SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

19ITE310 - MOBILE APPLICATION DEVELOPMENT

III YEAR - VI SEM

UNIT 1 – GETTING STARTED WITH MOBILITY

TOPIC 1 – Mobility landscape

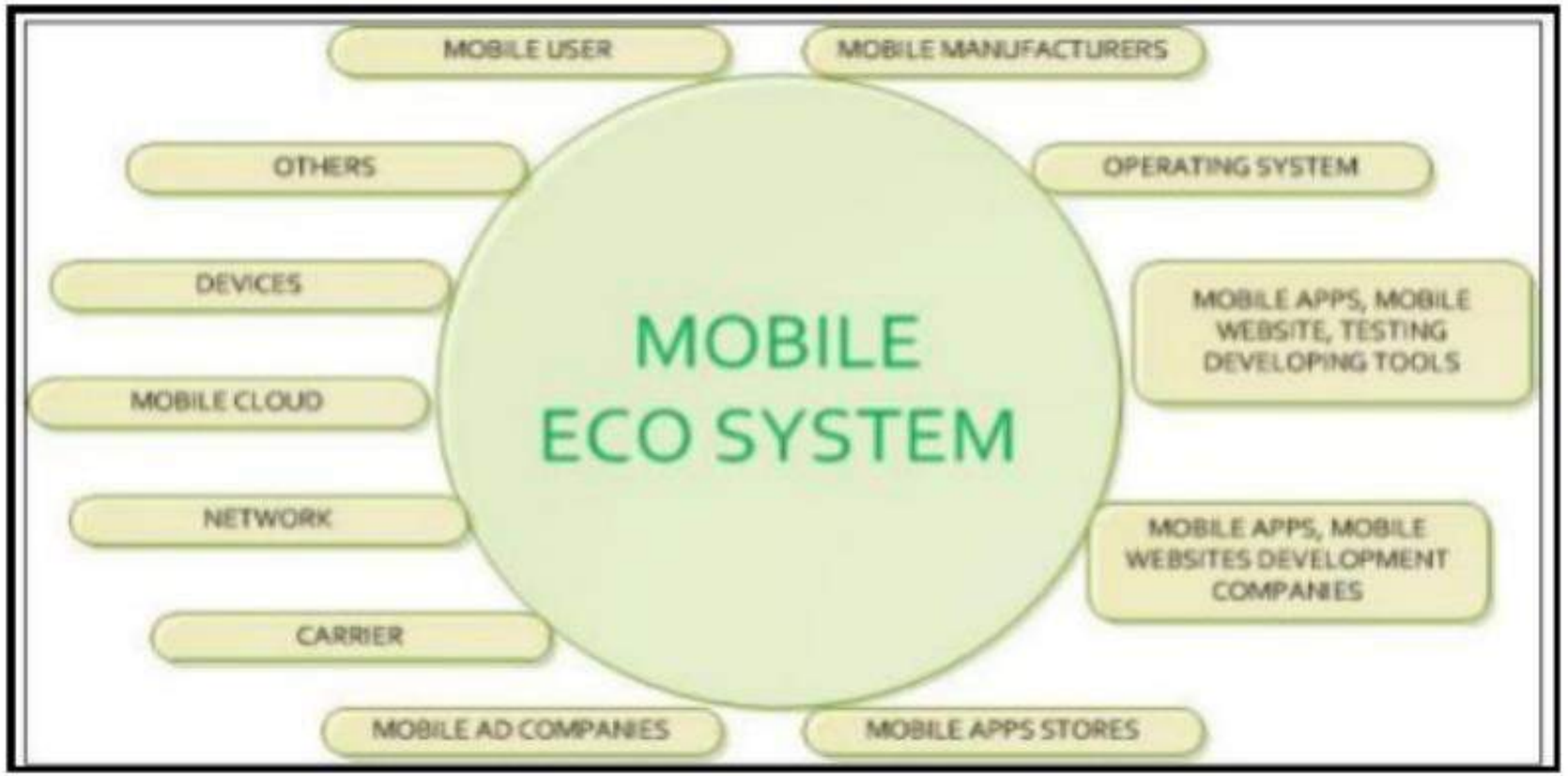
UNIT – 1
GETTING STARTED WITH MOBILITY

Mobility landscape – Mobile platforms - Mobile apps development – Overview of Android platform – setting up the mobile app development environment along with an emulator– a case study on Mobile app development

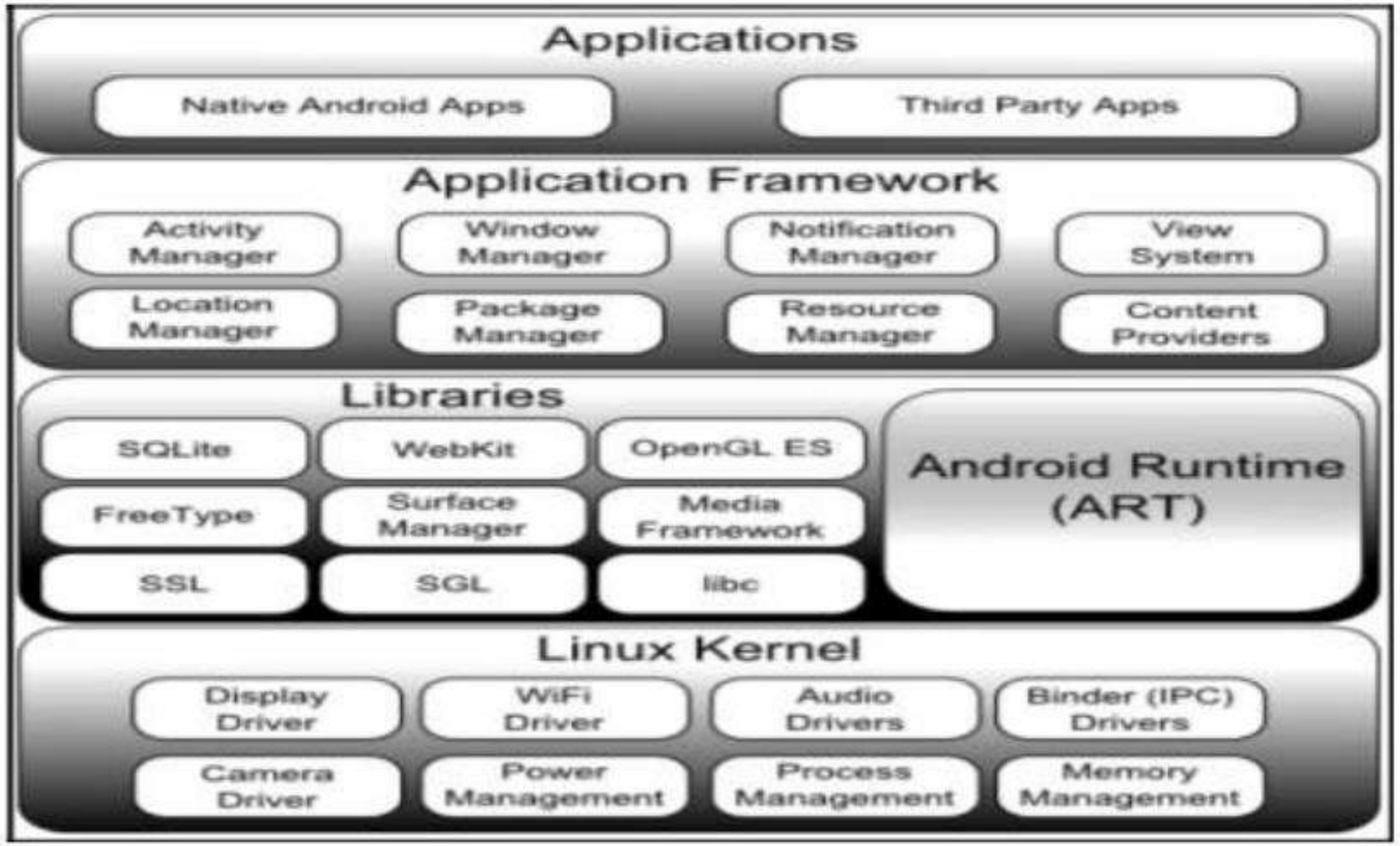
Lab Experiments:

1. Installation and setup of Android studio (mobile app development environment)
2. Development of Hello World Application

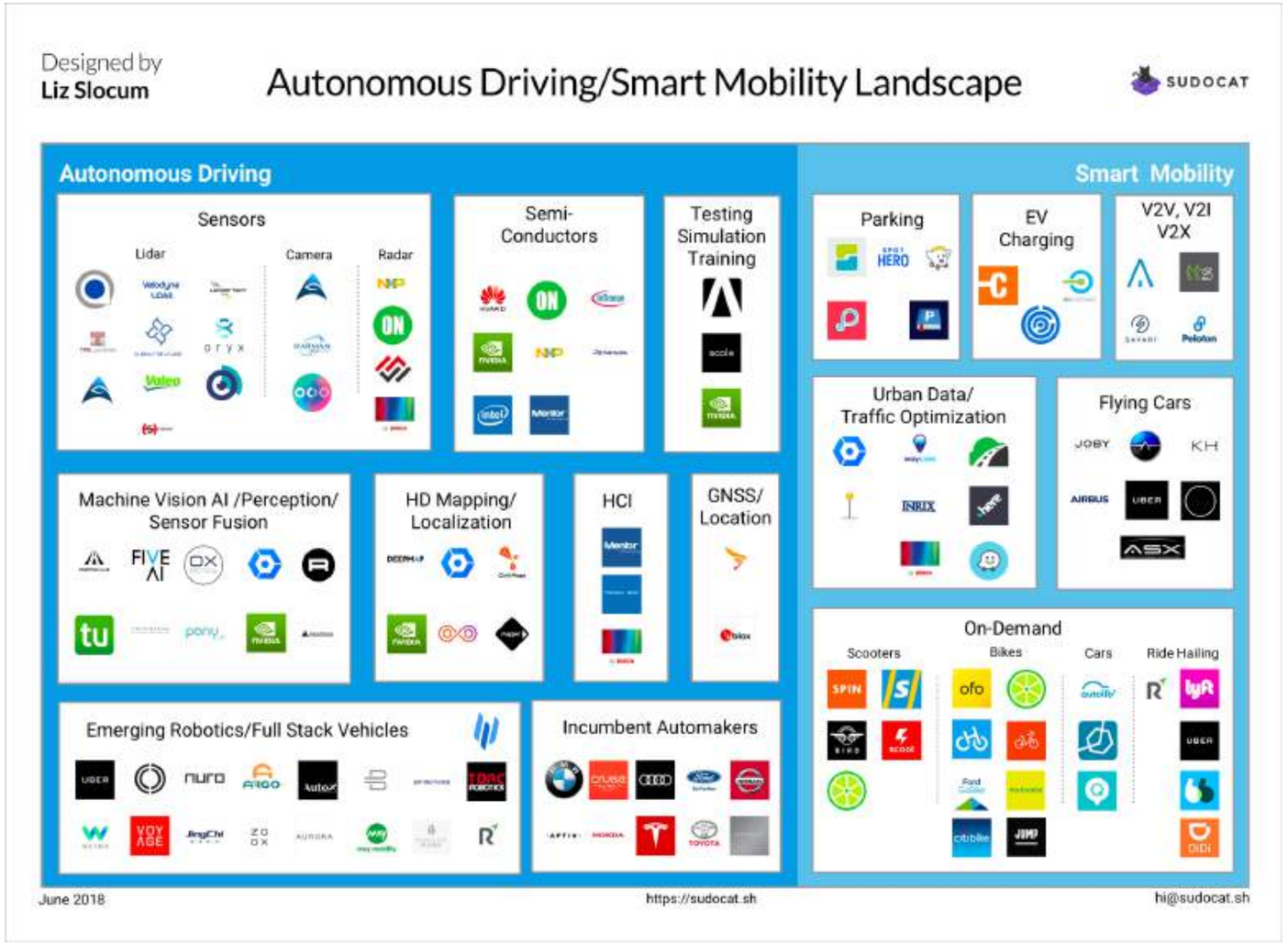
GETTING STARTED WITH MOBILITY



GETTING STARTED WITH MOBILITY



GETTING STARTED WITH MOBILITY



GETTING STARTED WITH MOBILITY



Programming with Android:
System Architecture

GETTING STARTED WITH MOBILITY

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

SMARTPHONES

TABLETS

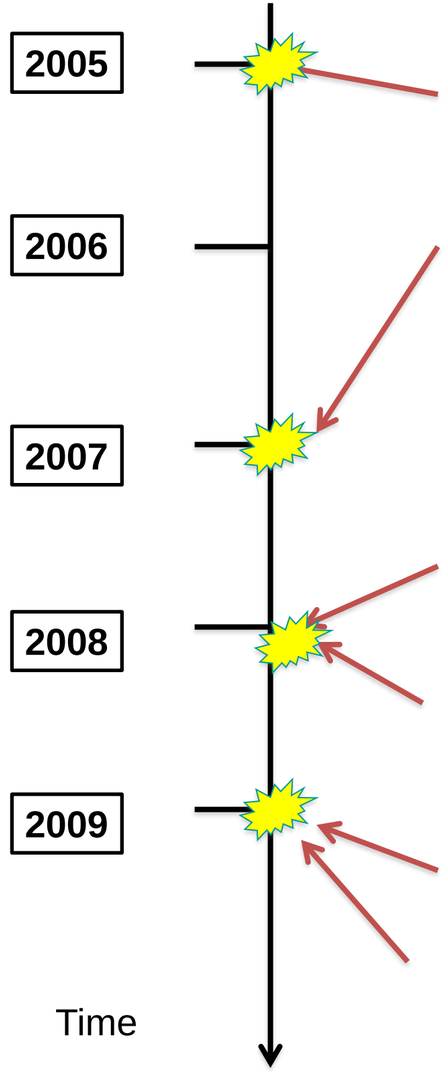
EREADERS

ANDROID TV

GOOGLE GLASSES

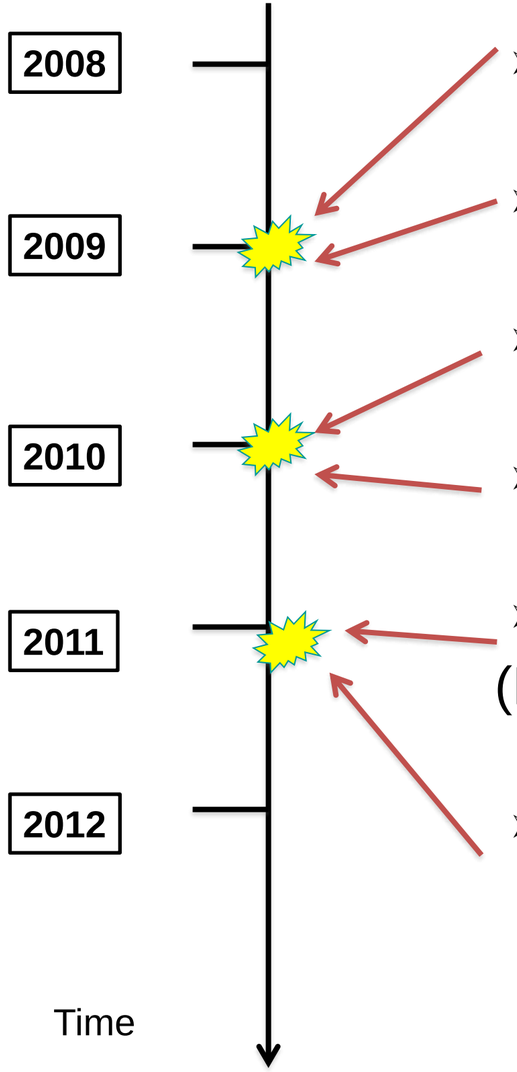
?

GETTING STARTED WITH MOBILITY



- **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

GETTING STARTED WITH MOBILITY



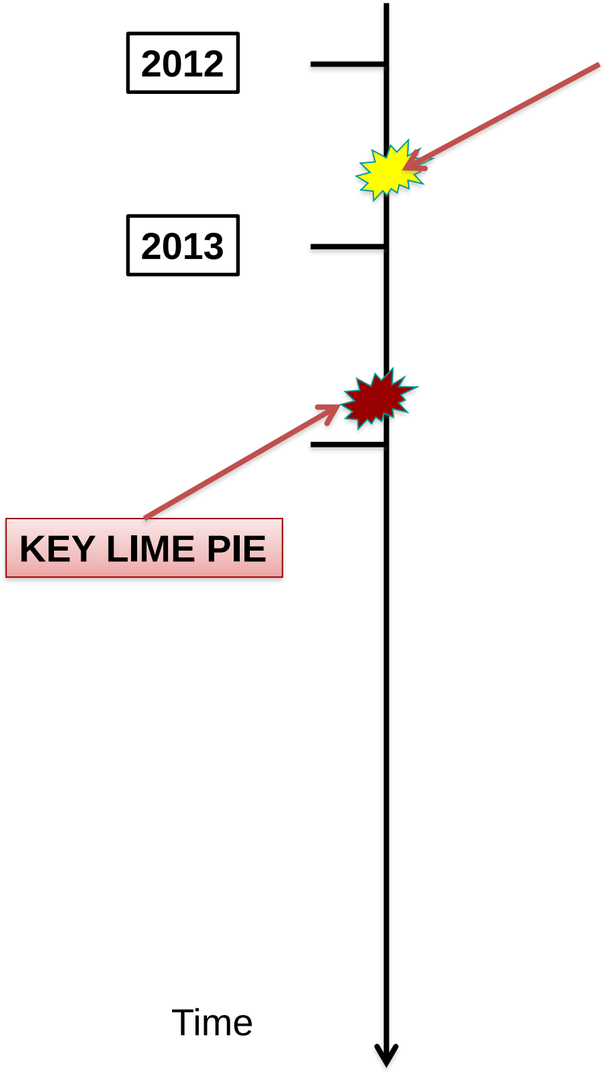
- Android **1.6** (**Donut**) Released
- Android **2.0** (**Eclair**) Released

- 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.)

GETTING STARTED WITH MOBILITY



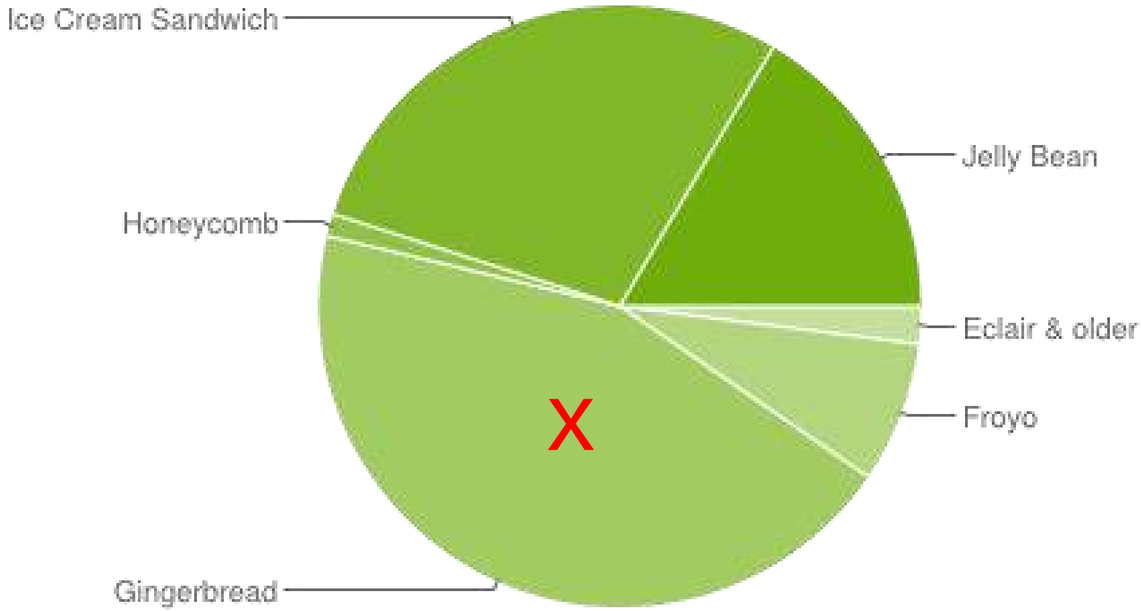
- Android **4.2** (**Jelly Bean**) Released
  - Gesture Mode for Accessibility
  - Improved browser performance
  - Easy data-sharing through NFC
  - Improved camera and face recognition functionalities
  - …

API Level 17 (Android 4.2):

- Daydream: screensaver customization API
- Support to multi-user environments
- Nested fragments for UI improvements
- …

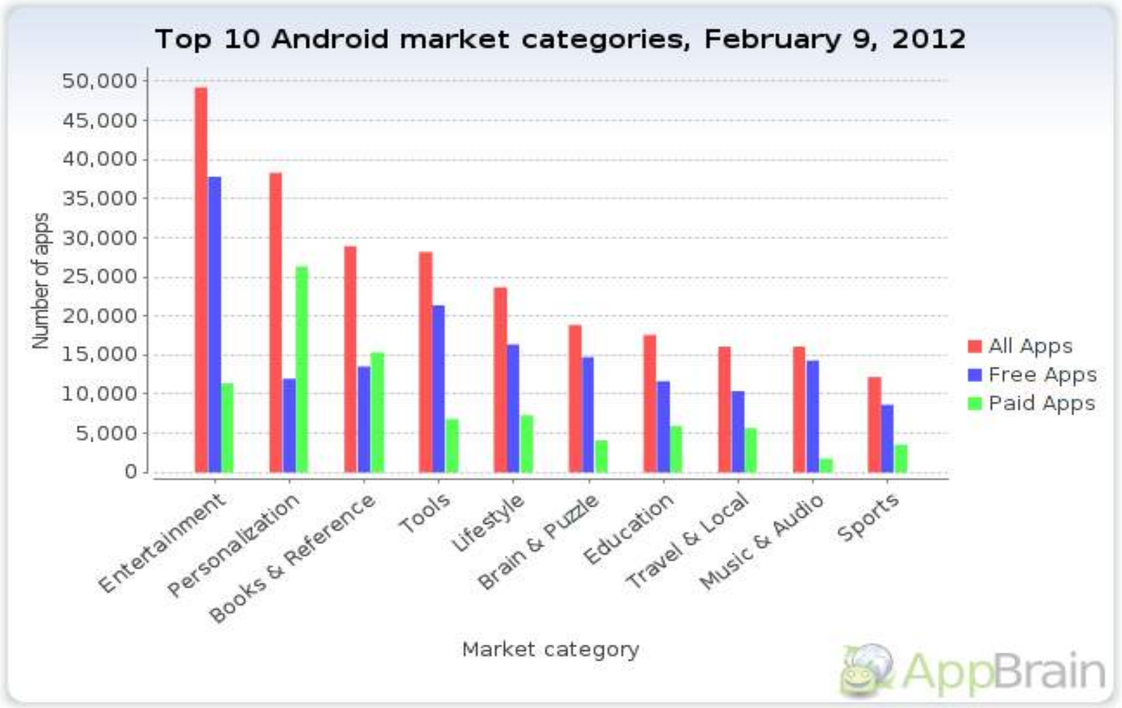
GETTING STARTED WITH MOBILITY

ANDROID DISTRIBUTIONS



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

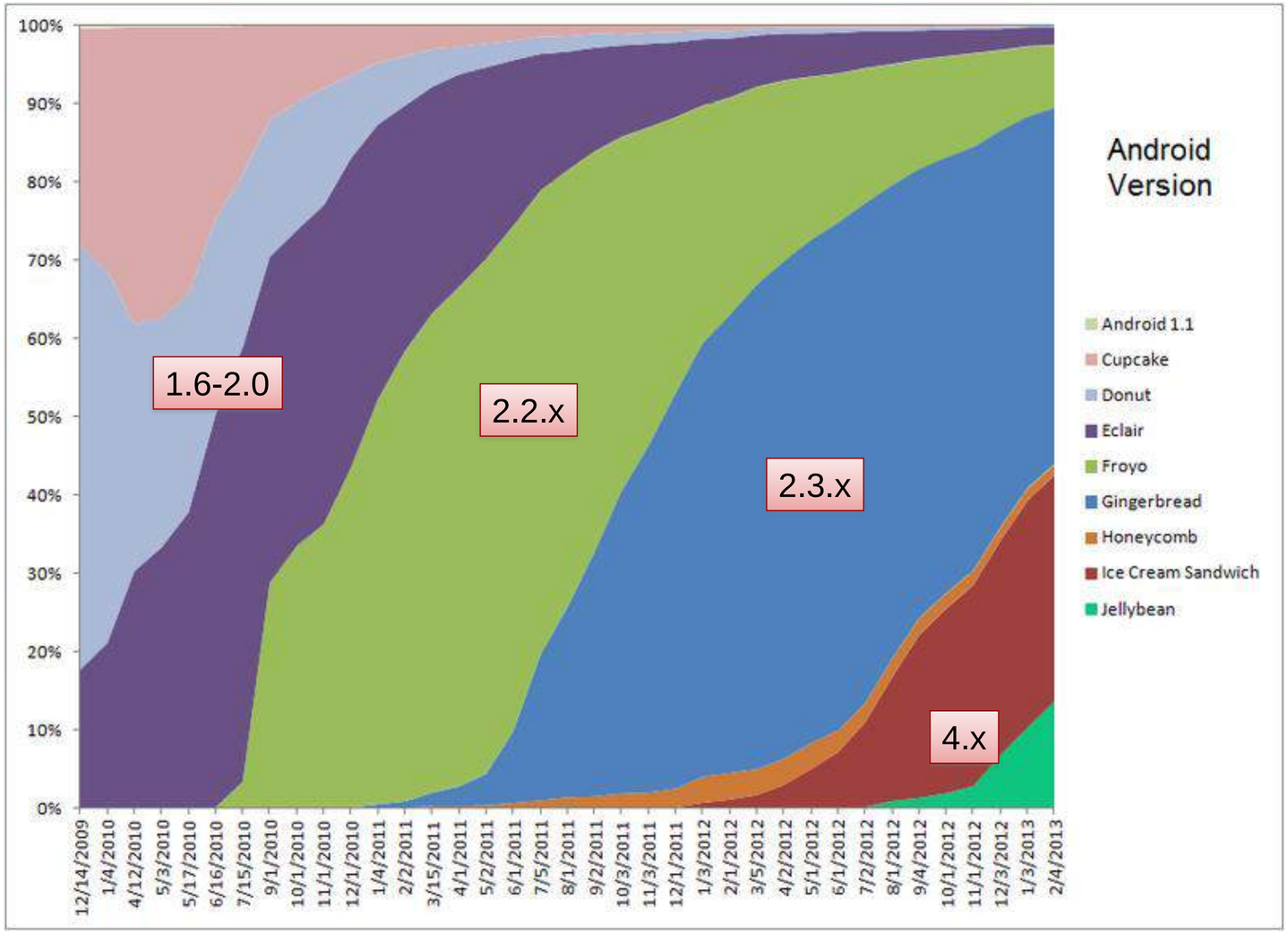
ANDROID APPLICATIONS



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

GETTING STARTED WITH MOBILITY

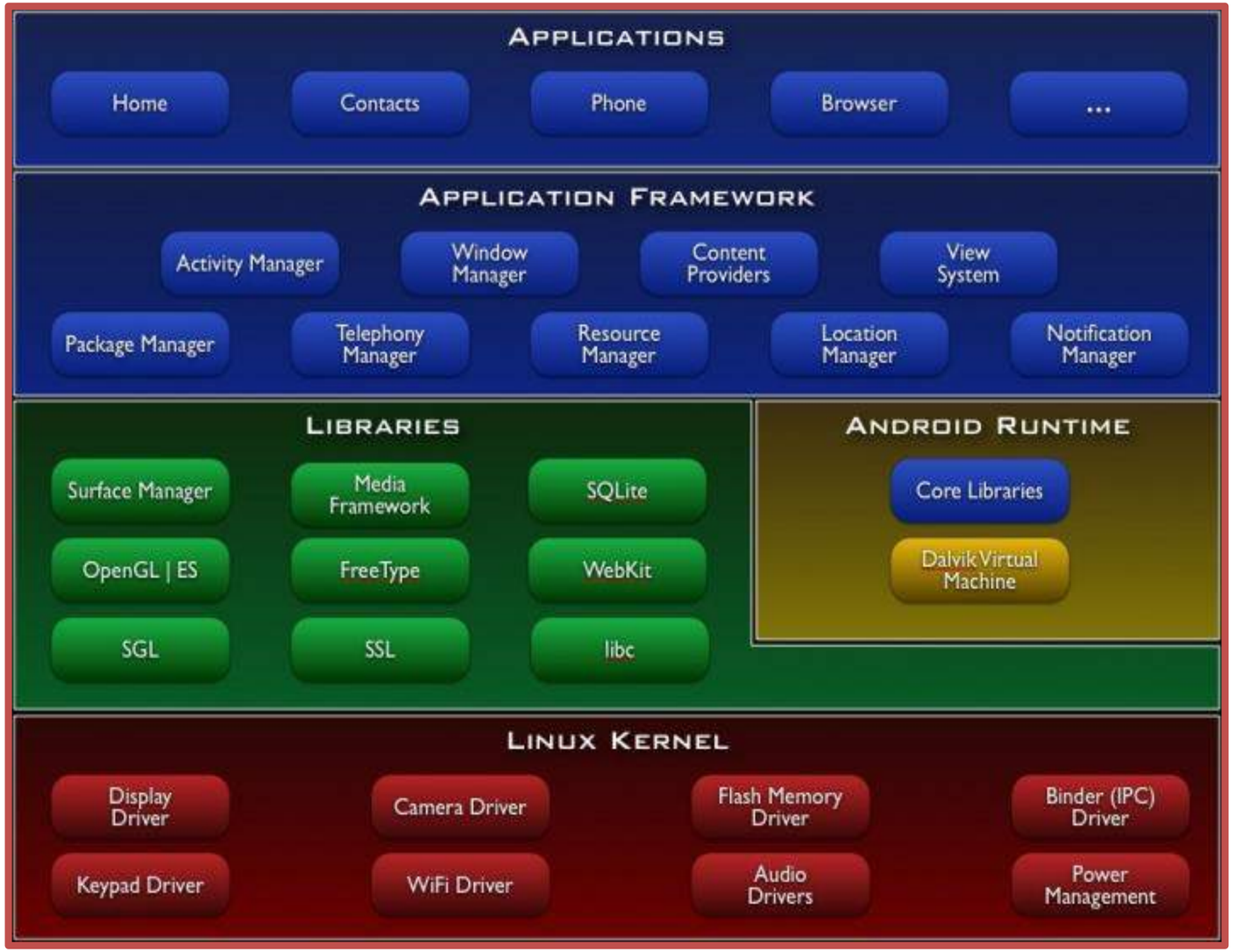
http://en.wikipedia.org/wiki/Android_version_history



ANDROID VERSION HISTORY AND POPULARITY

(2009-2013)

GETTING STARTED WITH MOBILITY

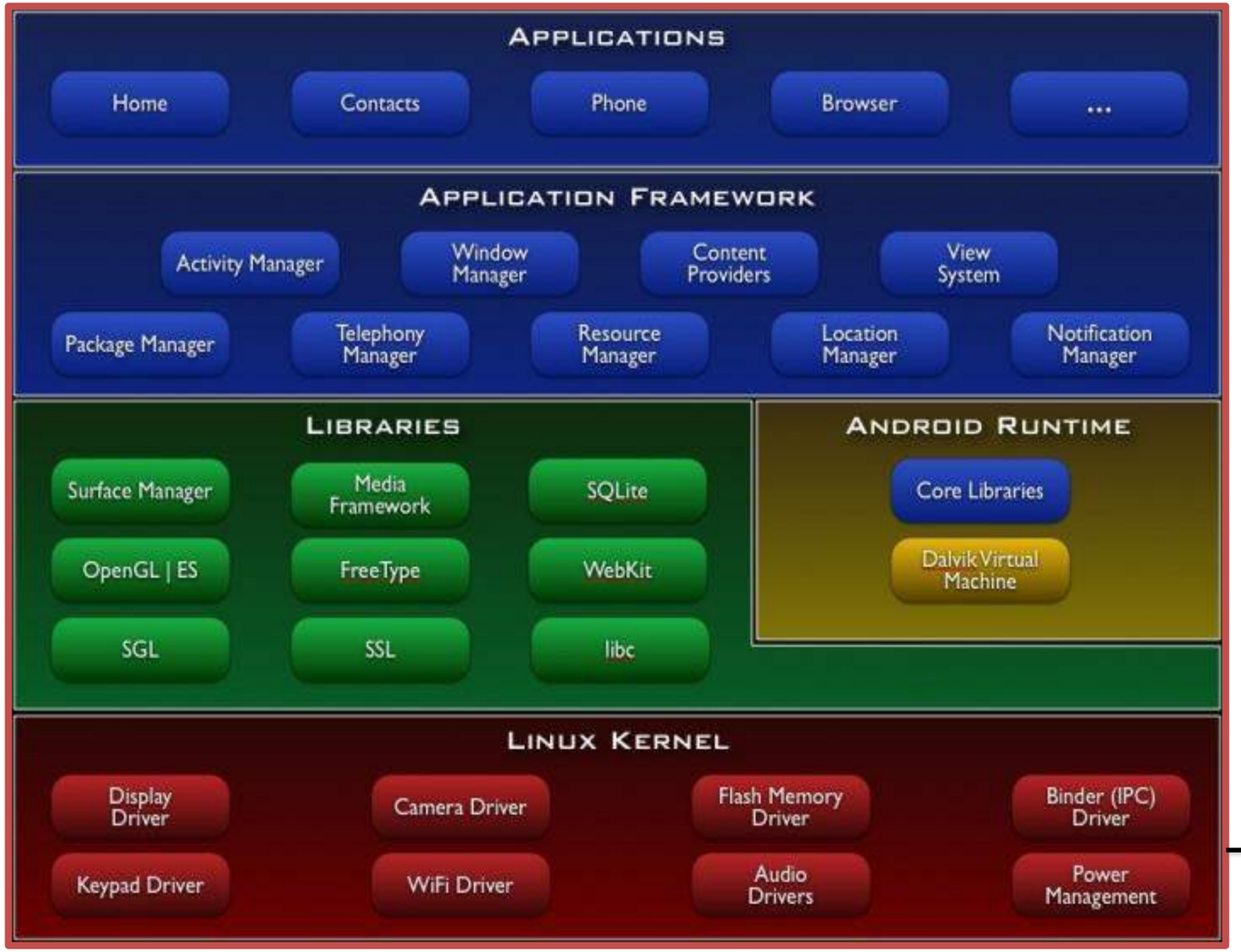


Stack
Architecture

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

Business-friendly License

GETTING STARTED WITH MOBILITY

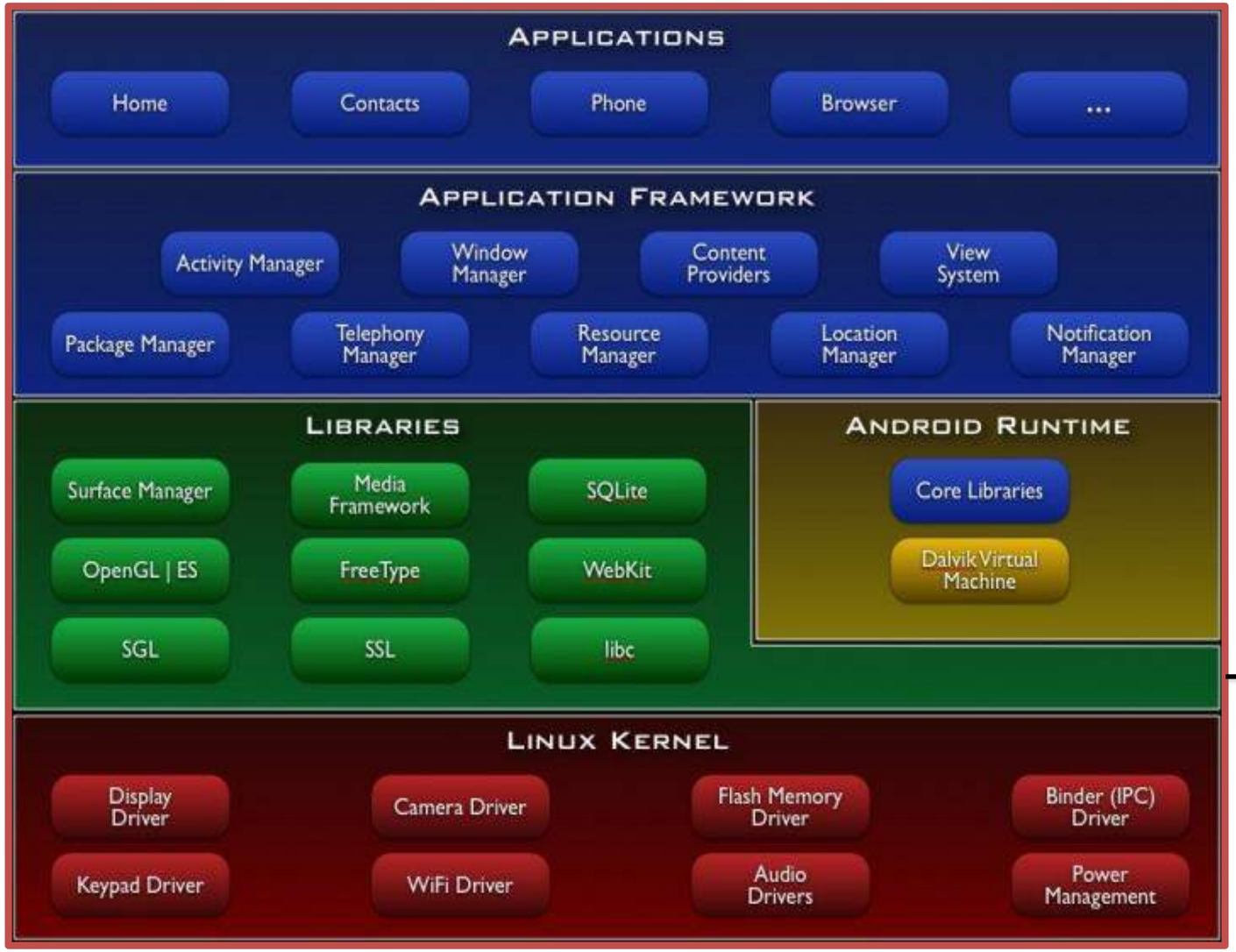


Built on top of **Linux kernel** (v. 2.6-3.0)

Advantages:

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

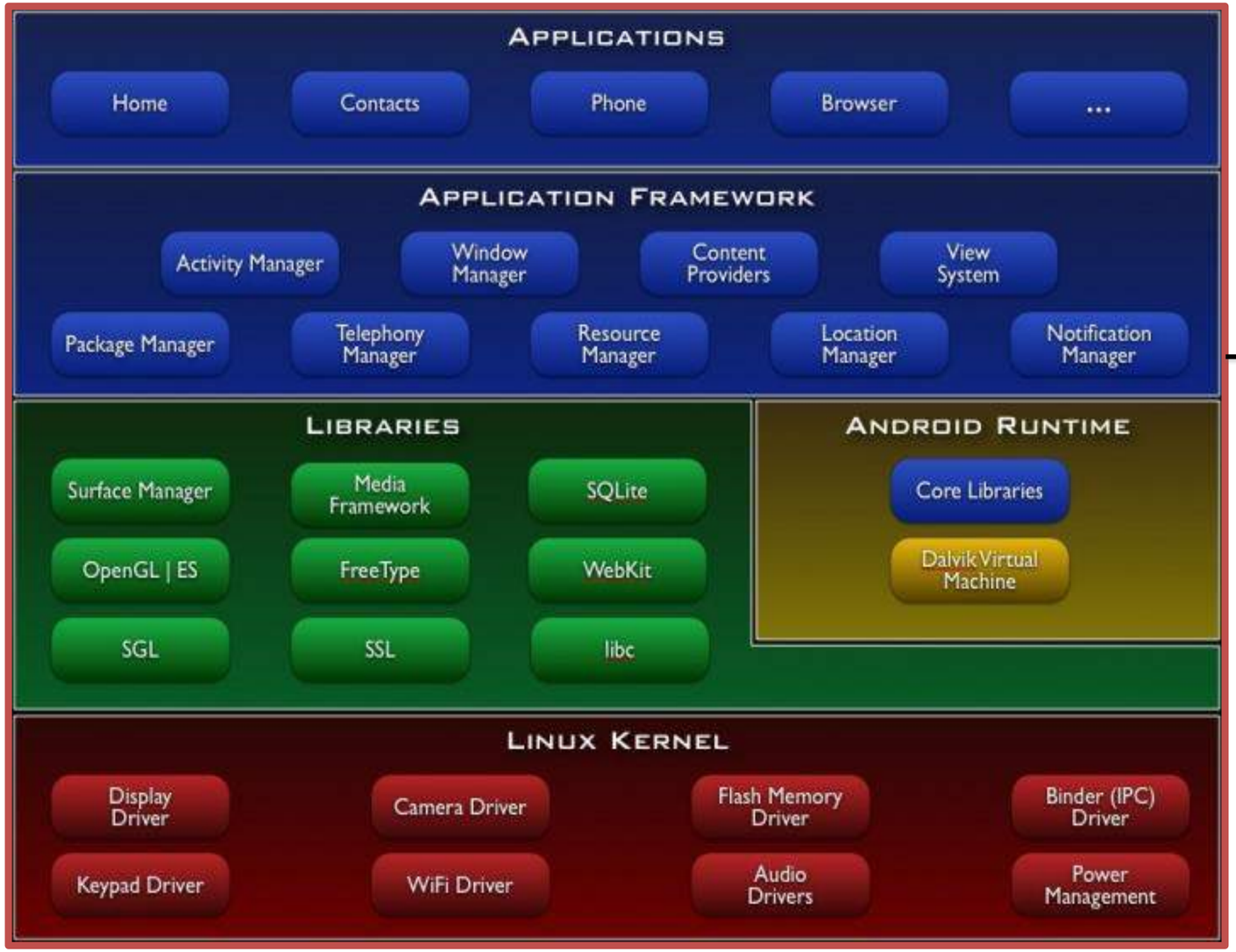
GETTING STARTED WITH MOBILITY



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

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

GETTING STARTED WITH MOBILITY

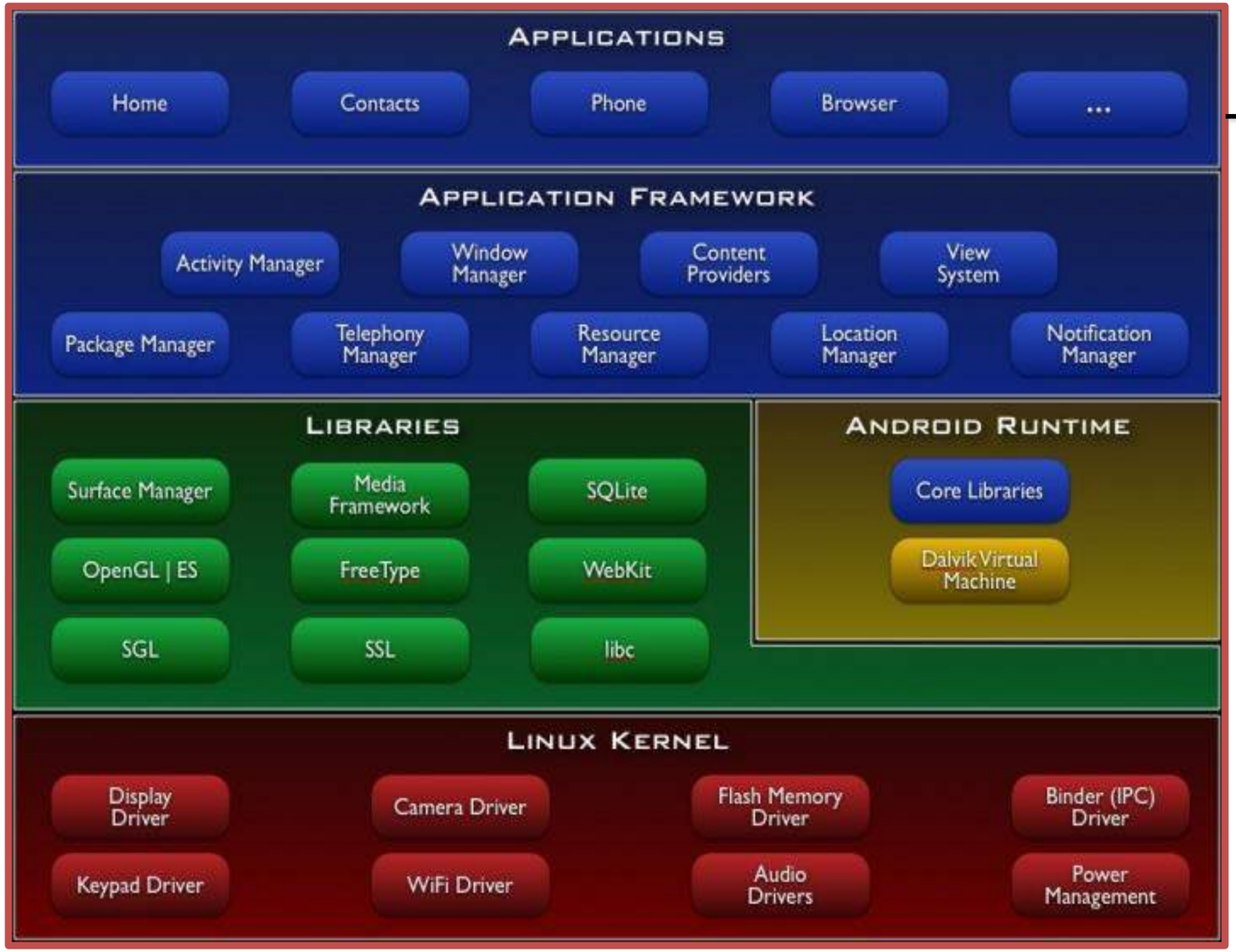


Application Libraries

(Core Components of Android)

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

GETTING STARTED WITH MOBILITY

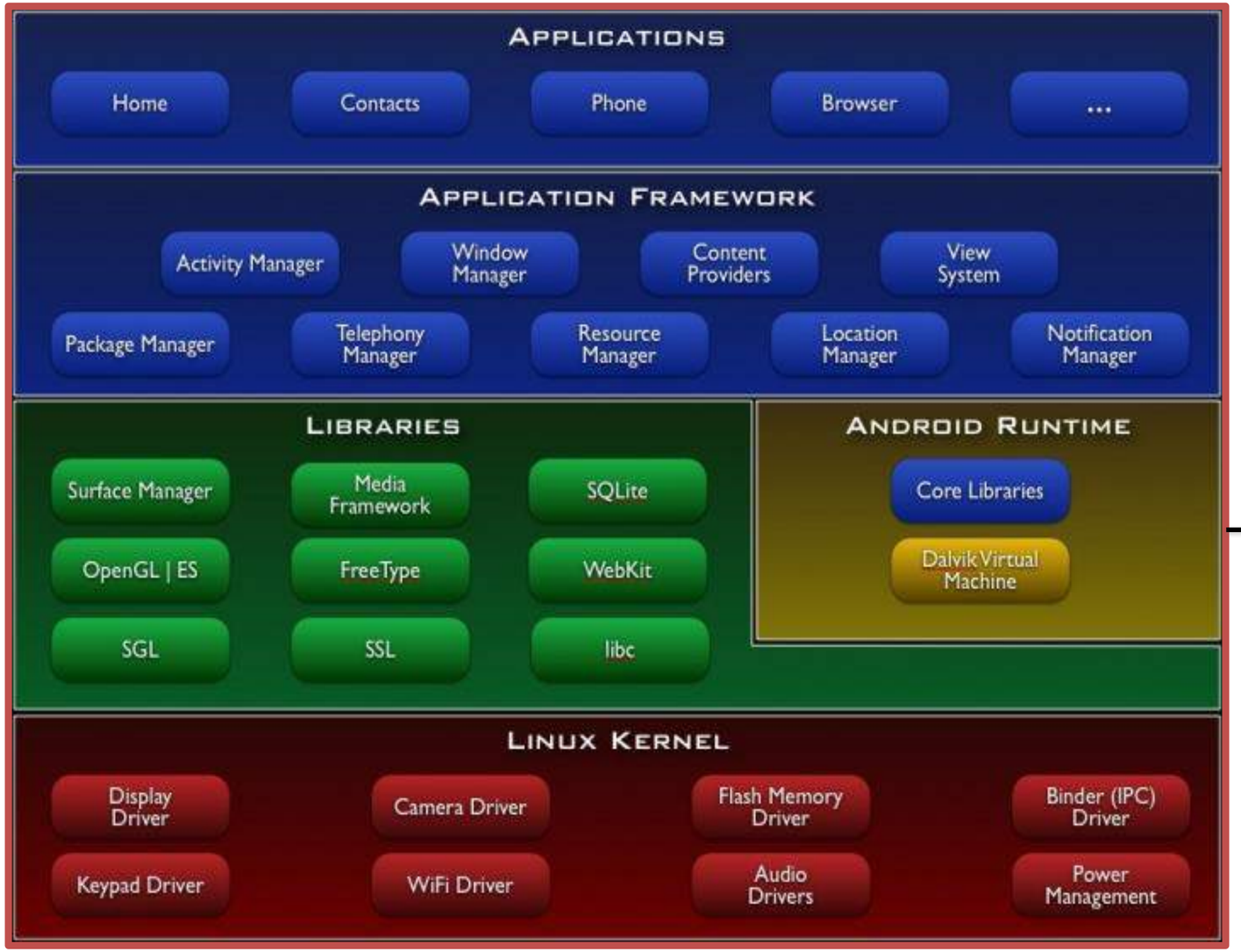


Applications

(Written in **Java** code)

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

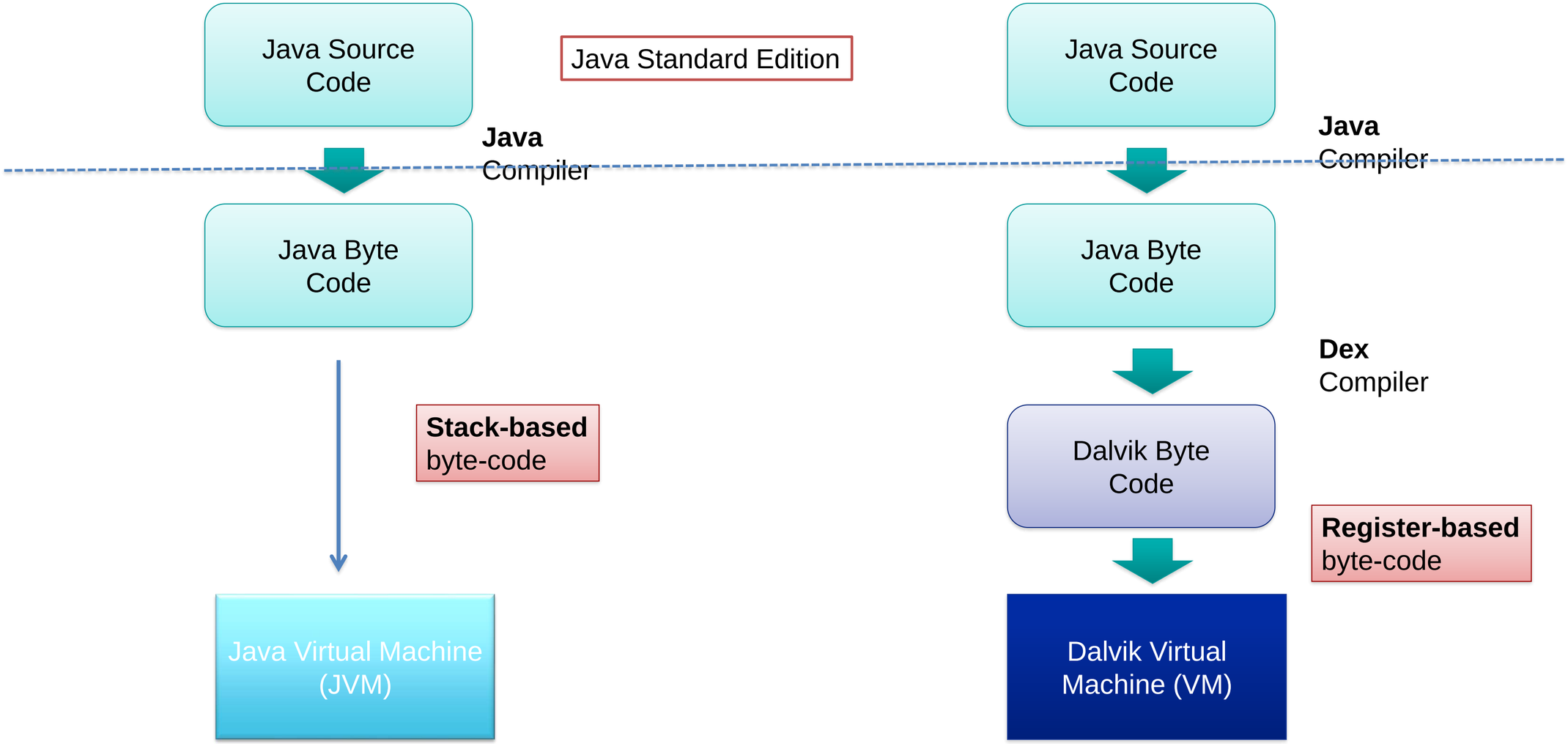
GETTING STARTED WITH MOBILITY



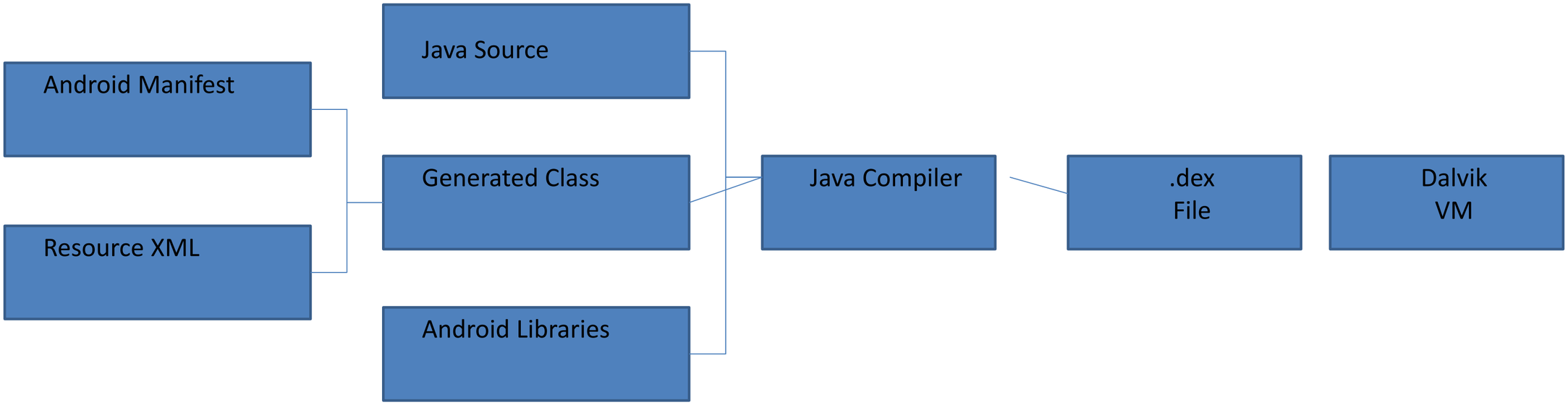
Dalvik Virtual Machine (VM)

- **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
- ….

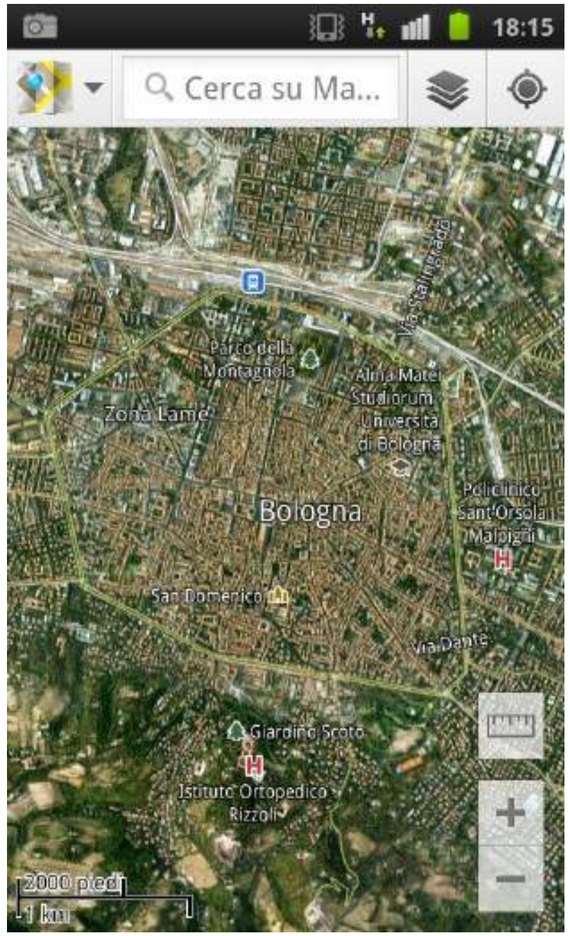
GETTING STARTED WITH MOBILITY



GETTING STARTED WITH MOBILITY



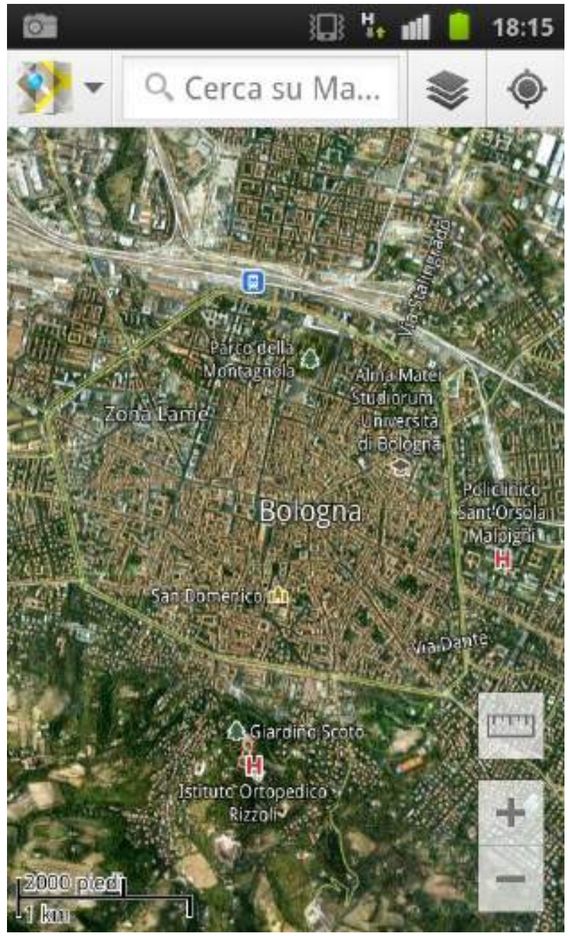
GETTING STARTED WITH MOBILITY



APPLICATION **DESIGN**:

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

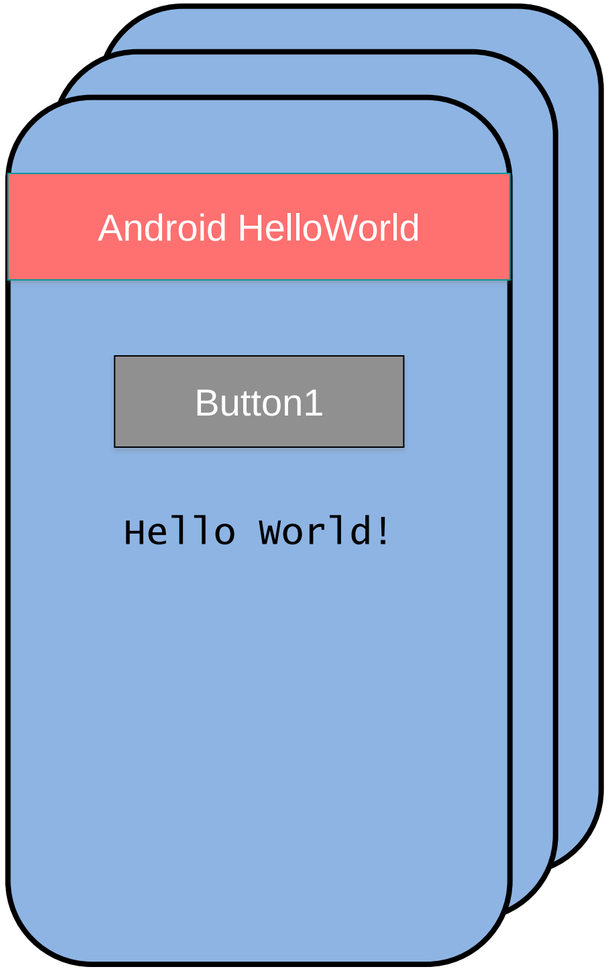
GETTING STARTED WITH MOBILITY



APPLICATION COMPONENTS

- **Activities**
- **Intents**
- **Services**
- **Content Providers**
- **Broadcast Receivers**

GETTING STARTED WITH MOBILITY



- 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.

GETTING STARTED WITH MOBILITY

- 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");
```

GETTING STARTED WITH MOBILITY

- 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:

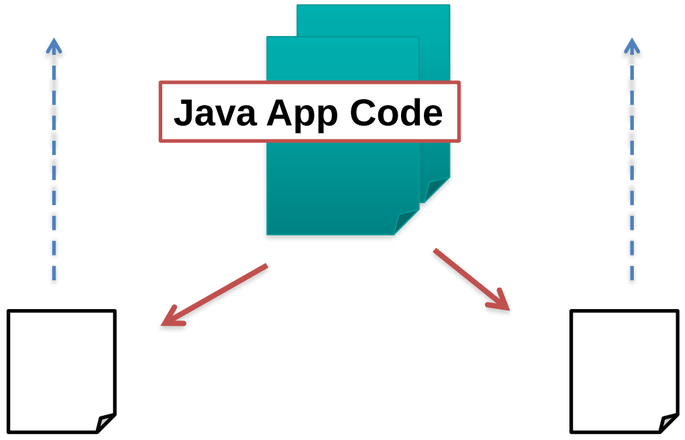
```
< TextView android.text=@string/hello" android:textcolor=@color/blue
android:layout_width="fill_parent" android:layout_height="wrap_content" />
< Button android.id="@+id/Button01" android:textcolor="@color/blue"
android:layout_width="fill_parent" android:layout_height="wrap_content" />
```

GETTING STARTED WITH MOBILITY

EXAMPLE

Device 1
HIGH screen pixel density

Device 2
LOW screen pixel density



XML Layout File
Device 1

XML Layout File
Device 2

- 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

GETTING STARTED WITH MOBILITY

- *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.

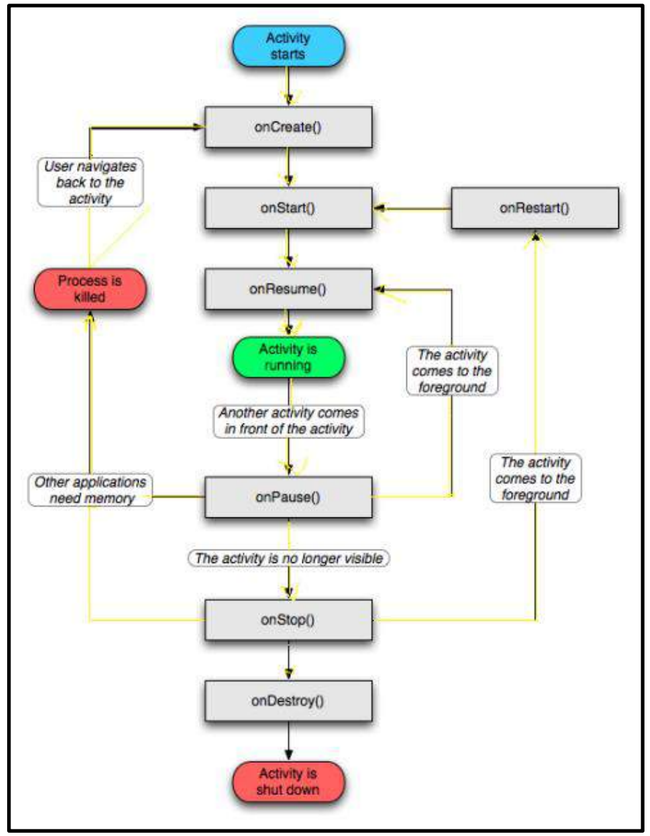
GETTING STARTED WITH MOBILITY

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

ESEMPIO

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

GETTING STARTED WITH MOBILITY



- 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 …)

GETTING STARTED WITH MOBILITY

- 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 …

GETTING STARTED WITH MOBILITY

```
public class MyApp extends Activity {

    public void onCreate() { ... }
    public void onPause()  { ... }
    public void onStop()   { ... }
    public void onDestroy(){ ... }
    ….
}
```

Called when the Activity is **created** the first time.

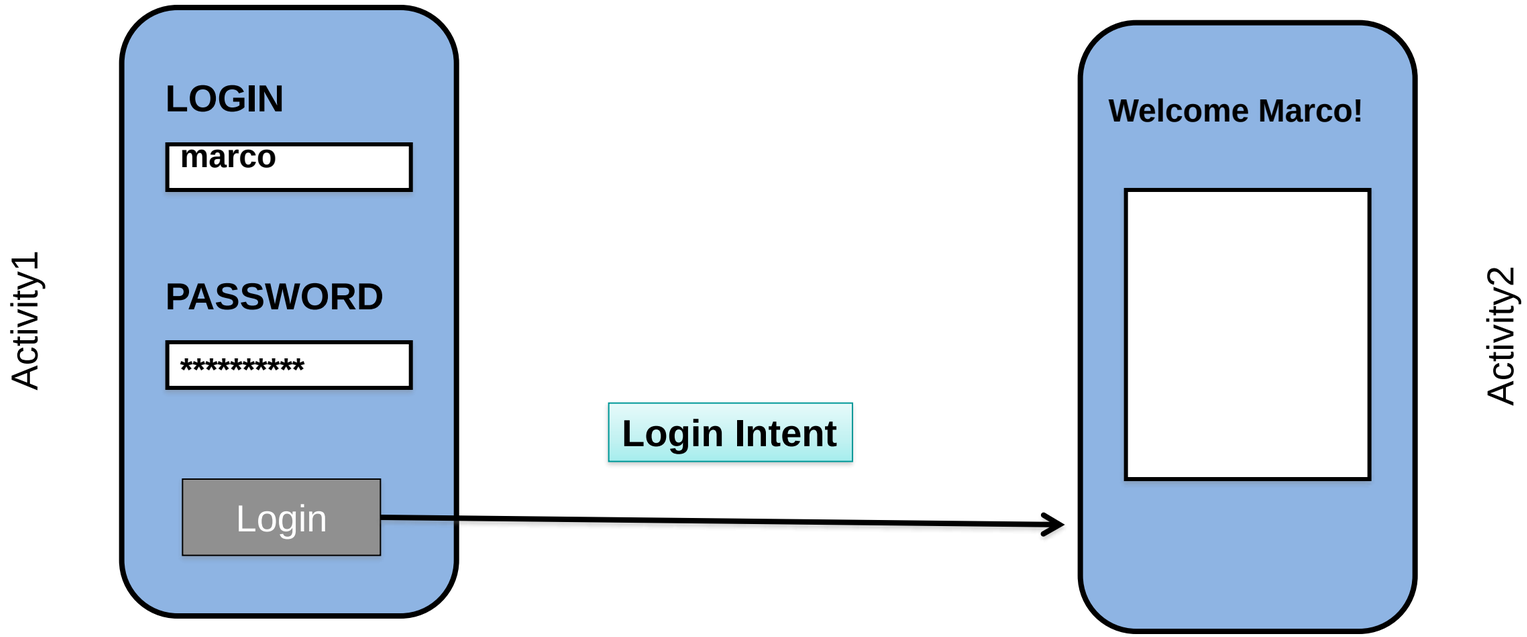
Called when the Activity is **partially visible**.

Called when the Activity is **no longer visible**.

Called when the Activity is **dismissed**.

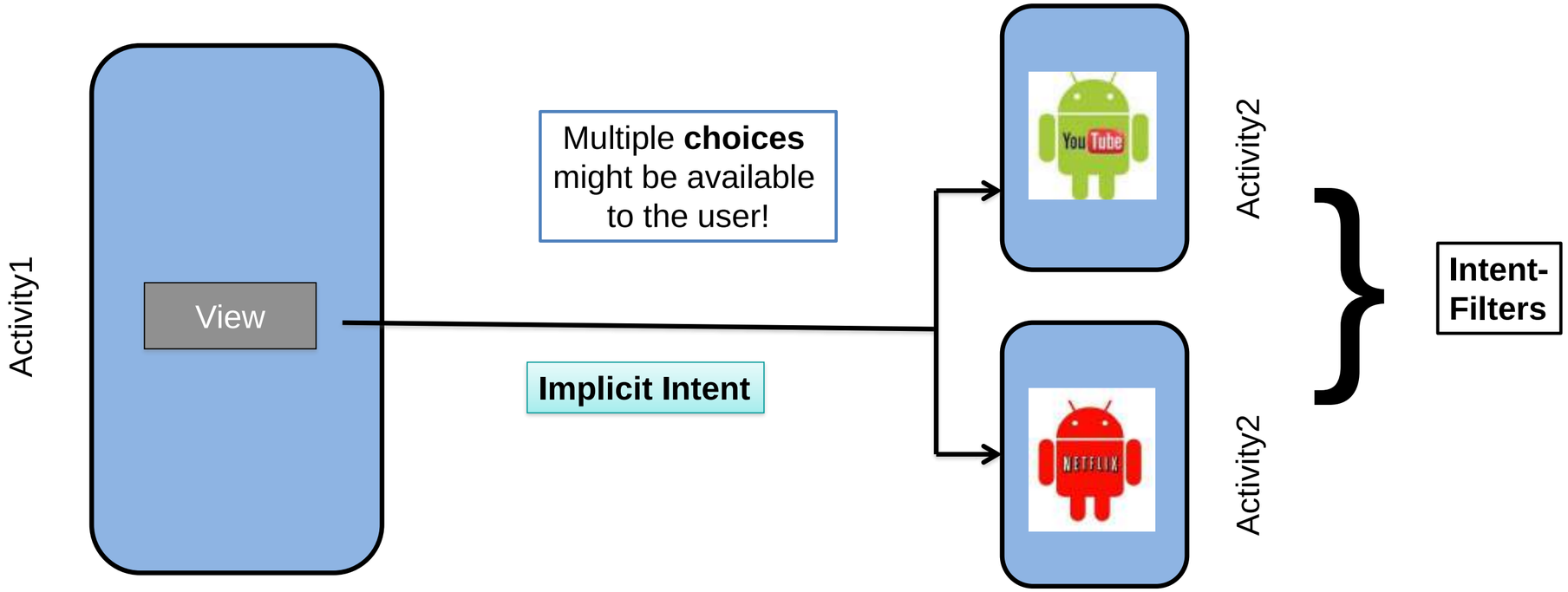
GETTING STARTED WITH MOBILITY

- **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)*.



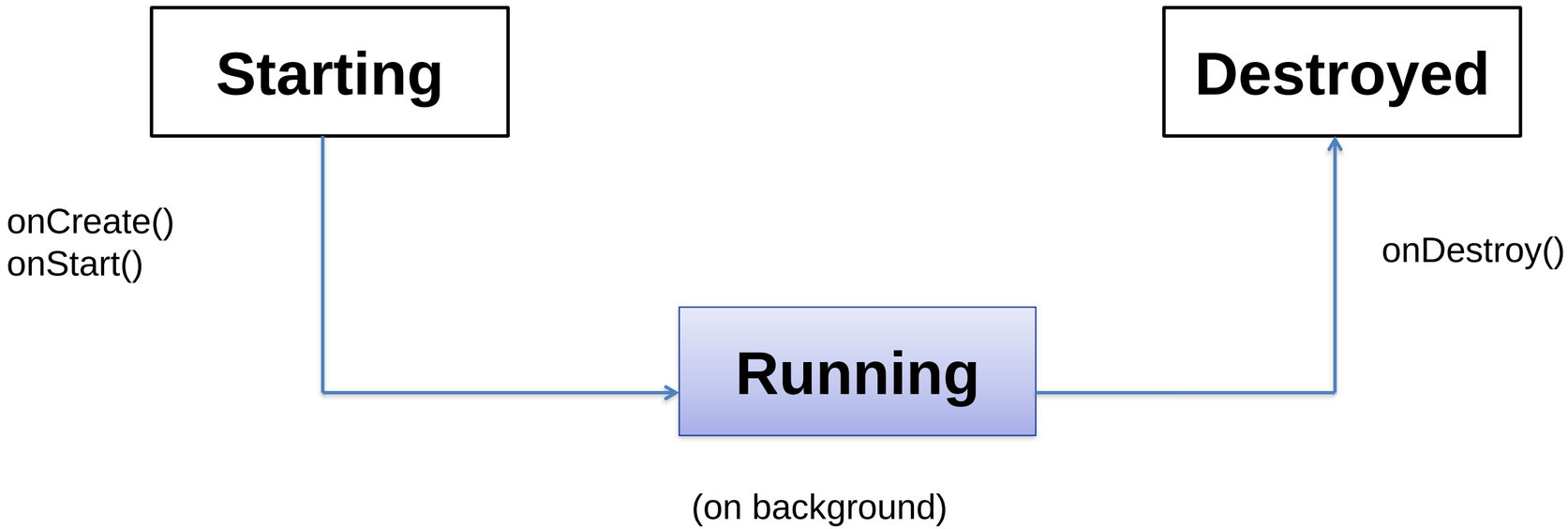
GETTING STARTED WITH MOBILITY

- **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")*.



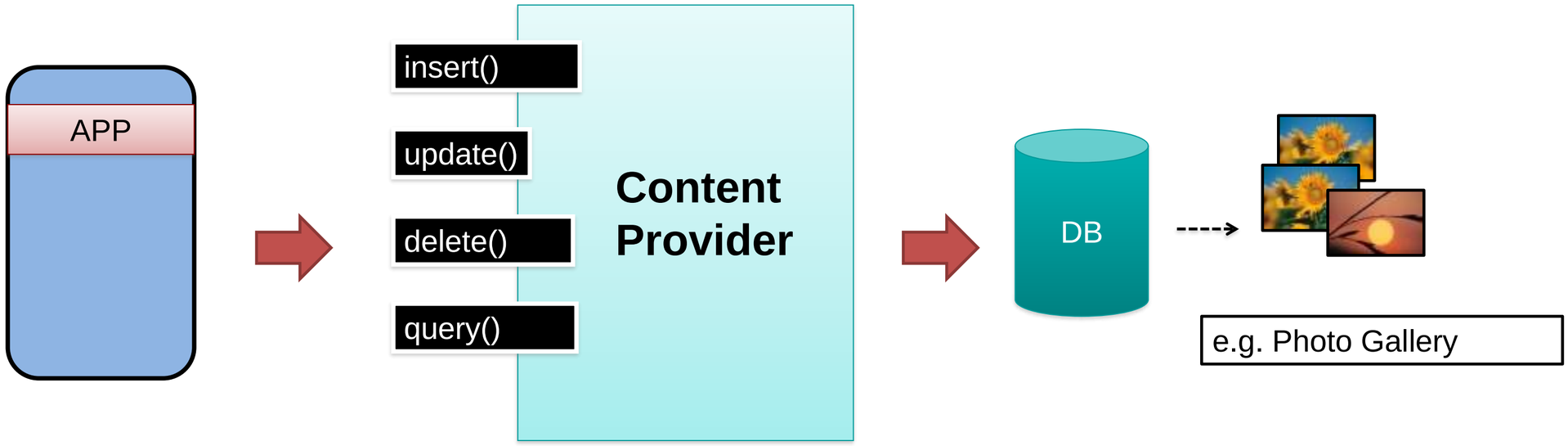
GETTING STARTED WITH MOBILITY

- **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:

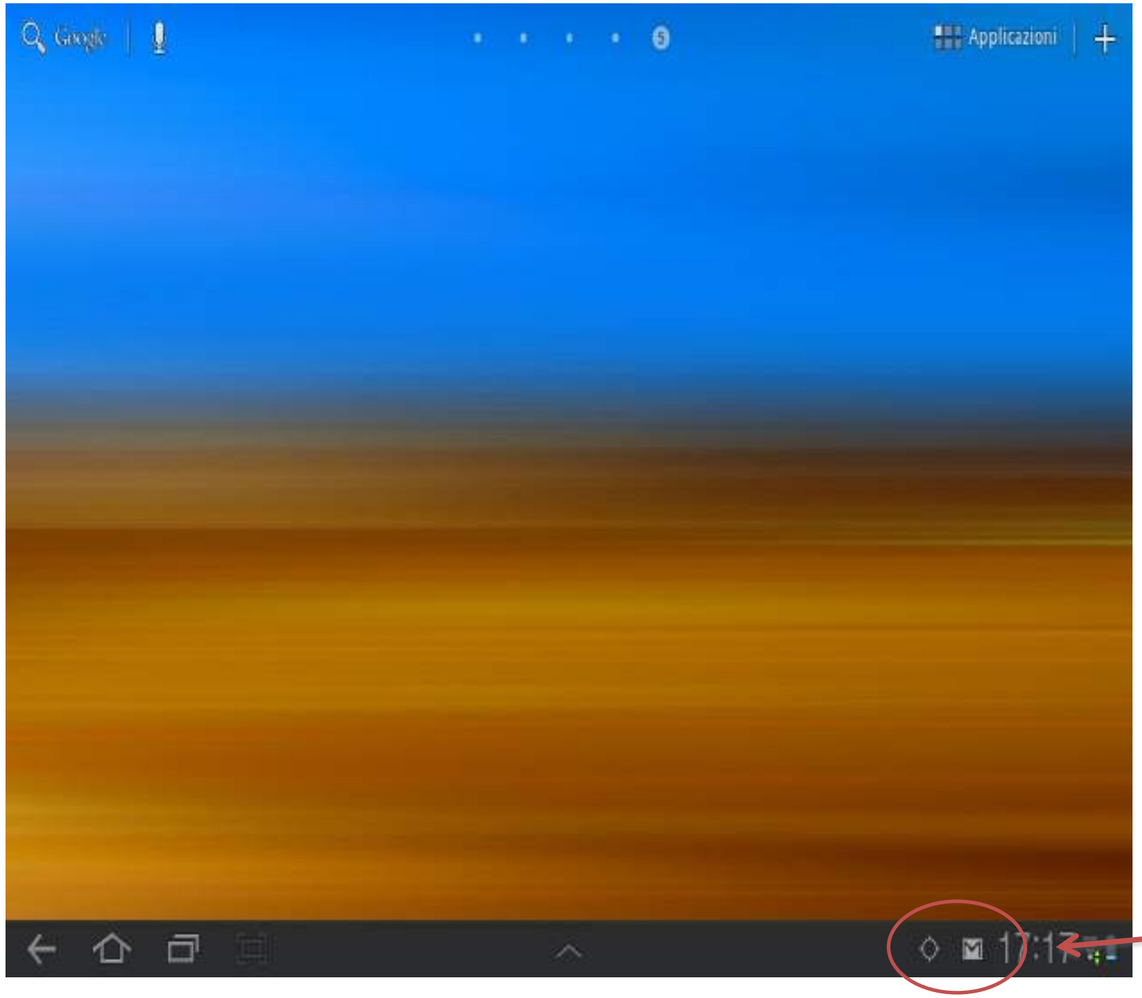


GETTING STARTED WITH MOBILITY

- 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*.



GETTING STARTED WITH MOBILITY



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

GETTING STARTED WITH MOBILITY

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);
        }
    }
```

GETTING STARTED WITH MOBILITY

BROADCAST RECEIVER example

```
public class WifiTester extends Activity {
        WifiManager mainWifi;
        WifiReceiver receiverWifi;
        List<ScanResult> wifiList;
          public void onCreate(Bundle savedInstanceState) {
              …
              mainWifi = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
              receiverWifi = new WifiReceiver();
              registerReceiver(receiverWifi, new
IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
              mainWifi.startScan();
}
```

GETTING STARTED WITH MOBILITY

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

SOME EXAMPLEs …

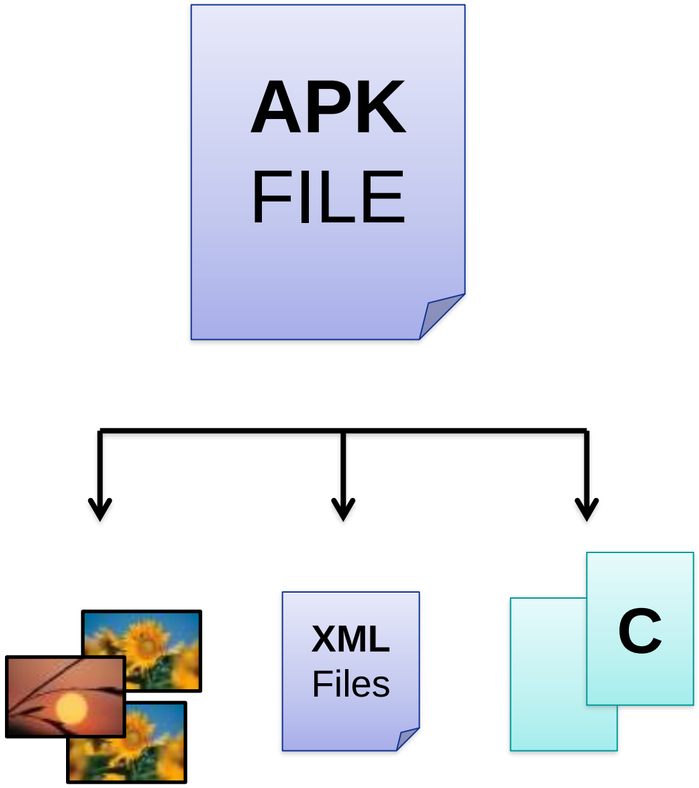
- *Telephony Manager* data access (call, SMS, etc)
- *Sensor* management (GPS, accelerometer, etc)
- Network *connectivity* (Wifi, bluetooth, NFC, etc)
- *Web* surfing (HTTP client, WebView, etc)
- *Storage* management (files, SQLite db, etc)
- ….

GETTING STARTED WITH MOBILITY

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

GETTING STARTED WITH MOBILITY 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)



GETTING STARTED WITH MOBILITY

- 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).

GETTING STARTED WITH MOBILITY

- 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" />
```