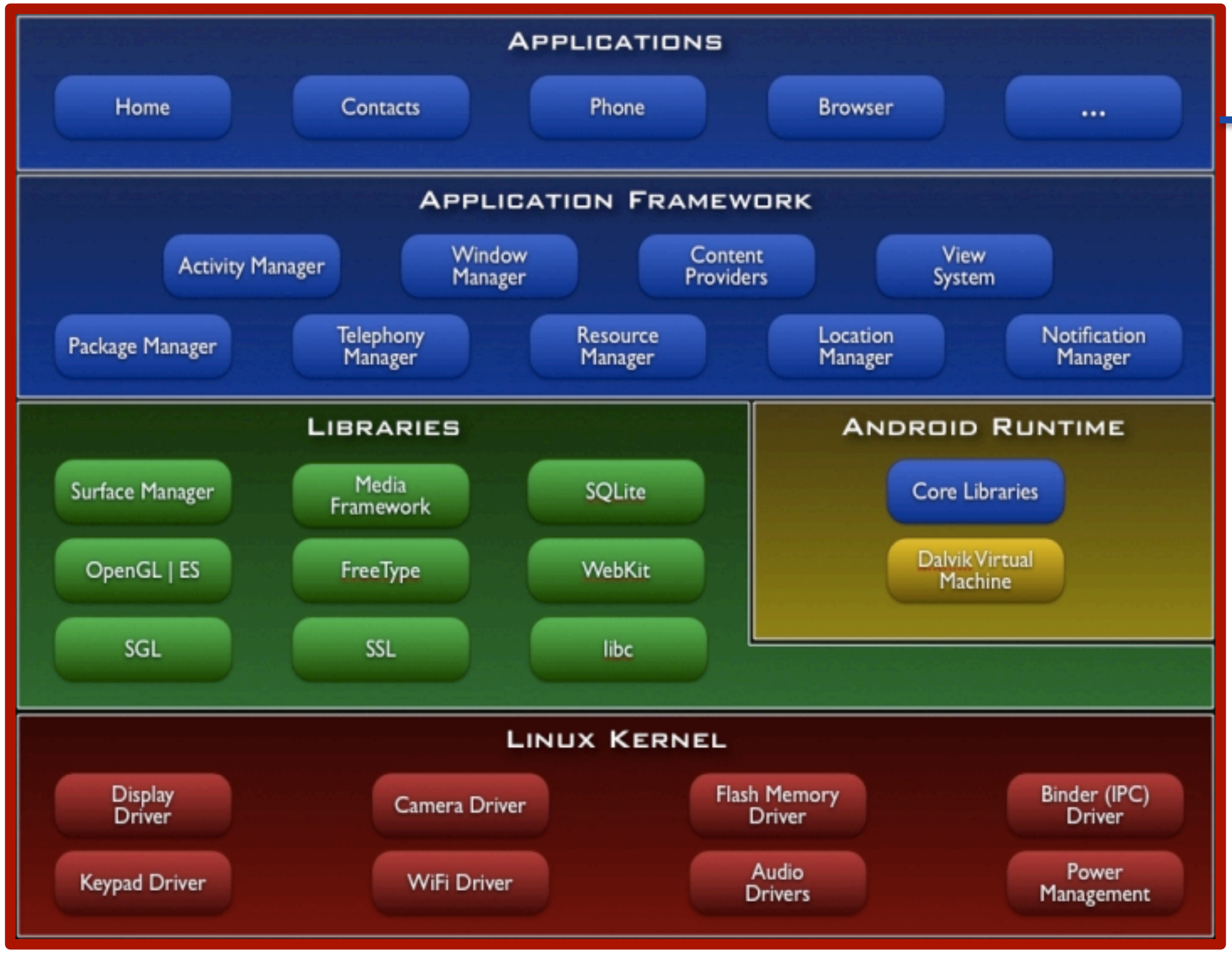
The Android Architecture



Application Libraries
(Core Components of Android)

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

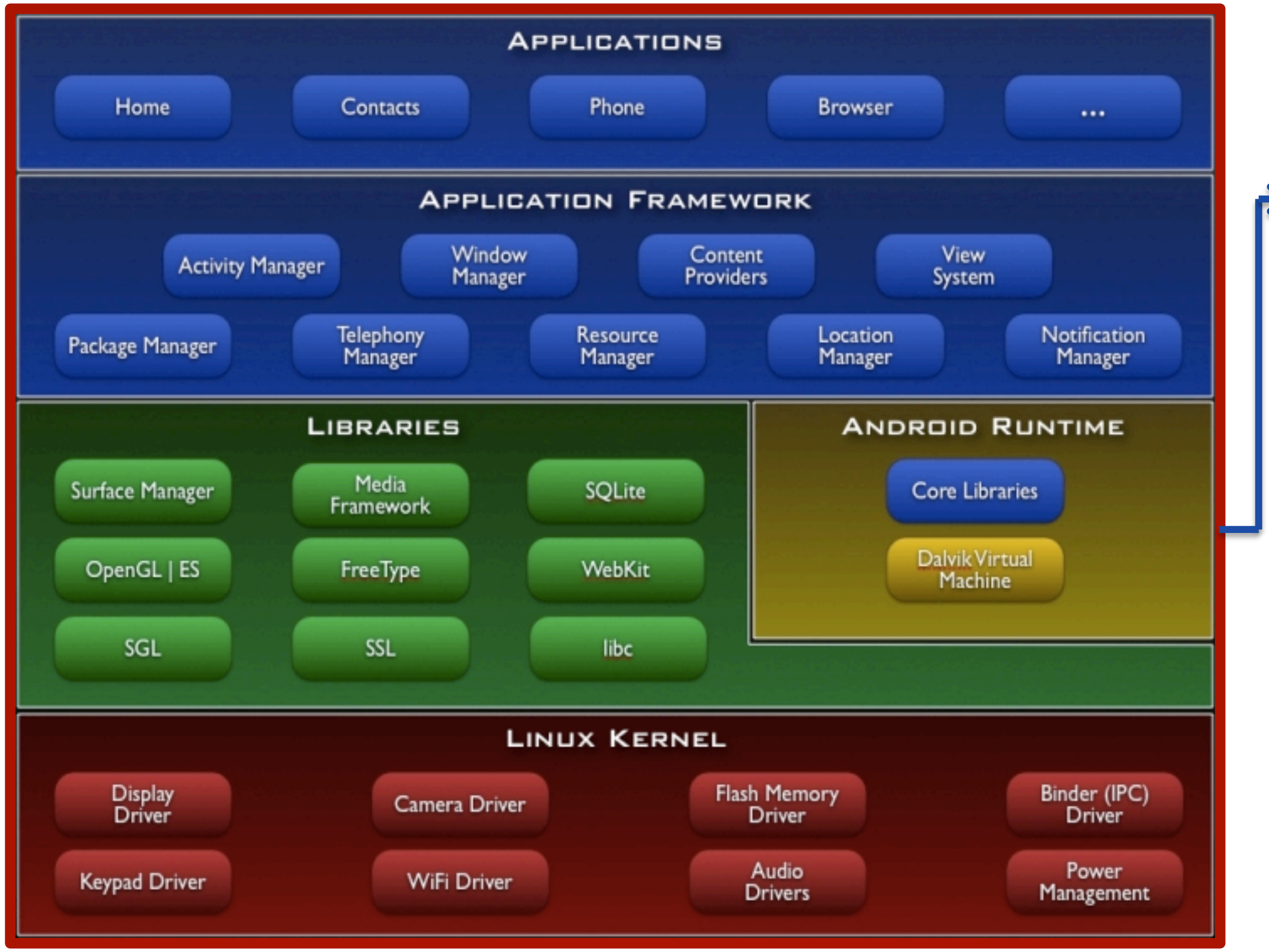
The Android Architecture



Applications
(Written in **Java** code)

- **Android Play Store**
- **Entertainment**
- **Productivity**
- **Personalization**
- **Education**
- **Geo-communication**
- **….**

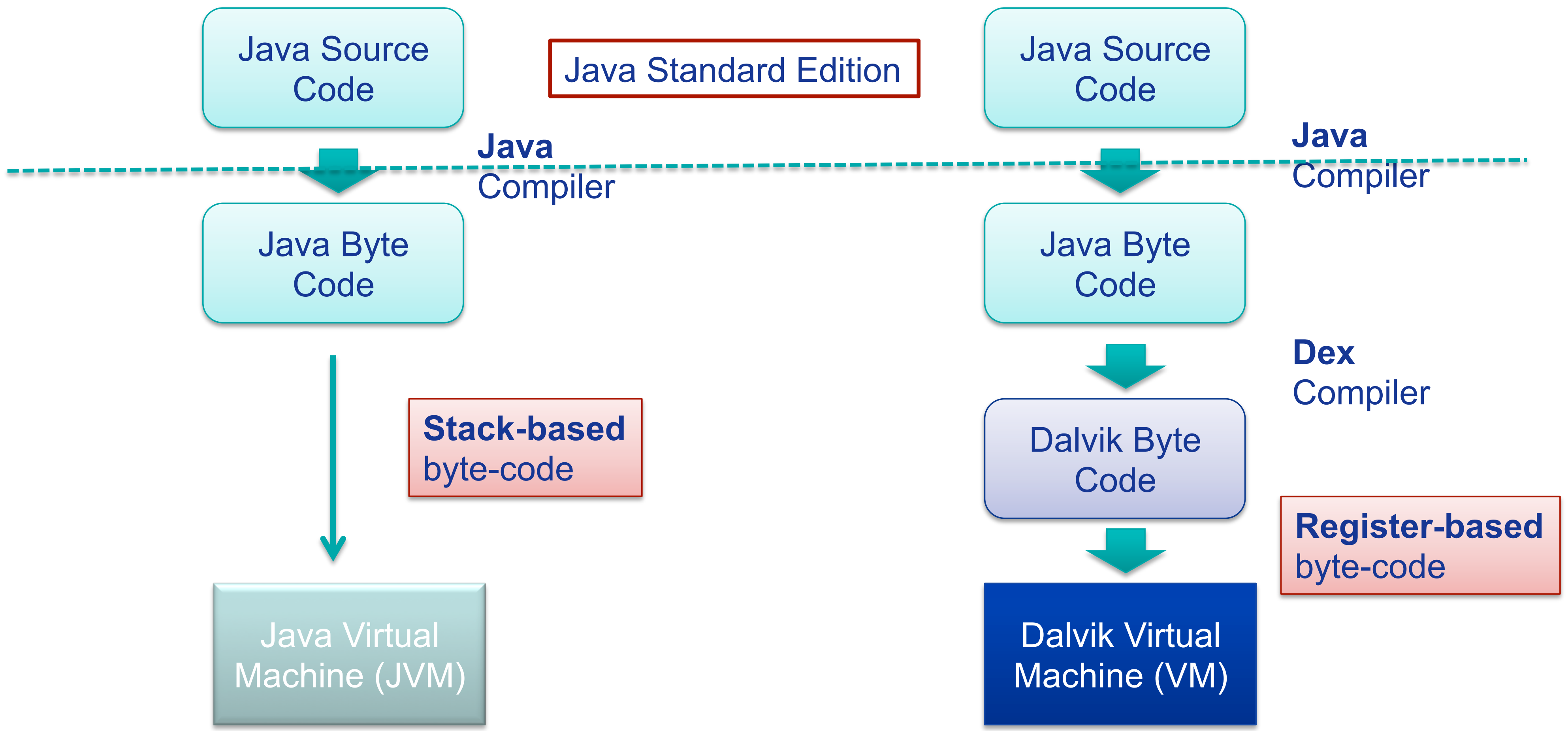
The Android Architecture



Dalvik Virtual Machine (VM)

- **Novel** Java Virtual Machine implementation (not using the Oracle JVM)
- Open **License** (Oracle JVM is not open!)
- **Optimized** for memory-constrained devices
- **Faster** than Oracle JVM
- ….

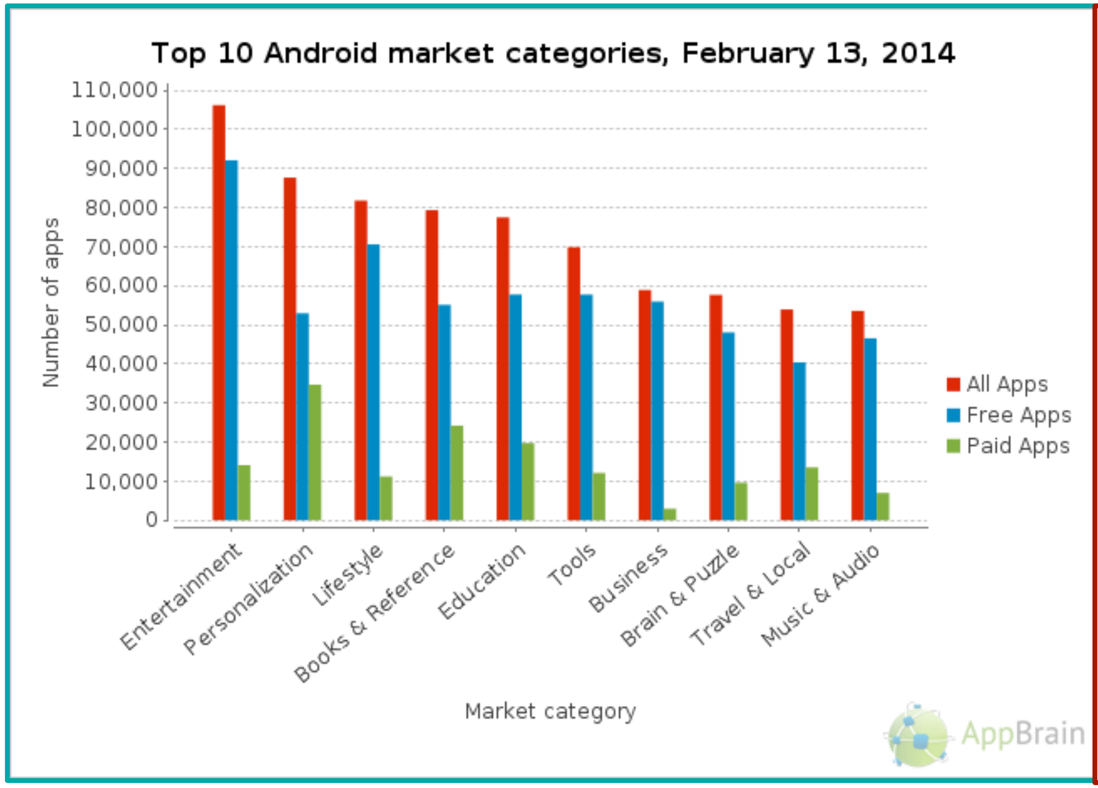
Dalvik Java Virtual Machine (JVM)

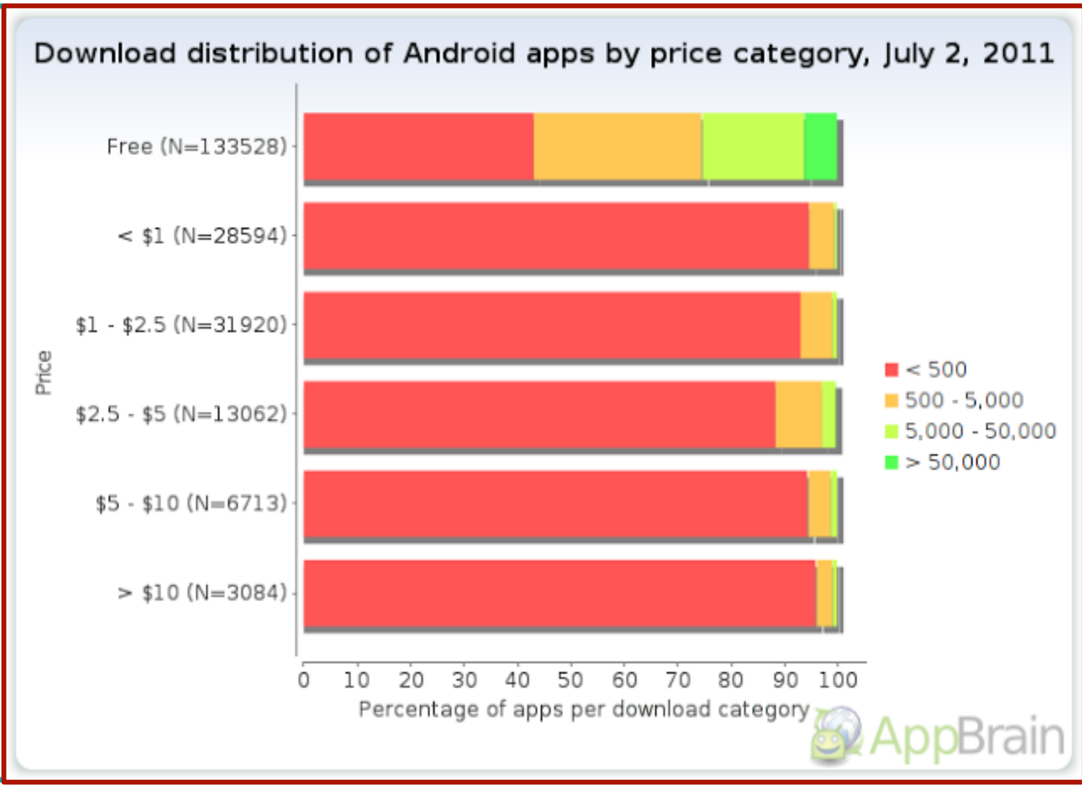


Android Applications

ANDROID APP CATEGORIES

ANDROID APP PRICE





http://www.appbrain.com/stats/android-market-app-categories

http://www.onlinemarketing-trends.com/2011/07/android-marketplace-top-5-statistics.html

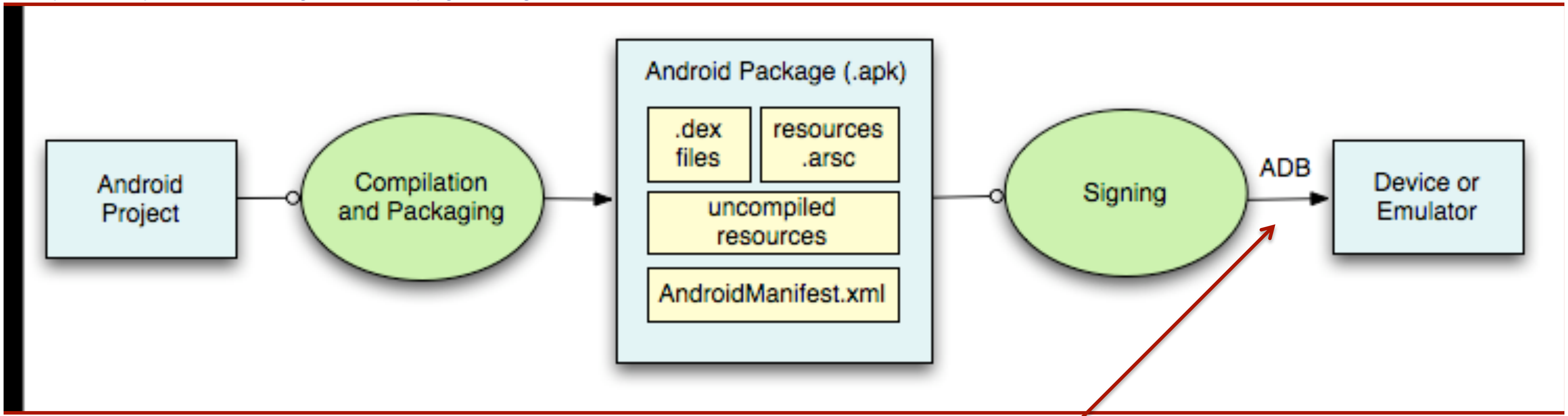
Android Applications Design

APPLICATION **DESIGN**:

- **GUI** Definition
- **Events** Management
- Application **Data** Management
- **Background** Operations
- **User** Notifications

Android Applications: Development

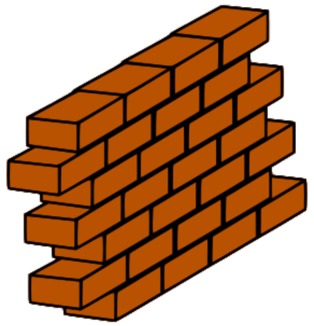
http://developer.android.com/guide/developing/building/index.html#detailed-build

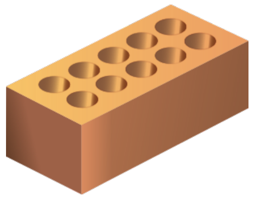


- **ADB** is a client server program that connects clients on developer machine to devices/emulators to facilitate development.
- An IDE like **Android Studio** handles the entire **development process**

Android Applications Design

- Developing an Android Application means using in a proper way the **Android basic components** …



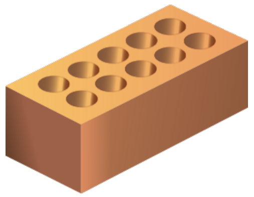


Activity

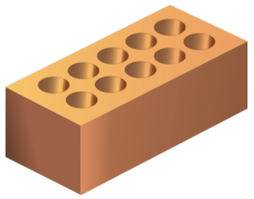
Fragment

Layout

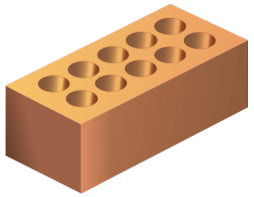
Views



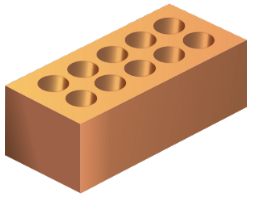
Intent



Content Providers



Service



Broadcast Receiver

Android Applications Design

- Beside using the basic components, an Android Application can rely on **system services** and **external libreries**

Android System Services

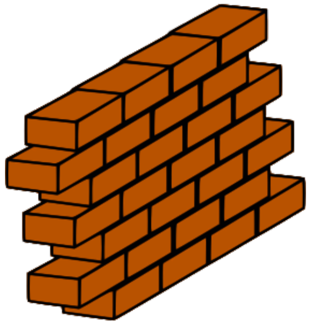
✧ WiFi Service
✧ Embedded Sensor Service
✧ Notification Manager Service
✧ …

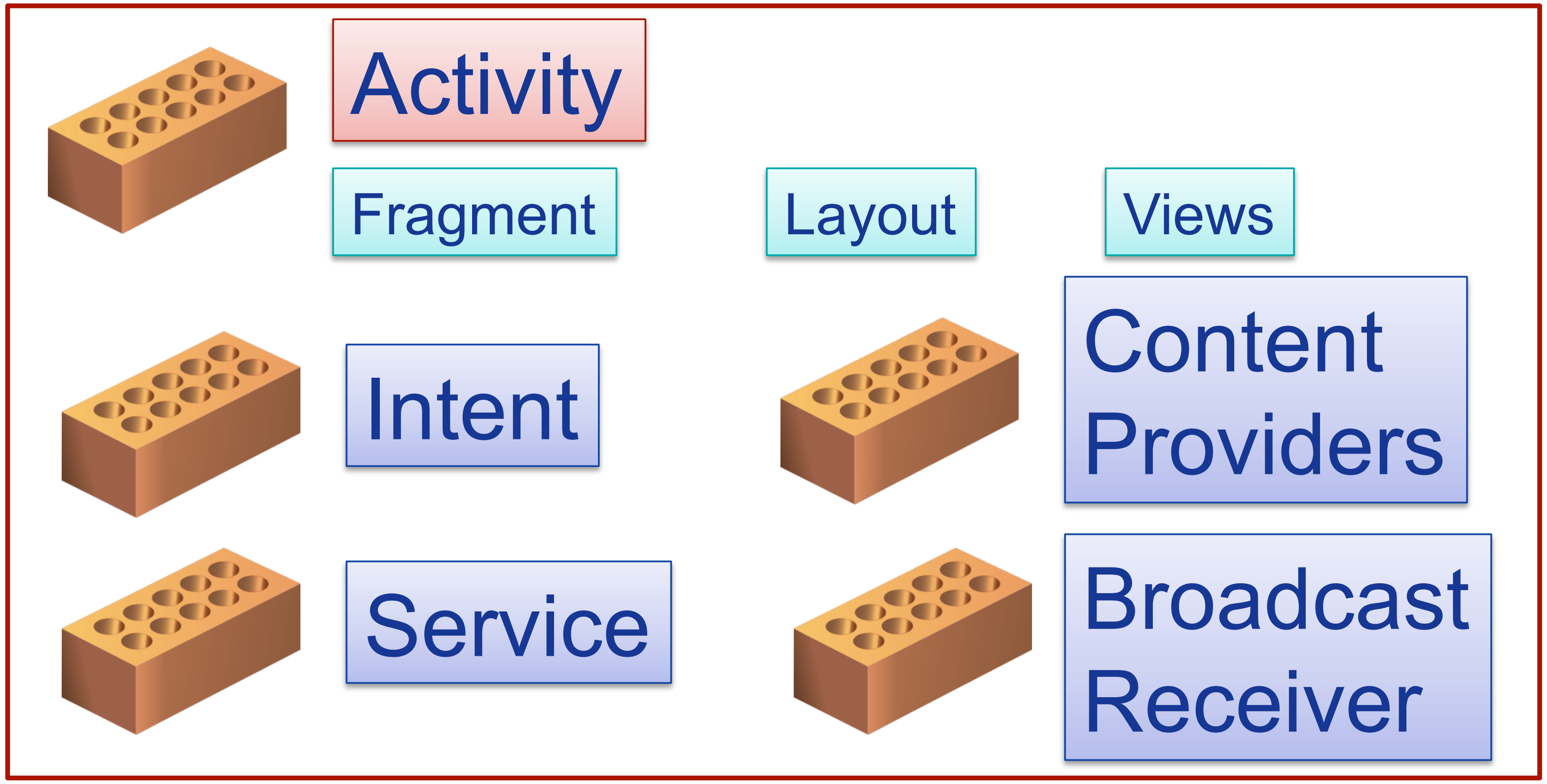
Google Play Libraries

✧ Google Maps API
✧ Activity Recognition API
✧ Google Cloud Messaging
✧ …

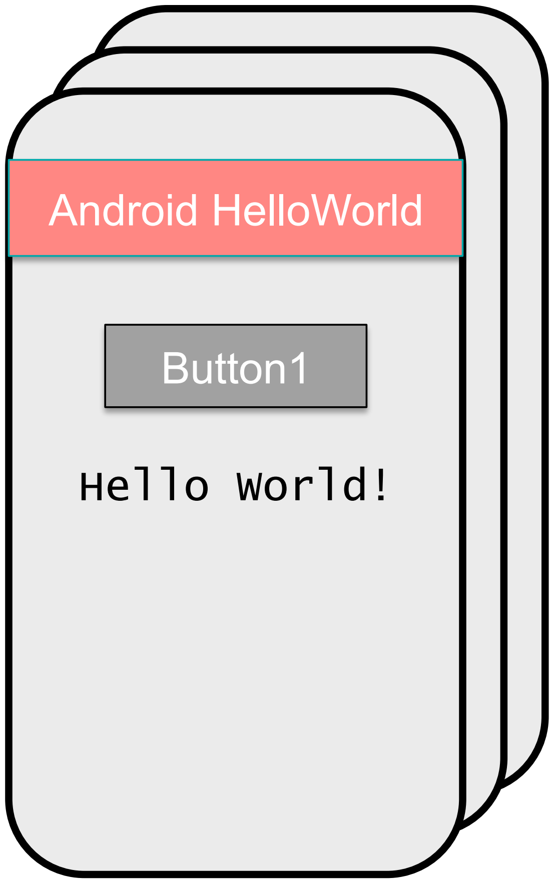
Android Applications Design

- Developing an Android Application means using in a proper way the **Android basic components** …





Android Components: Activities



- 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 exhange information one with each other.

Android Components: Activities

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

PROGRAMMATIC APPROACH

Example:

```
Button button=new Button (this);
TextView text= new TextView();
text.setText("Hello world");
```

Android Components: Activities

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

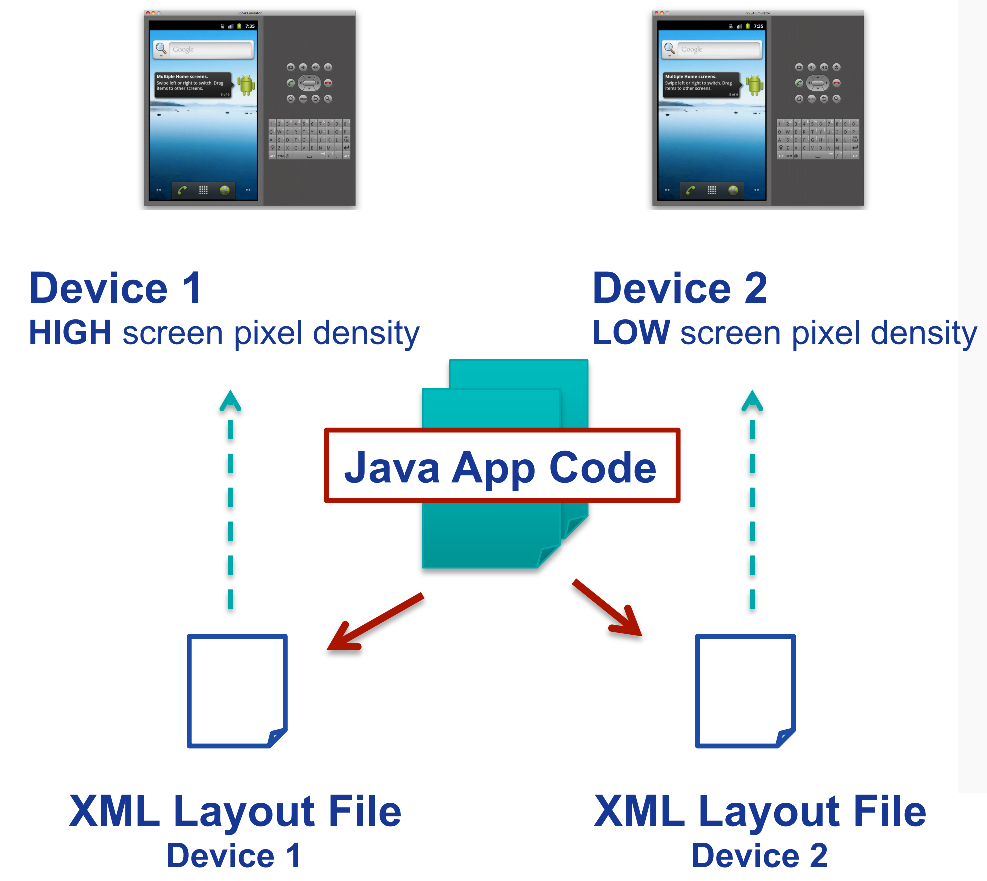
DECLARATIVE APPROACH

Example:

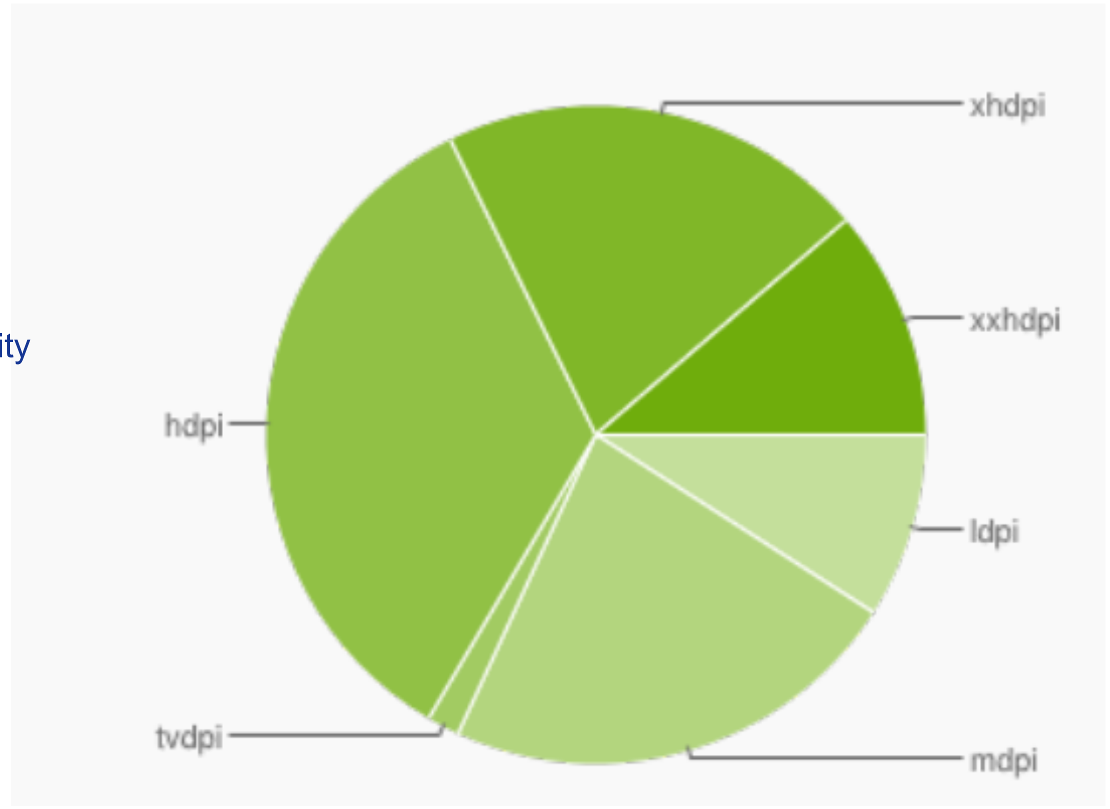
```
< TextView android.text=@string/hello" android:textcolor=@color/blue
android:layout_width="fill_parent" android:layout_height="wrap_content" />
< Button android.id="@+id/Button01" android:textcolor="@color/blue"
android:layout_width="fill_parent" android:layout_height="wrap_content" />
```

Android Components: Activities

EXAMPLE



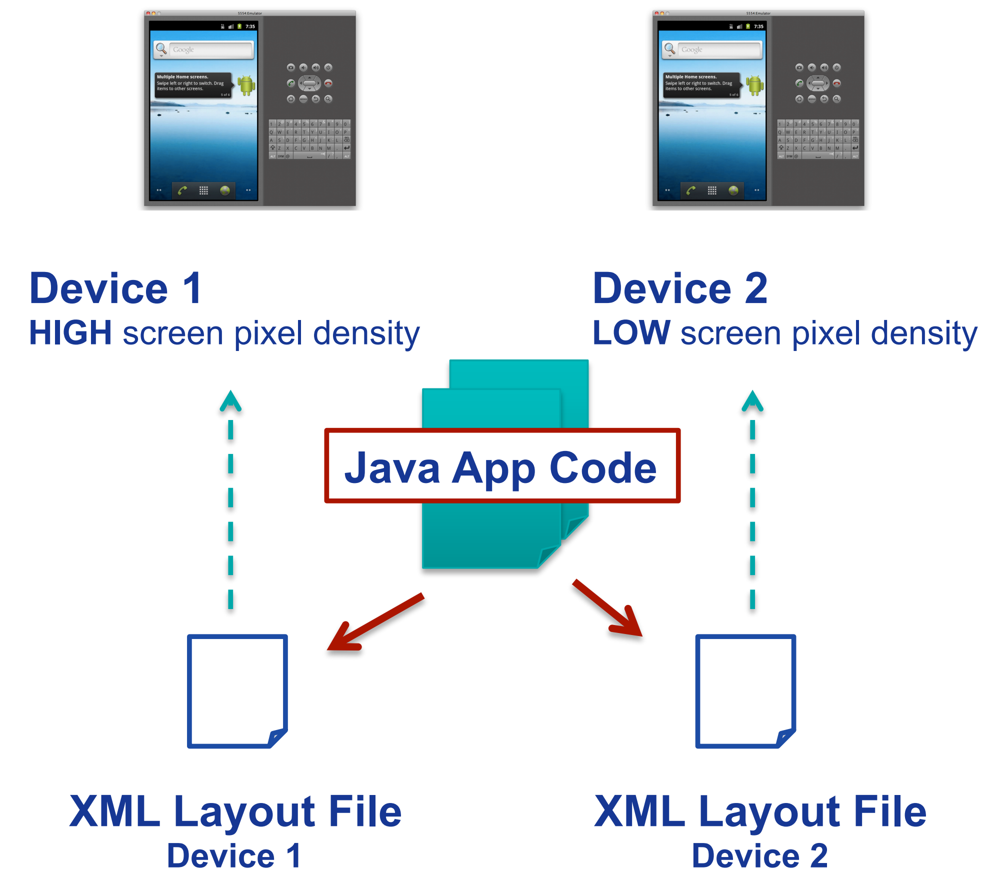
SCREEN CONFIGURATION DISTRIBUTION



http://developer.android.com/about/dashboards/index.html

Android Components: Activities

EXAMPLE



- Build the **application layout** through XML files (like HTML)
- Define **two** different XML **layouts** for two different devices
- At **runtime**, Android detects the current device configuration and loads the appropriate resources for the application
- **No need to recompile**!
- Just add a new XML file if you need to support a new device

Android Components: Activities

- *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

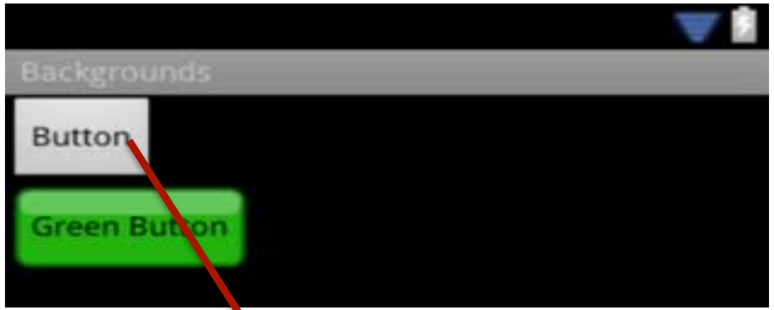
⇩

Java Code ⇨ Manages the **events**, and handles the **interaction** with the user.

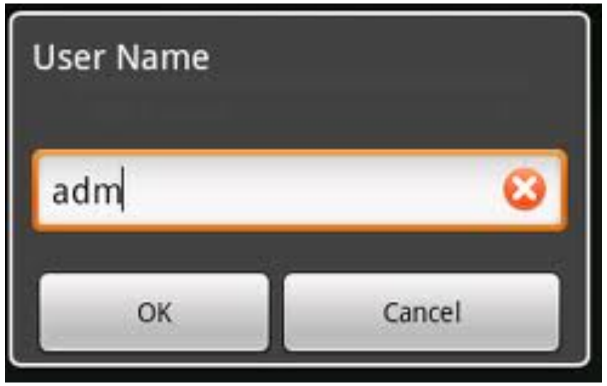
Android Components: Activities

- **Views** can generate **events** (caused by human interactions) that must be managed by the Android-developer.

Button



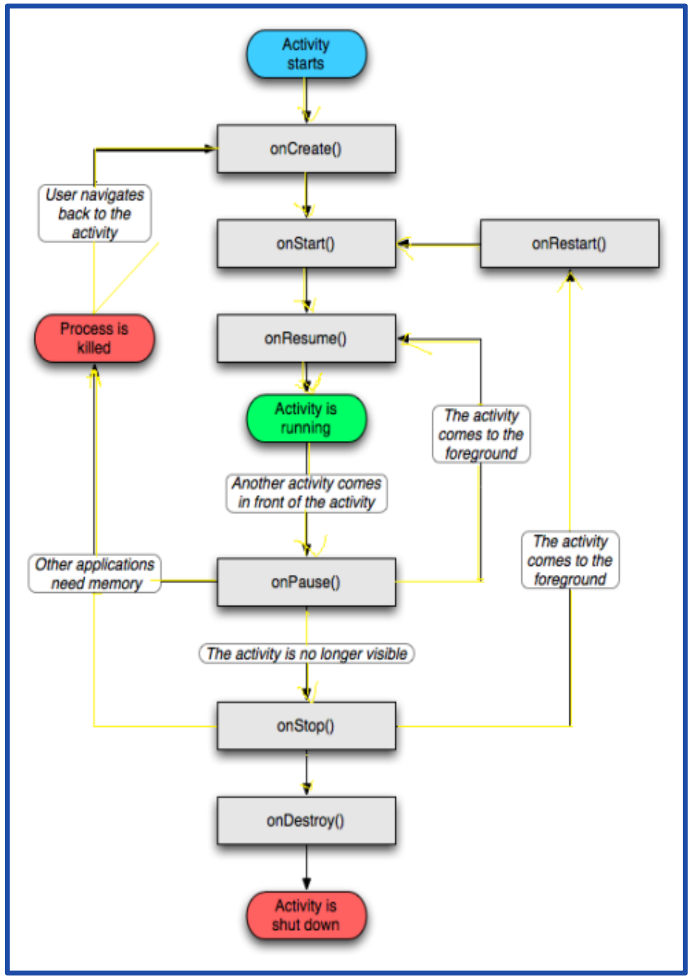
TextEdit



ESEMPIO

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

Android Components: Activities



Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

- The **Activity Manager** is responsible for creating, destroying, managing activities.
- Activities can be on different **states**: *starting*, *running*, *stopped*, *destroyed*, *paused*.
- Only one activity can be on the **running** state at a time.
- Activities are organized on a **stack**, and have an event-driven life cycle (details later …)

Android Components: Activities

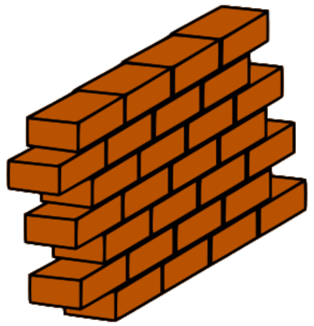
- Main difference between Android-programming and Java (Oracle) -programming:
  - **Mobile devices have constrained resource capabilities!**

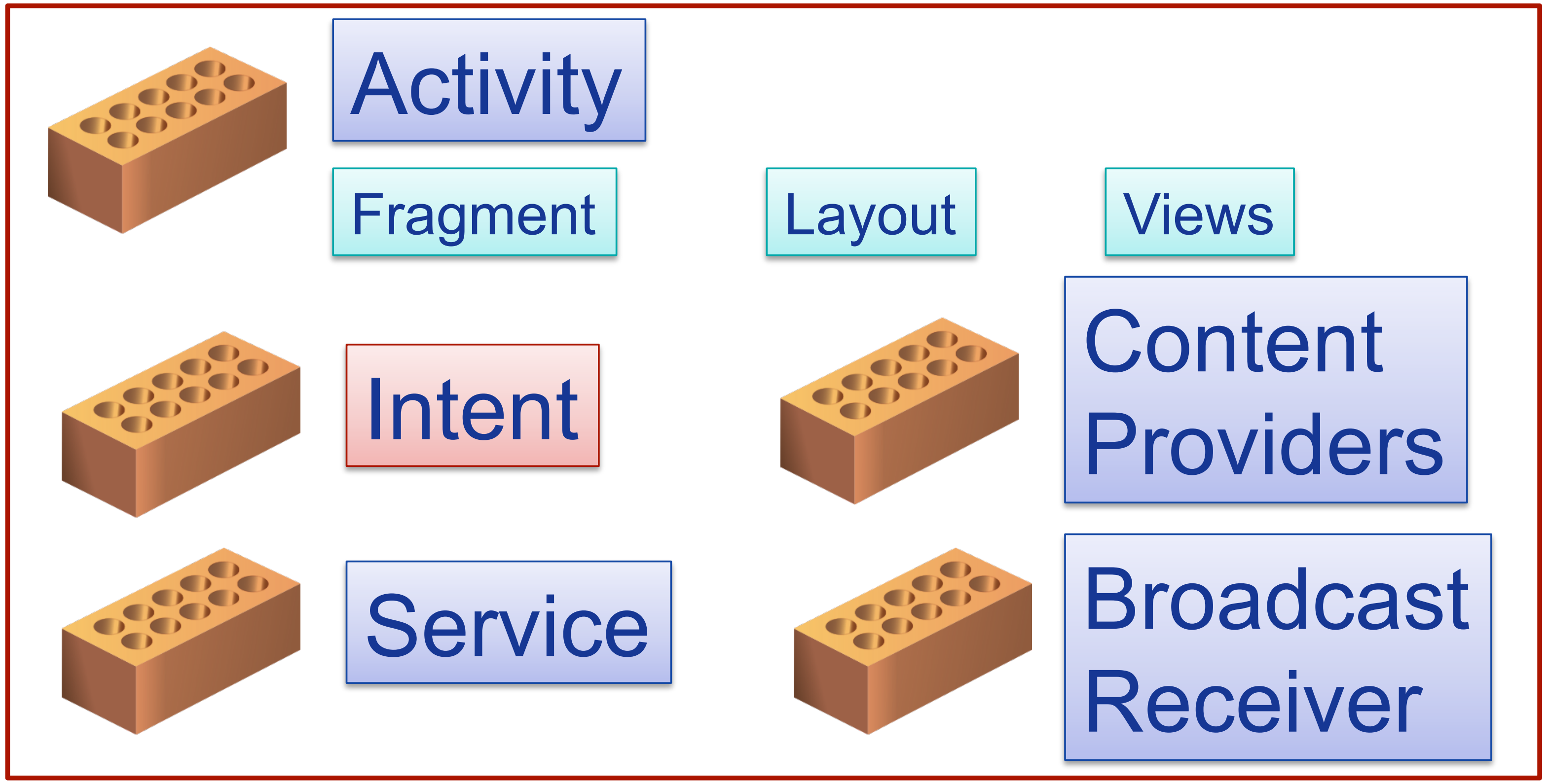
- Activity lifetime depends on **users' choice** (i.e. change of visibility) as well as on **system contraints** (i.e. memory shortage).

- Developer must implement **lifecycle methods** to account for state changes of each Activity …

Android Applications Design

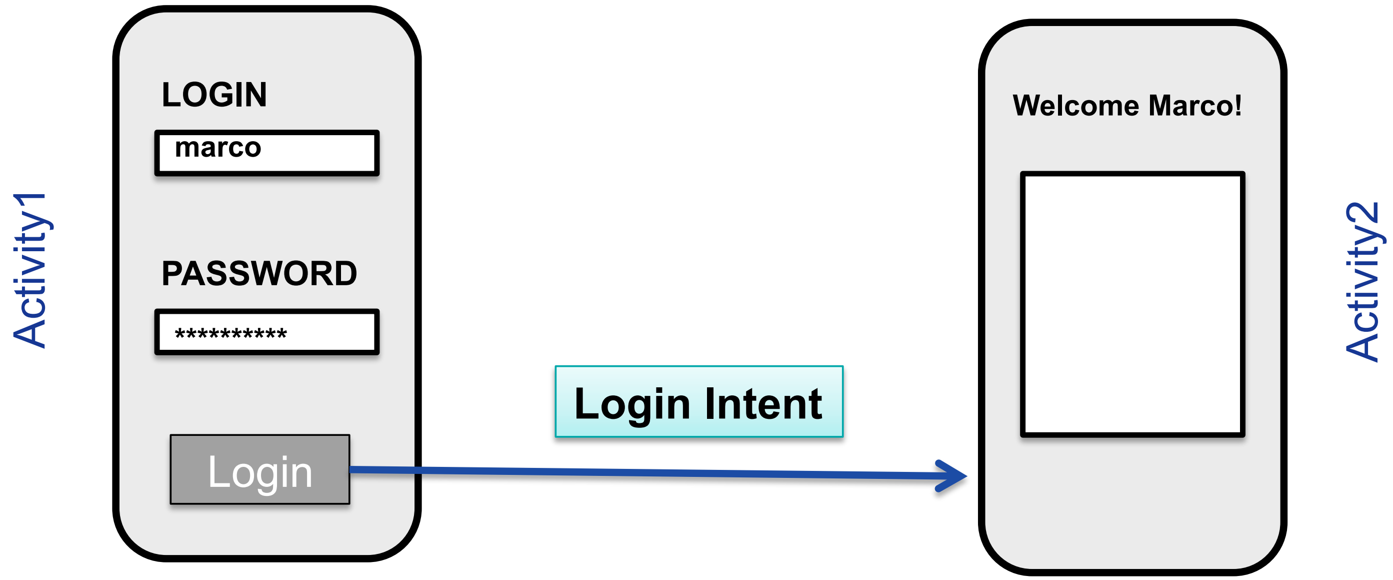
- Developing an Android Application means using in a proper way the **Android basic components** …





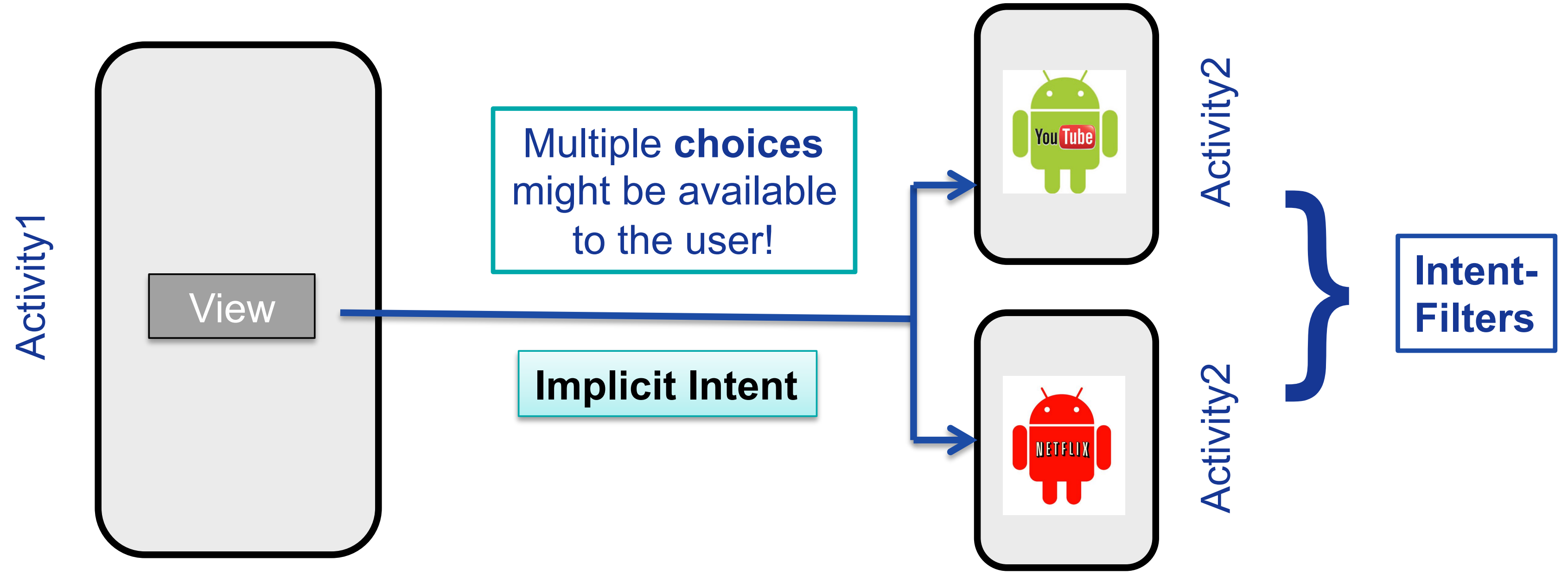
Android Components: Intents

- **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. Activity 2)*.



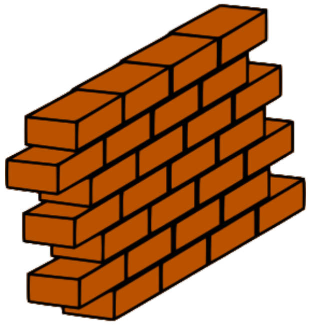
Android Components: Intents

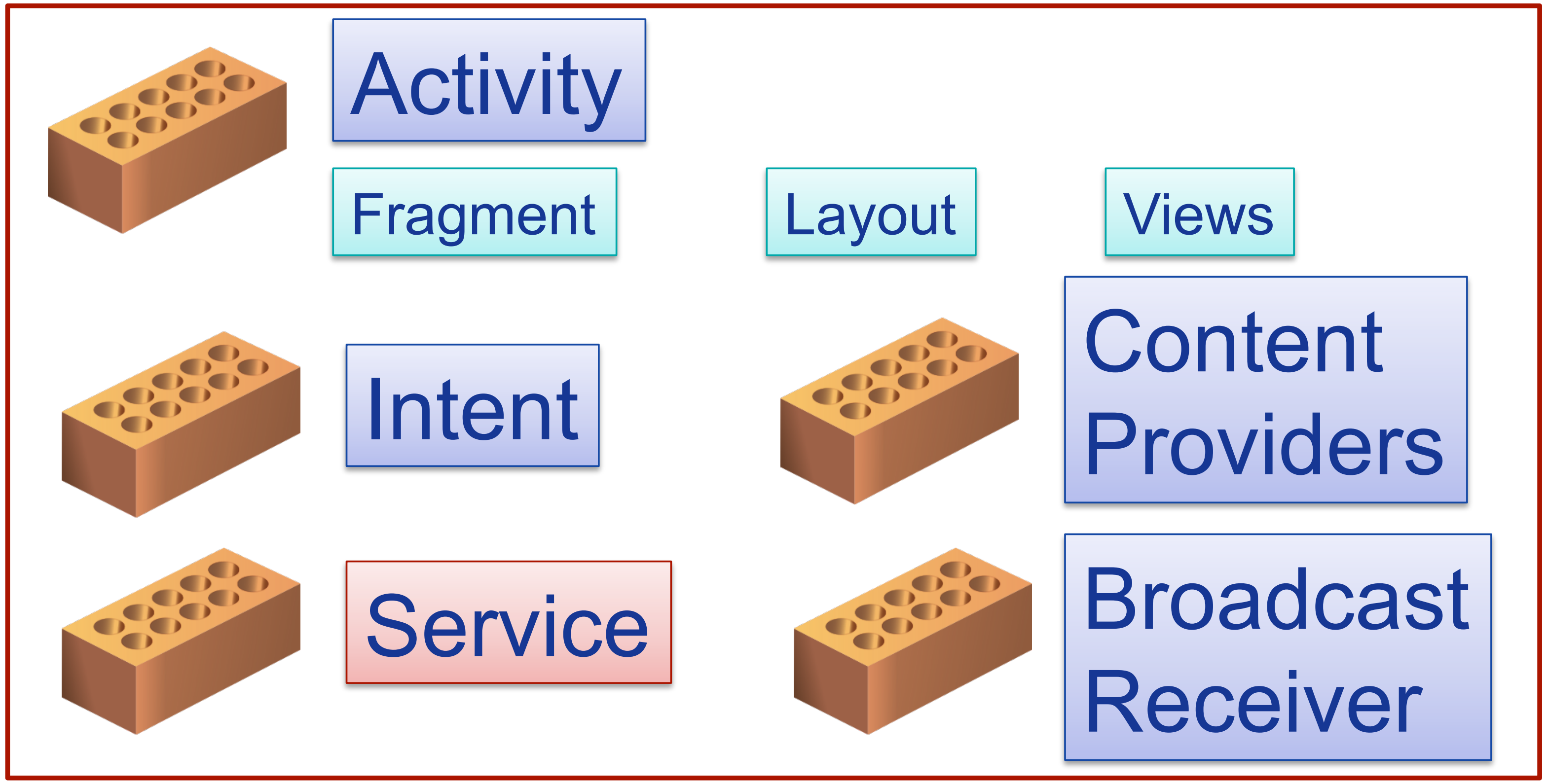
- **Intents**: asynchronous **messages** to activate core Android components (e.g. Activities).
- **Implicit** Intent → The component *(e.g. Activity1)* specifies the type of the intent *(e.g. "View a video")*.



Android Applications Design

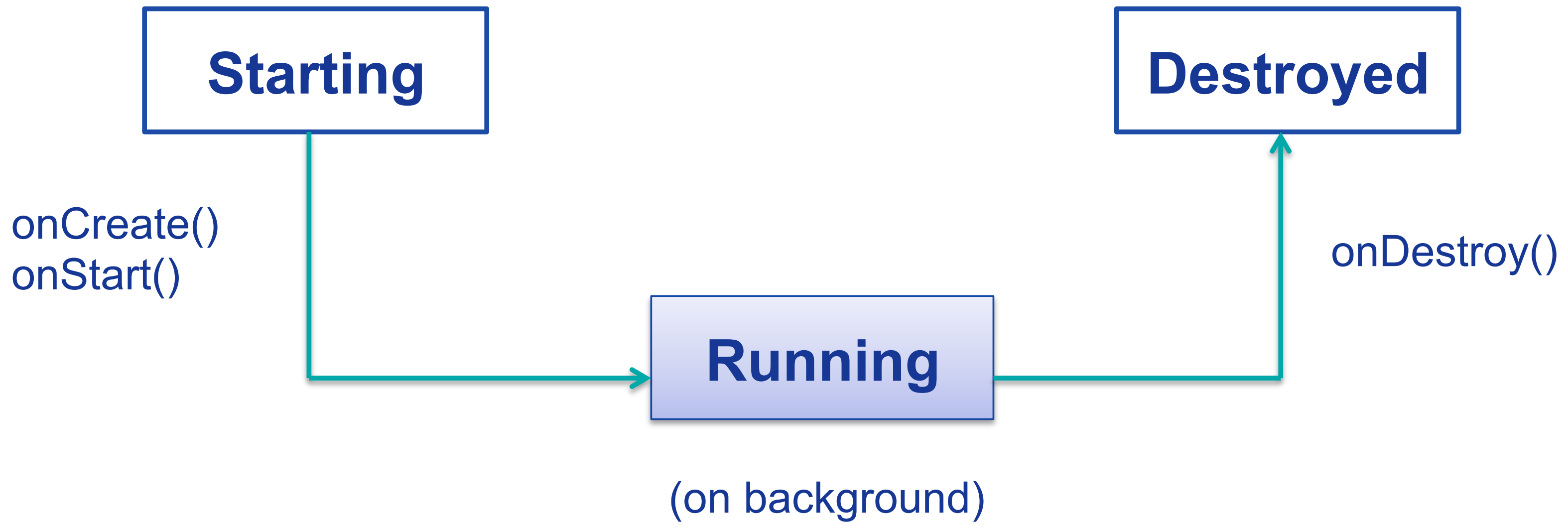
- Developing an Android Application means using in a proper way the **Android basic components** …





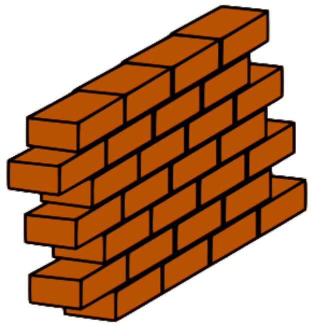
Android Components: Services

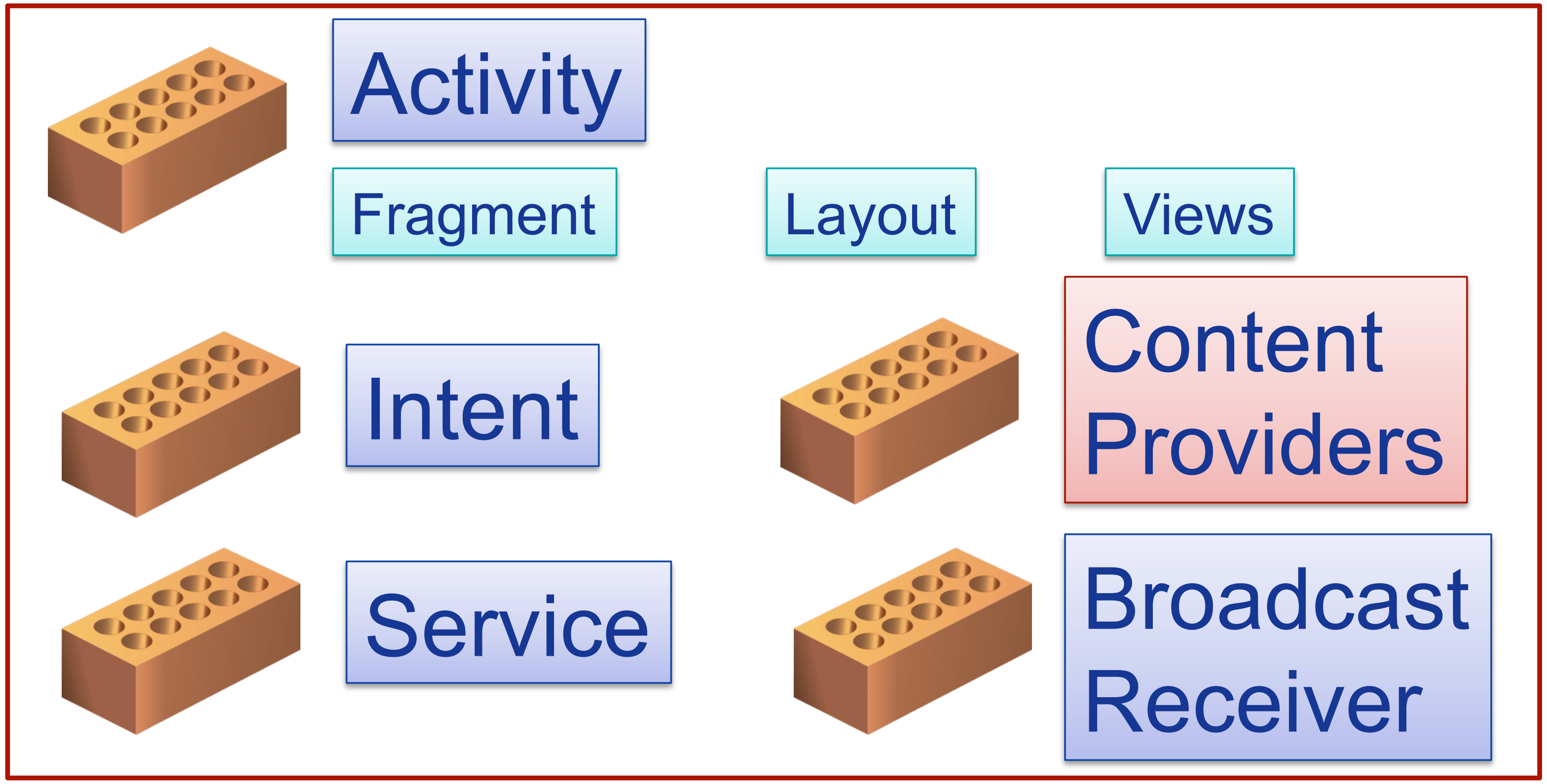
- **Services**: like Activities, but run in **background** and do not provide an user interface.
- Used for **non-interactive** tasks (e.g. networking).
- Service life-time composed of 3 states:



Android Applications Design

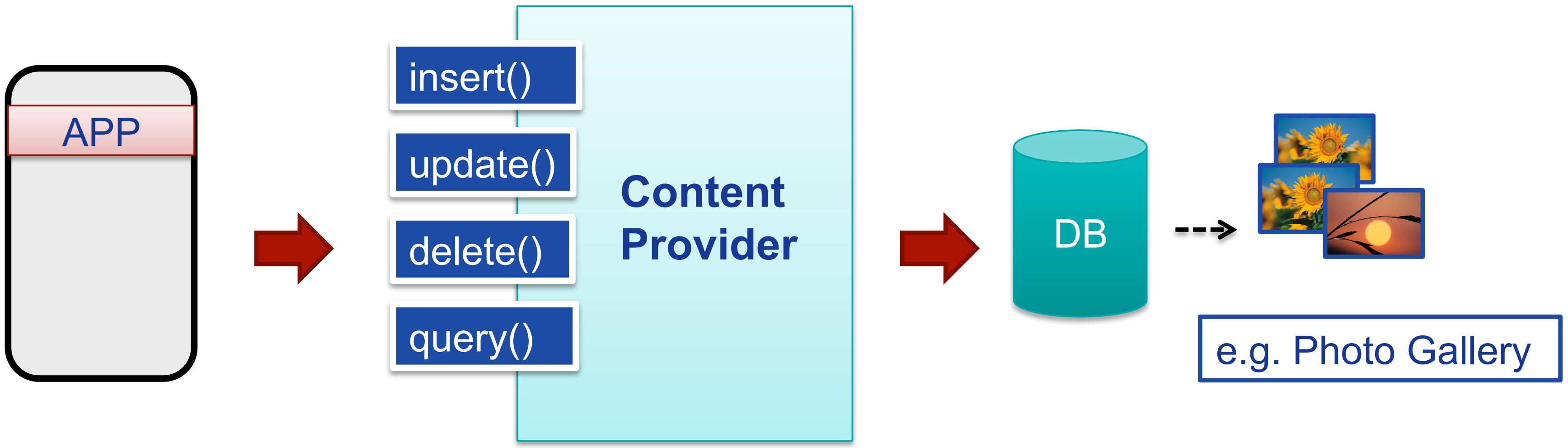
- Developing an Android Application means using in a proper way the **Android basic components** …





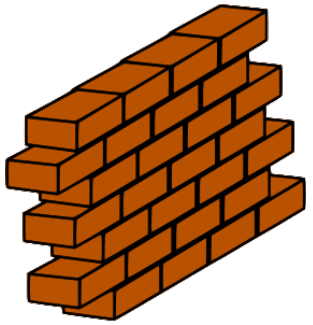
Android Components: Content Providers

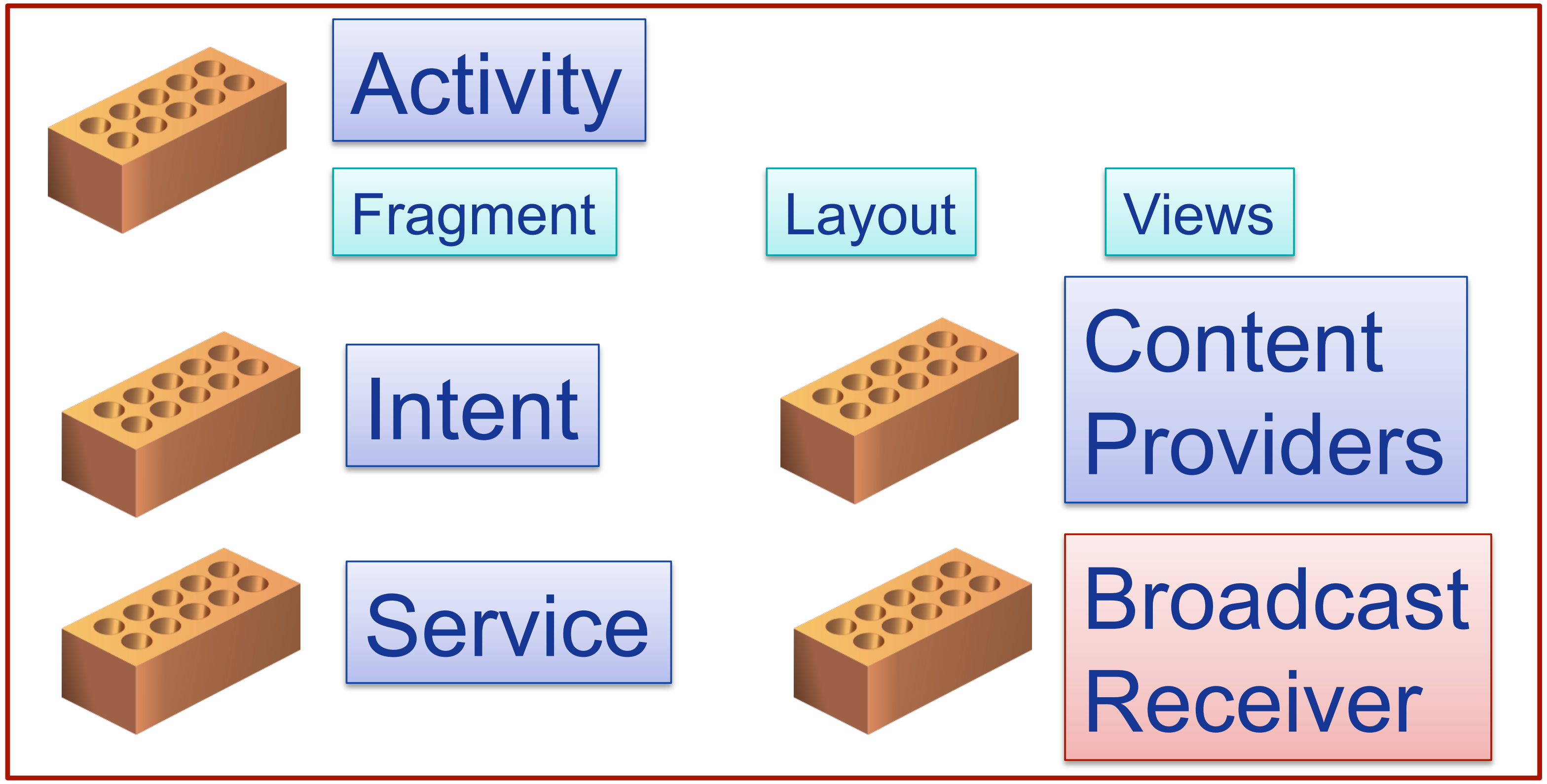
- Each Android **application** has its own **private** set of data (managed through *files* or through *SQLite* database).
- **Content Providers**: Standard **interface** to *access and share data among different applications*.



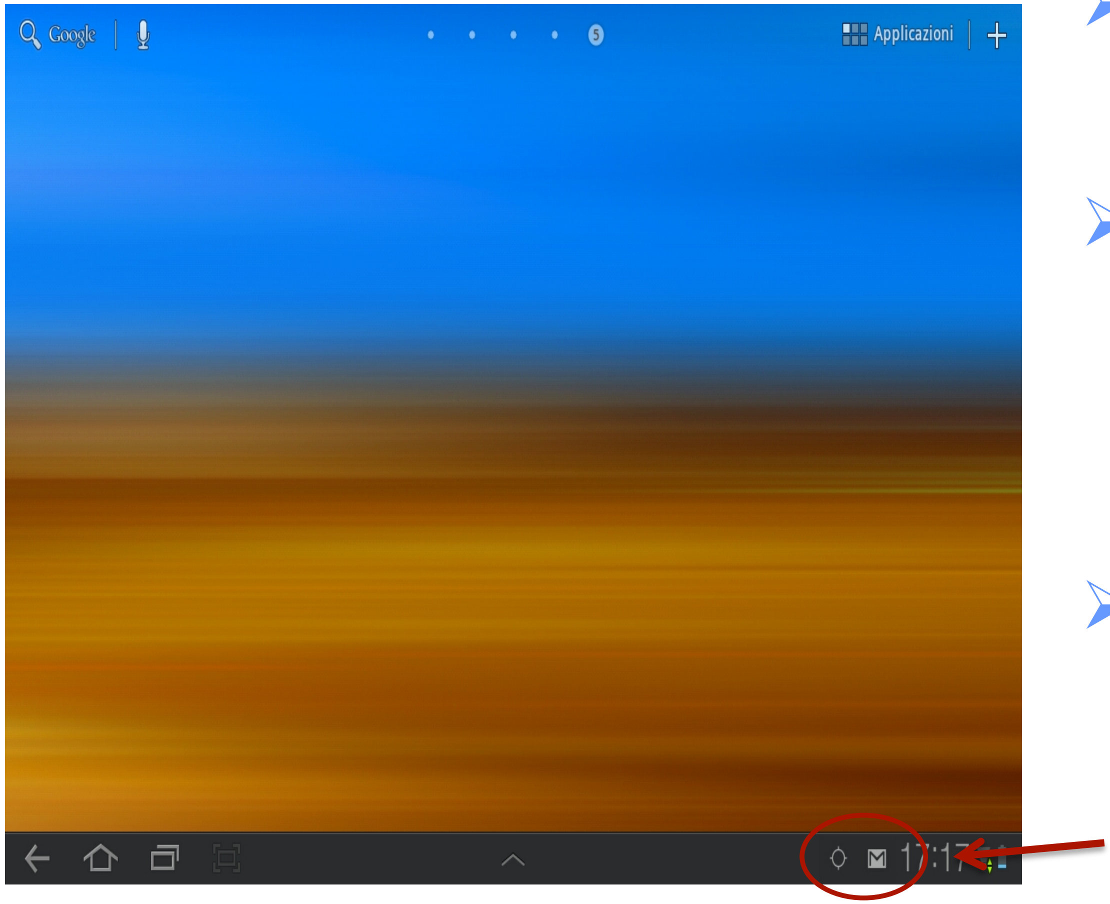
Android Applications Design

- Developing an Android Application means using in a proper way the **Android basic components** …





Android Components: Broadcast Receivers



- *Publish/Subscribe* paradigm
- **Broadcast Receivers**: An application can be signaled of **external events**.
- **Notification** types: Call incoming, SMS delivery, Wifi network detected, etc

Android Components: Broadcast Receivers

BROADCAST RECEIVER example

```
class WifiReceiver extends BroadcastReceiver {
        public void onReceive(Context c, Intent intent) {
            String s = new StringBuilder();
            wifiList = mainWifi.getScanResults();
            for(int i = 0; i < wifiList.size(); i++){
                s.append(new Integer(i+1).toString() + ".");
                s.append((wifiList.get(i)).toString());
                s.append("\\n");
            }
            mainText.setText(sb);
        }
    }
```

Android Components: System API

- Using the **components** described so far, Android applications can then leverage the system API …

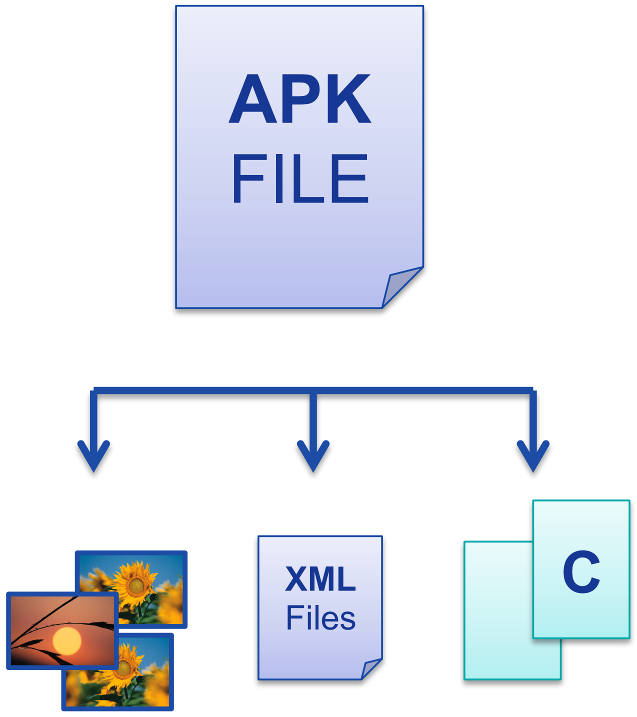
SOME EXAMPLEs …

- *Telephony Manager* data access (call, SMS, etc)
- *Sensor* management (GPS, accelerometer, etc)
- Network *connectivity* (Wifi, bluetooth, NFC, etc)
- *Web* surfing (HTTP client, WebView, etc)
- *Storage* management (files, SQLite db, etc)
- ….

Android Components: Google API

- … or easily interface with other **Google services**:

Android Application Distribution



- Each Android **application** is contained on a single **APK** file.
  - Java **Byte-code** (*compiled for Dalvik JVM*)
  - **Resources** (e.g. images. videos, XML layout files)
  - **Libraries** (optimal native C/C++ code)

Android Application **Distribution**

- Each application must be signed through a **key** before being distributed.

- Applications can be **distributed** via *Web* or via *Stores*.

- **Android Play Store:** application store run by Google … but several other application stores are available (they are just normal applications).

Android Application Security

- 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** during the installation.

ANDROIDMANIFEST.XML

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

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