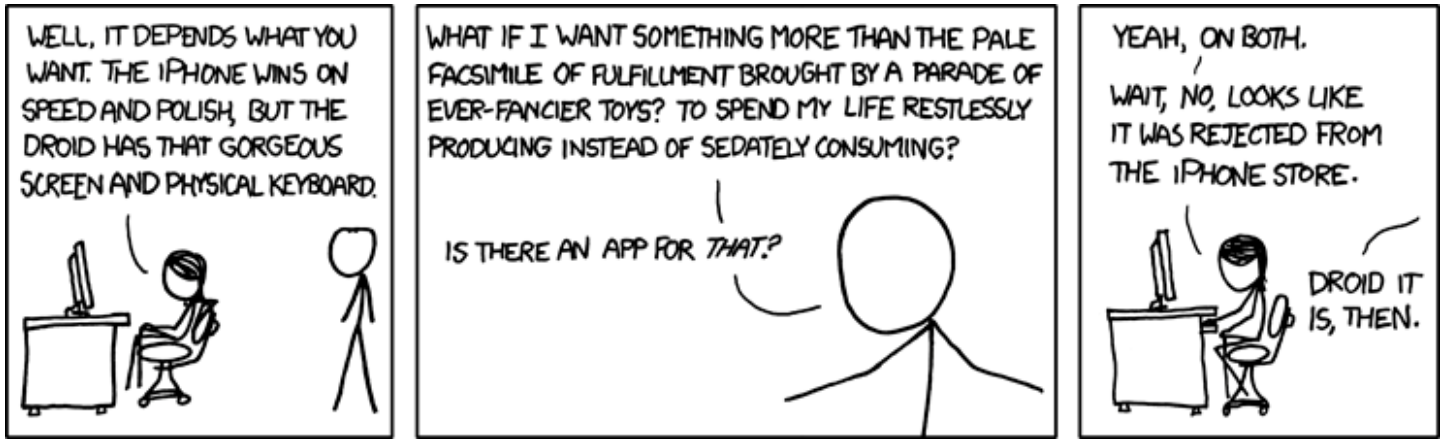
ANDROID

What would it take to build a better mobile phone?

Johann Bourcier - Université de Rennes 1



Context

- Android aims at providing a complete software stack for mobile and embedded devices (smartphone, tablet, cars, smartwatch …)

History

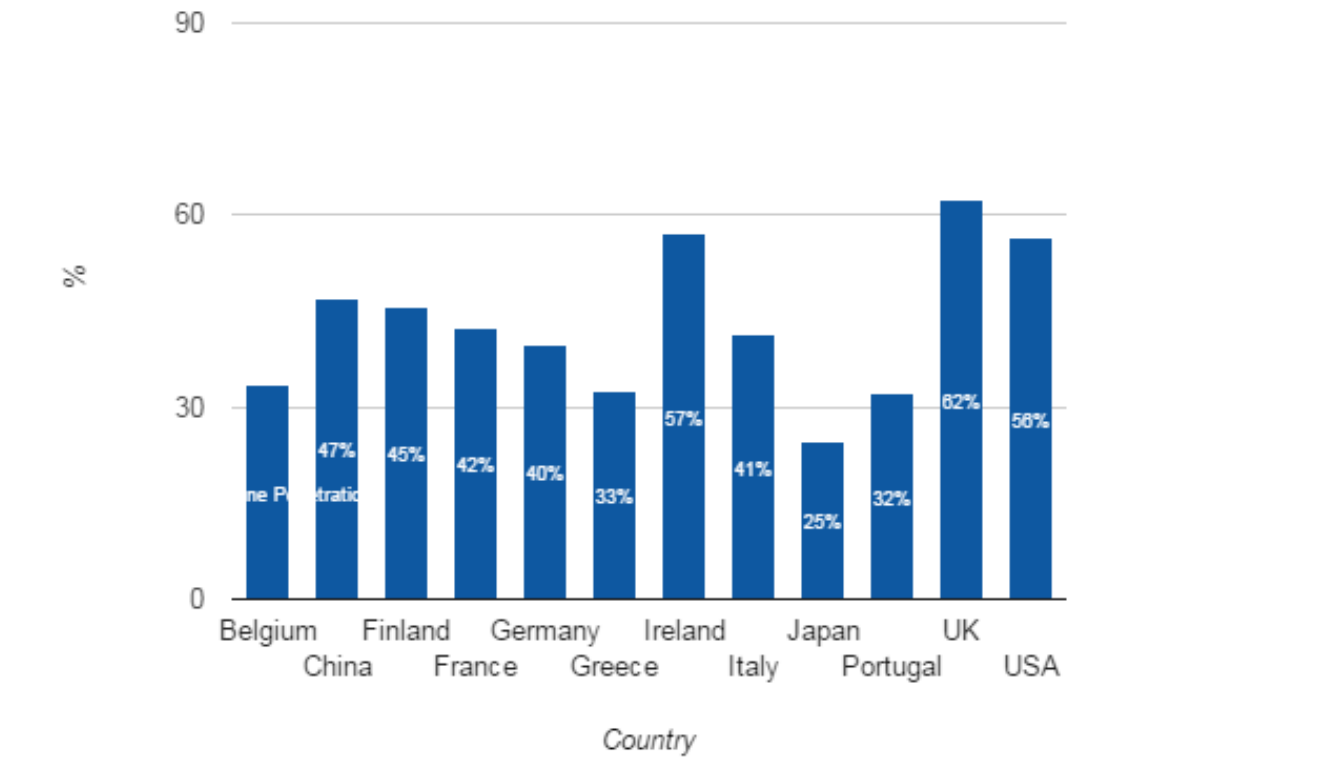
- Android comes from Androide which stand for a robot built to look like a human.

- Software platform from Google
- July 2005, Google acquired Android, Inc.
- November 2007, Open Handset Alliance formed to develop open standards for mobile devices
- October 2008, Android available as open source
- Various releases are regularly provided

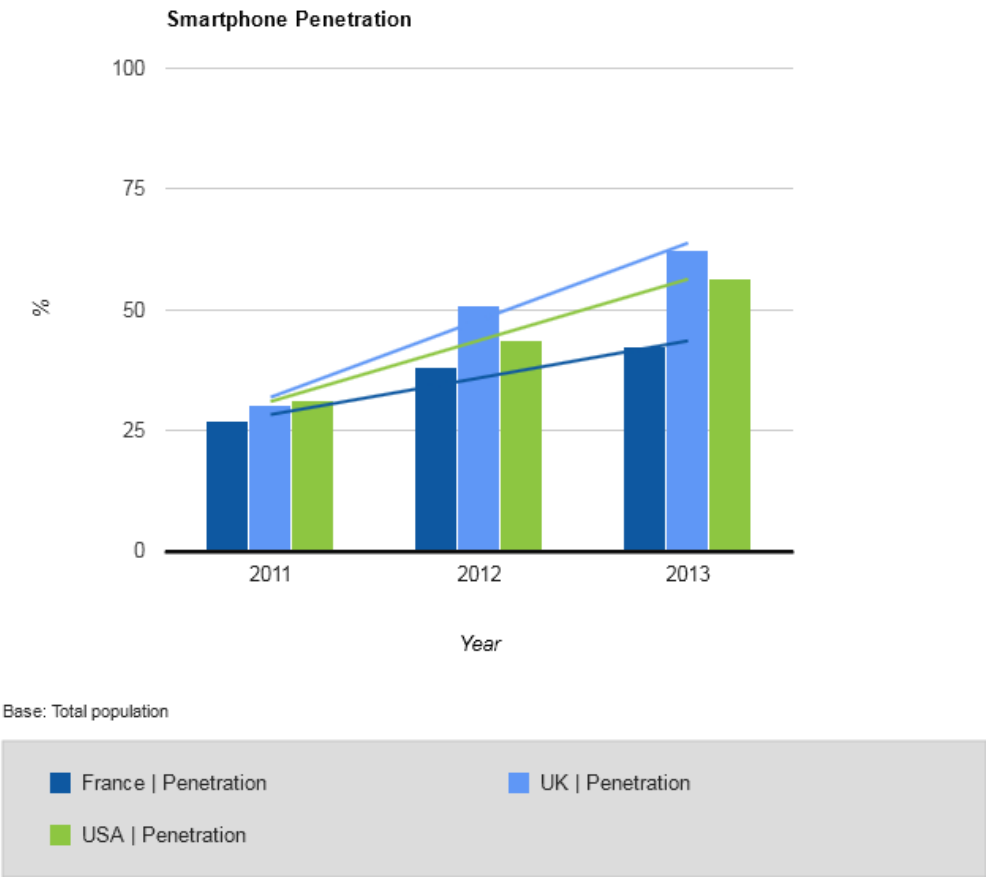
Android smartphone

- **Current number of app available 1,373,337**

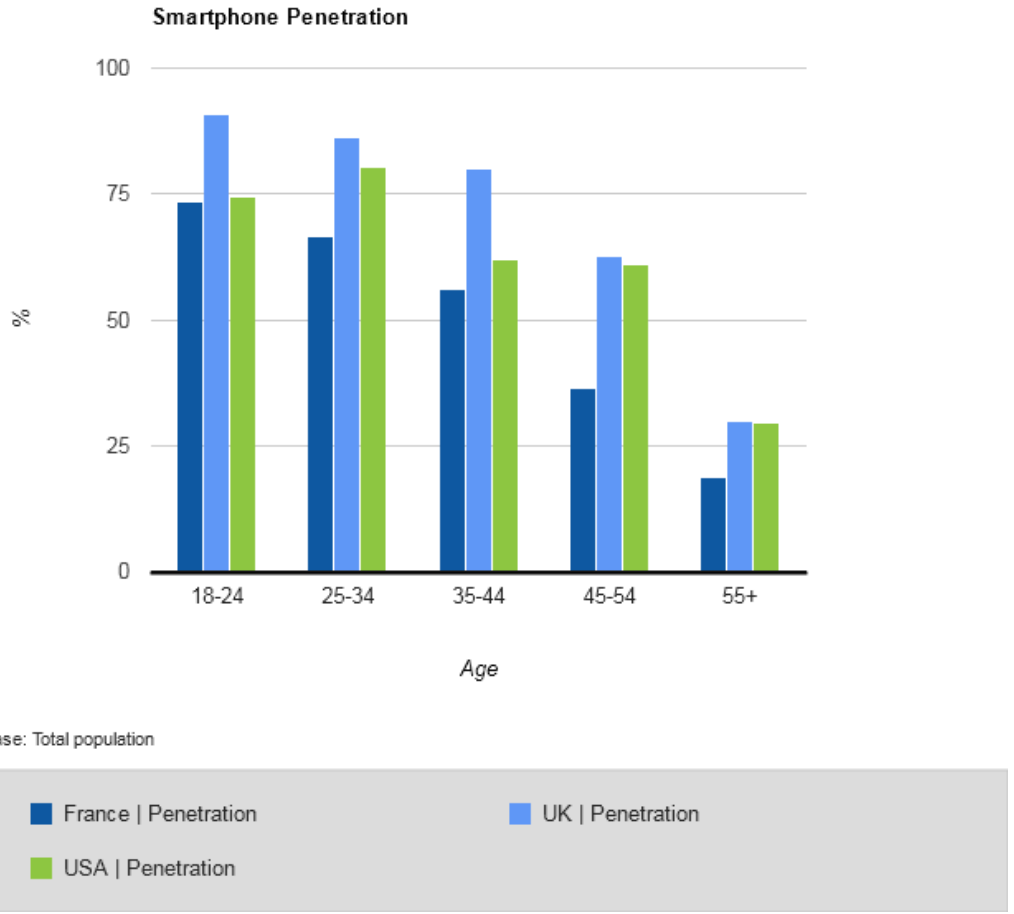
Smartphone penetration



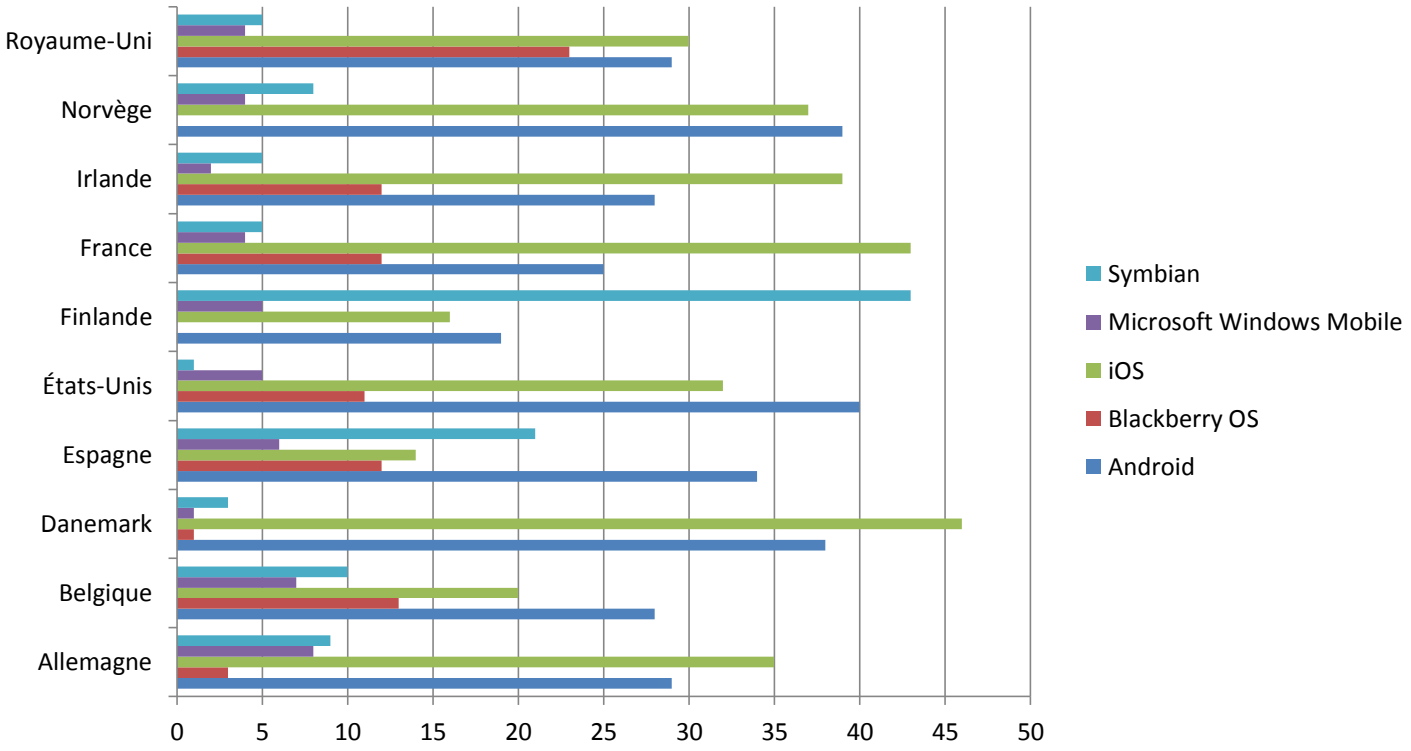
Smartphone user



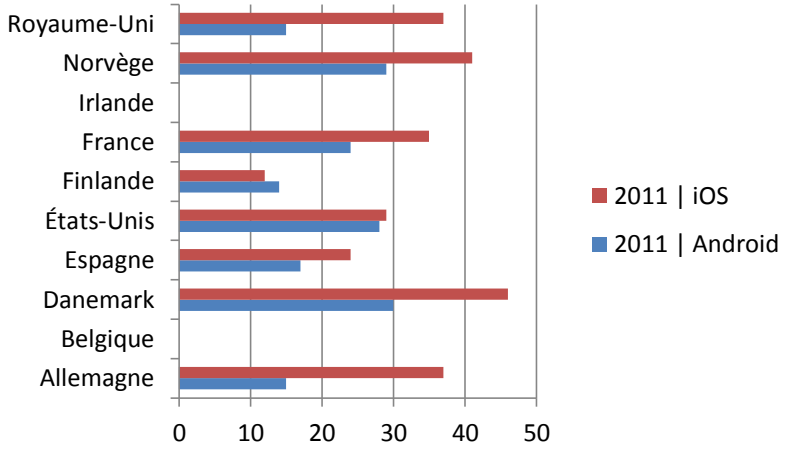
Usage by age

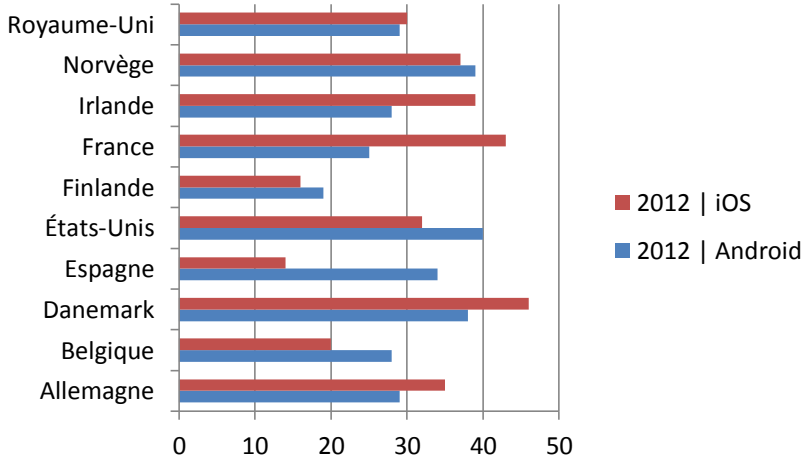

Smartphone Penetration
100
75
50
25
0
%
18-24
25-34
35-44
45-54
55+
Age
Base: Total population
France | Penetration
UK | Penetration
USA | Penetration

SmartPhone market in 2012



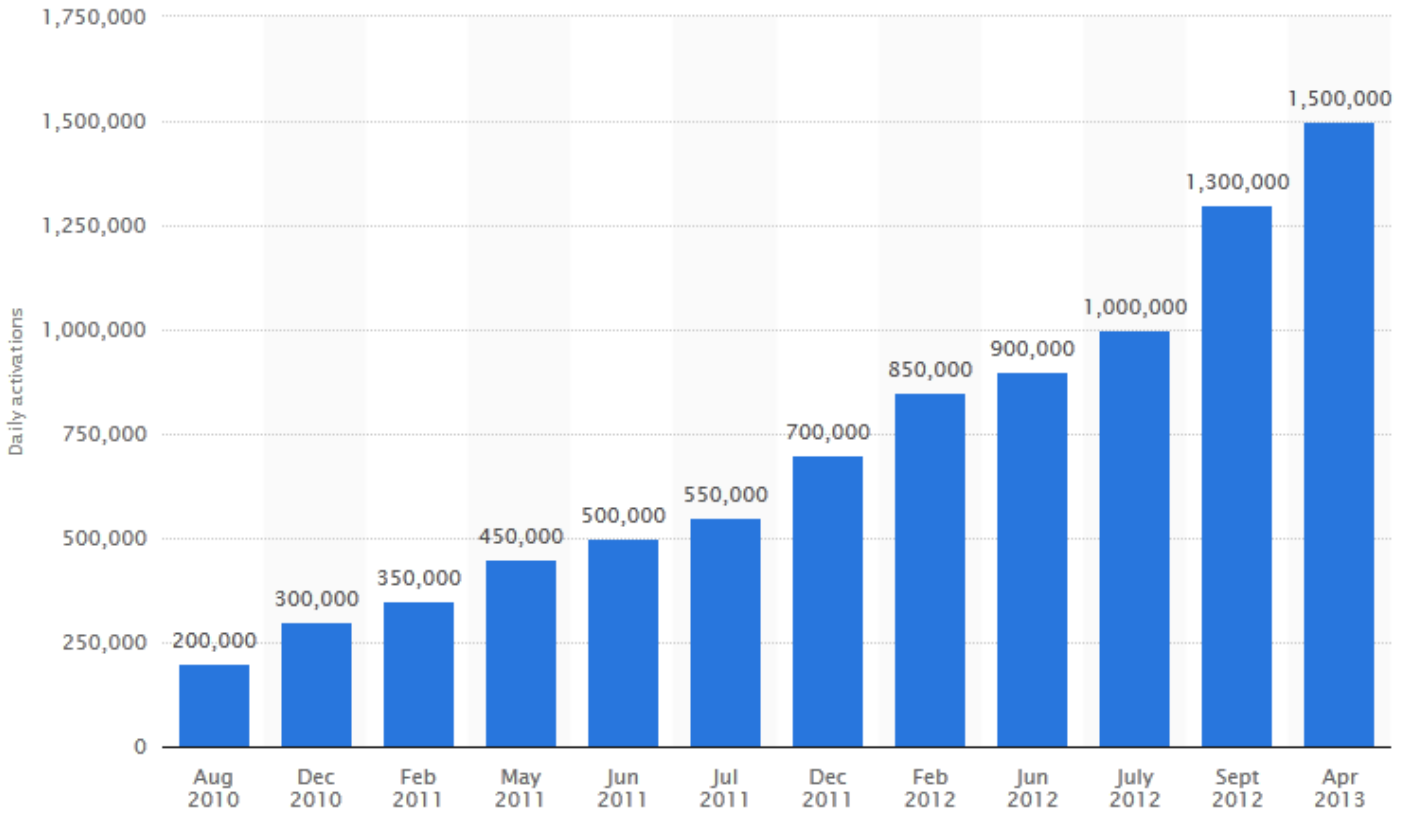
Evolution of the market



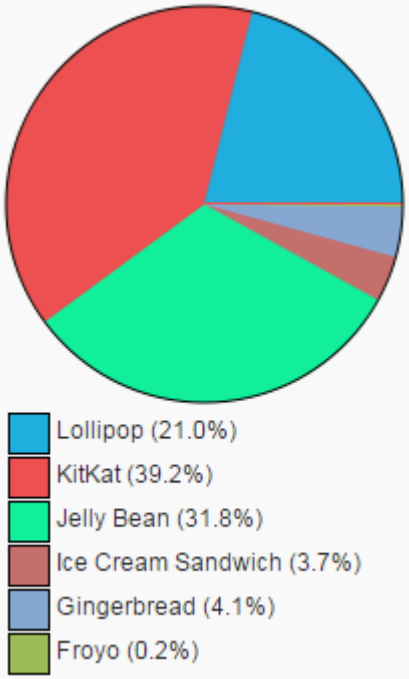


Android Smartphones	Values
Android's global market share	78.4%
Number of daily activations of Android devices	1,500,000
Global shipments of Android smartphones	1,133m
Distribution of Android Jelly Bean 4.1 x	29%
Number of Android smartphone users in the U.S.	76m

Number of daily activations of Android devices from August 2010 to April 2013



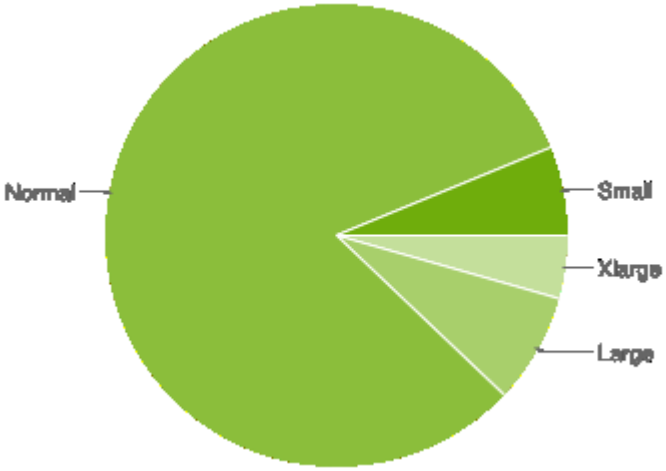
Android versions

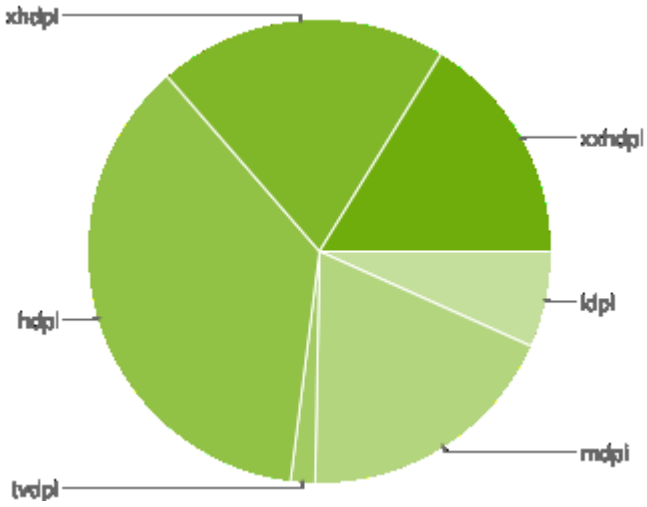


Version	Code name	Release date	API level	Distribution
5.1.x	Lollipop	March 9, 2015	22	5.1%
5.0–5.0.2		November 3, 2014	21	15.9%
4.4–4.4.4	KitKat	October 31, 2013	19	39.2%
4.3.x	Jelly Bean	July 24, 2013	18	4.5%
4.2.x		November 13, 2012	17	15.2%
4.1.x		July 9, 2012	16	12.1%
4.0.3–4.0.4	Ice Cream Sandwich	December 16, 2011	15	3.7%
2.3.3–2.3.7	Gingerbread	February 9, 2011	10	4.1%
2.2–2.2.3	Froyo	May 20, 2010	8	0.2%

Source : wikipedia - 21/09/2015

ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total	
Small	6.2%						6.2%
Normal		10.6%		35.7%	19.2%	16.2%	81.7%
Large	0.5%	4.3%	1.7%	0.6%	0.6%		7.7%
Xlarge		3.7%		0.3%	0.4%		4.4%
Total	6.7%	18.6%	1.7%	36.6%	20.2%	16.2%	



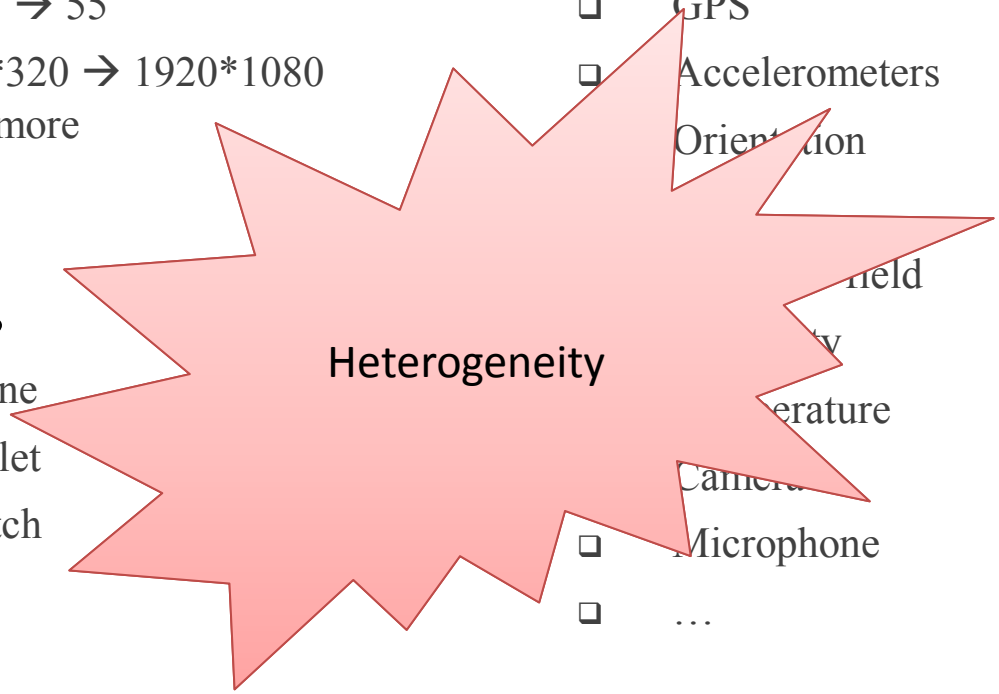


Plenty of Android based devices

- Screen size
  - 2.8'' → 55''
  - 240*320 → 1920*1080 and more
- Devices
  - Phone
  - Tablet
  - Watch
  - TV
  - Car
  - Game Console
  - …
- Features
  - GPS
  - Accelerometers
  - Orientation
  - Light
  - Magnetic field
  - Proximity
  - Temperature
  - Camera
  - Microphone
  - …

Plenty of Android based devices

- Screen size
  - 2.8'' → 55''
  - 240*320 → 1920*1080 and more
- Devices
  - Phone
  - Tablet
  - Watch
  - TV
  - Car
  - Game Console
  - …
- Features
  - GPS
  - Accelerometers
  - Orientation
  - …ield
  - …y
  - …erature
  - Camera
  - Microphone
  - …



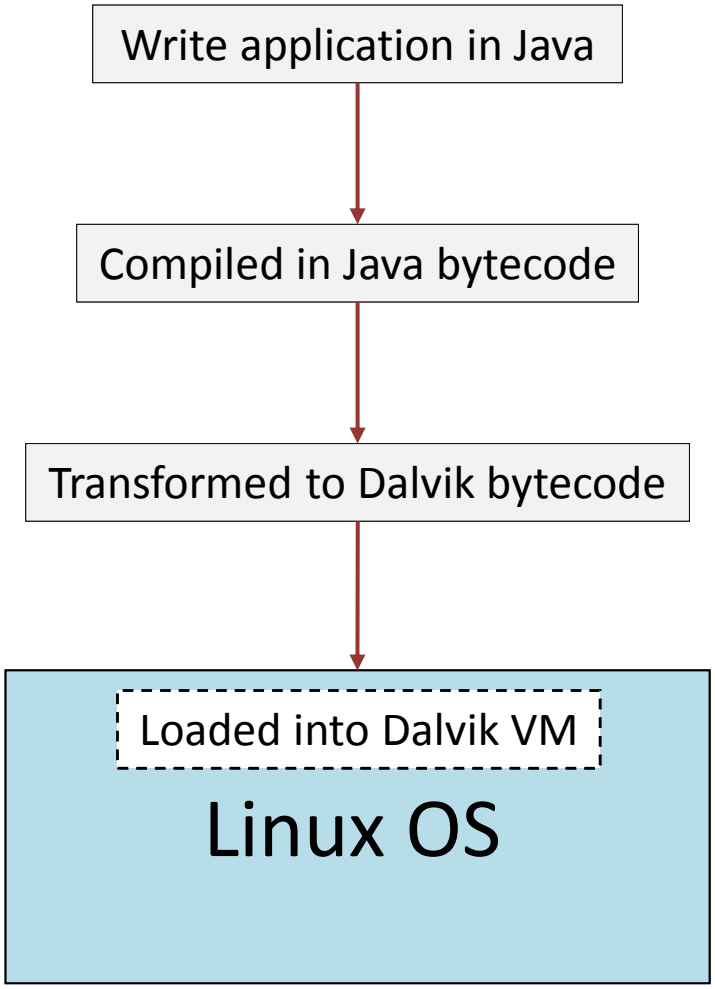
What is Android ?

- Open software platform for mobile development
- A complete stack – OS, Middleware, Applications
- Powered by Linux operating system
- Fast application development in Java
- Open source under the Apache 2 license

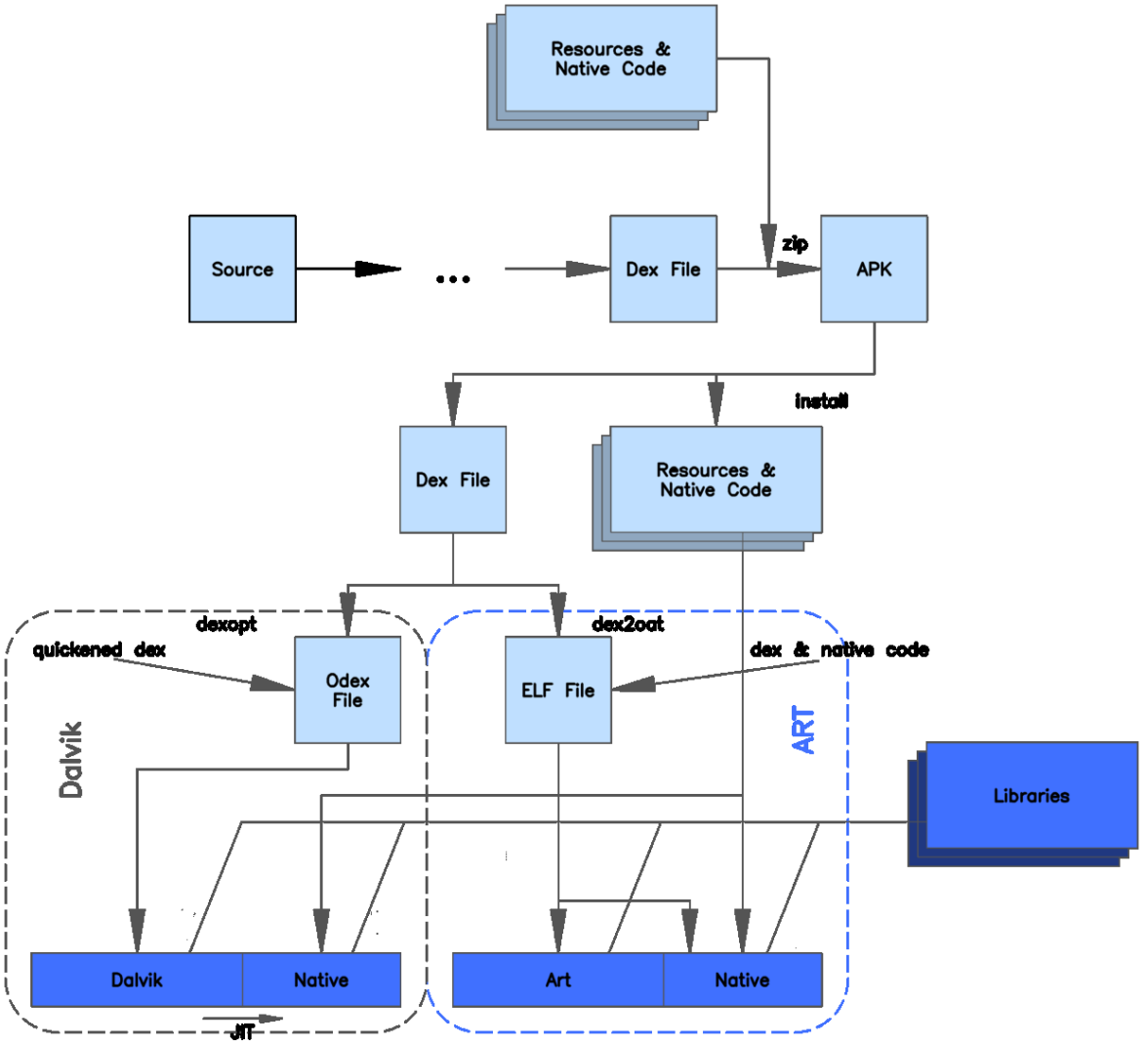
What is Android?

- Android is not a single piece of hardware
- It's a complete, end-to-end software platform that can be adapted to work on any number of hardware configurations.
- Everything is there, from the bootloader all the way up to the applications

Java development



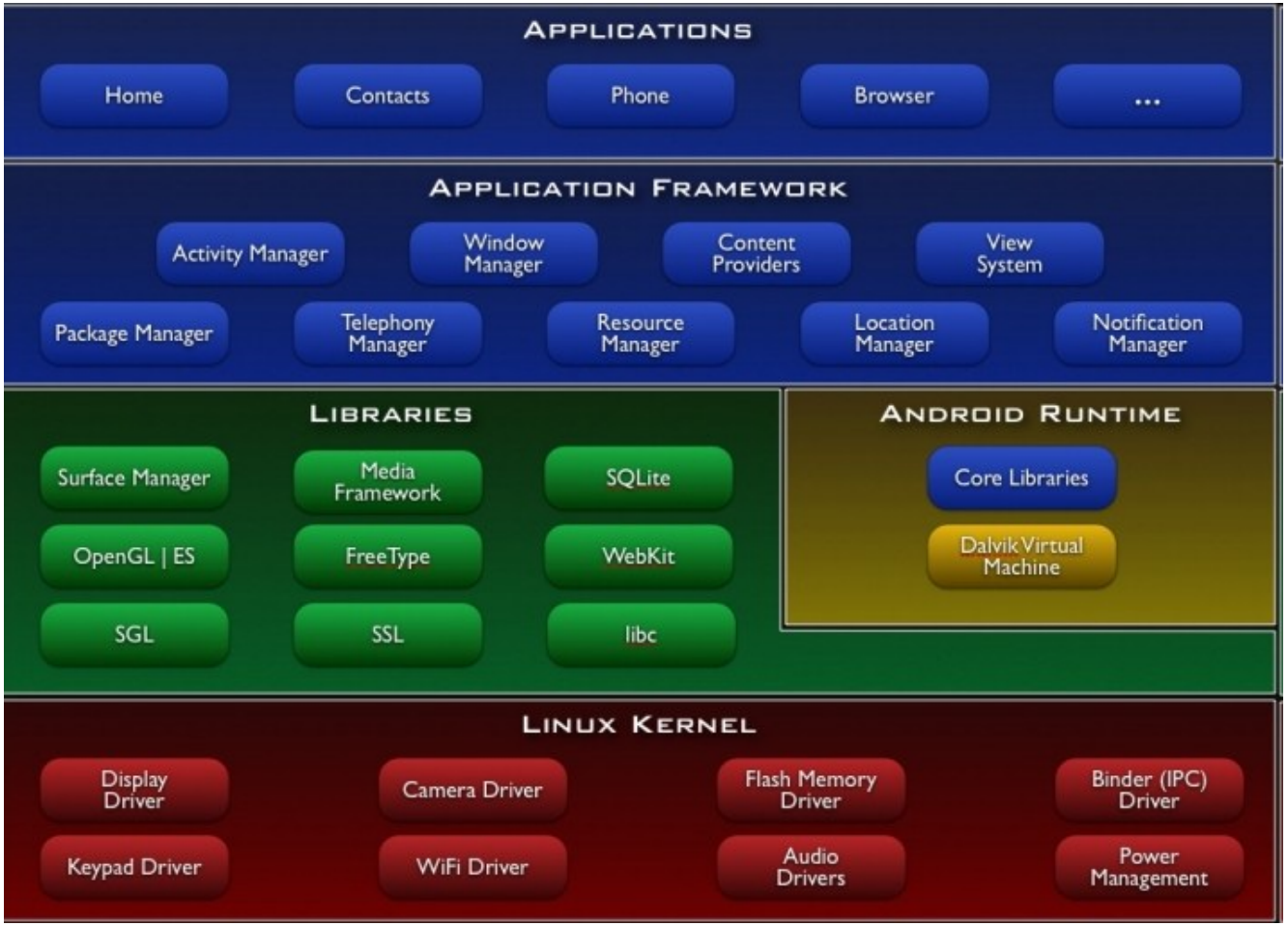
Java Development Environment from lollipop



Dalvik runtime

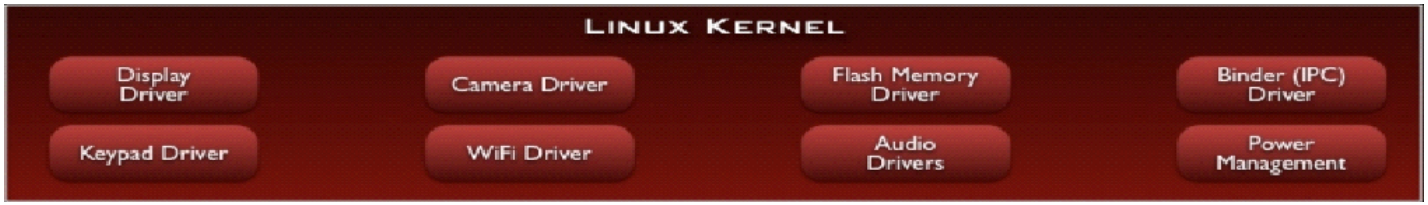
- Optimized for mobile (resource constraint) devices
  - Run multiple VMs efficiently
  - Minimal memory footprint
- Isolation between application
  - Applications run in separate VMs

Android Architecture



Linux Kernel

- Works as a HAL
- Device drivers
- Memory management
- Process management
- Networking

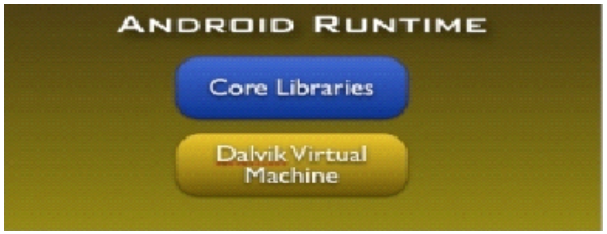


Libraries

- C/C++ libraries
- Interface through Java
- Surface manager – Handling UI Windows
- 2D and 3D graphics
- Media codecs, SQLite, Browser engine

Android Runtime

- Dalvik VM
  - Dex files
  - Compact and efficient than class files
  - Limited memory and battery power
- Core Libraries
  - Java 5 Std edition
  - Collections, I/O etc…

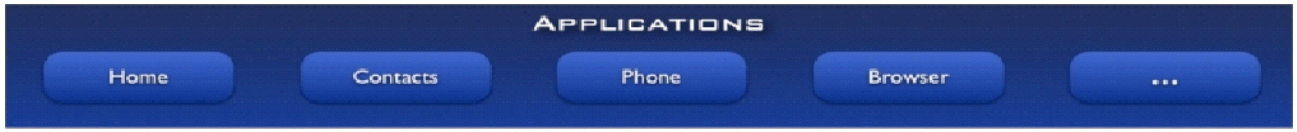


Application Framework

- API interface
- Activity manager – manages application life cycle.

Applications

- Built in and user apps
- Can replace built in apps

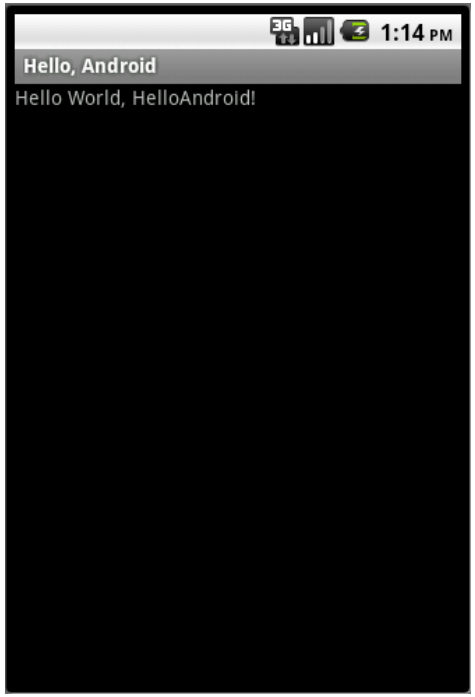


HelloWorld

```
package com.android.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```



Android Applications

- Android designed to enable reuse of components
- Each application can publish its capabilities
  - Other applications can use them

Concepts 1/2

- Activities
  - Each activity represents a single user interface
  - An activity uses one or more Views
  - May return one value
- Views
  - Display information to users
  - Interact with users
- Content Providers
  - Provide content to other applications
- Services
  - Provide support for background task

Concepts 2/2

- Intents
  - Message containers for inter-application communications
- Intent Filter
  - Specify which intents the application is interested in
- Broadcast Receivers
  - Applications that are registered to receive specific intents
- Permissions
  - Fine grained security mechanisms
    - Access contact information
    - Access GPS data
    - Access 3G connectivity
    - …

Application Lifecycle

- Application run in their own processes (VM, PID)
- Processes are started and stopped as needed to run an application's components
- Processes may be killed to reclaim resources

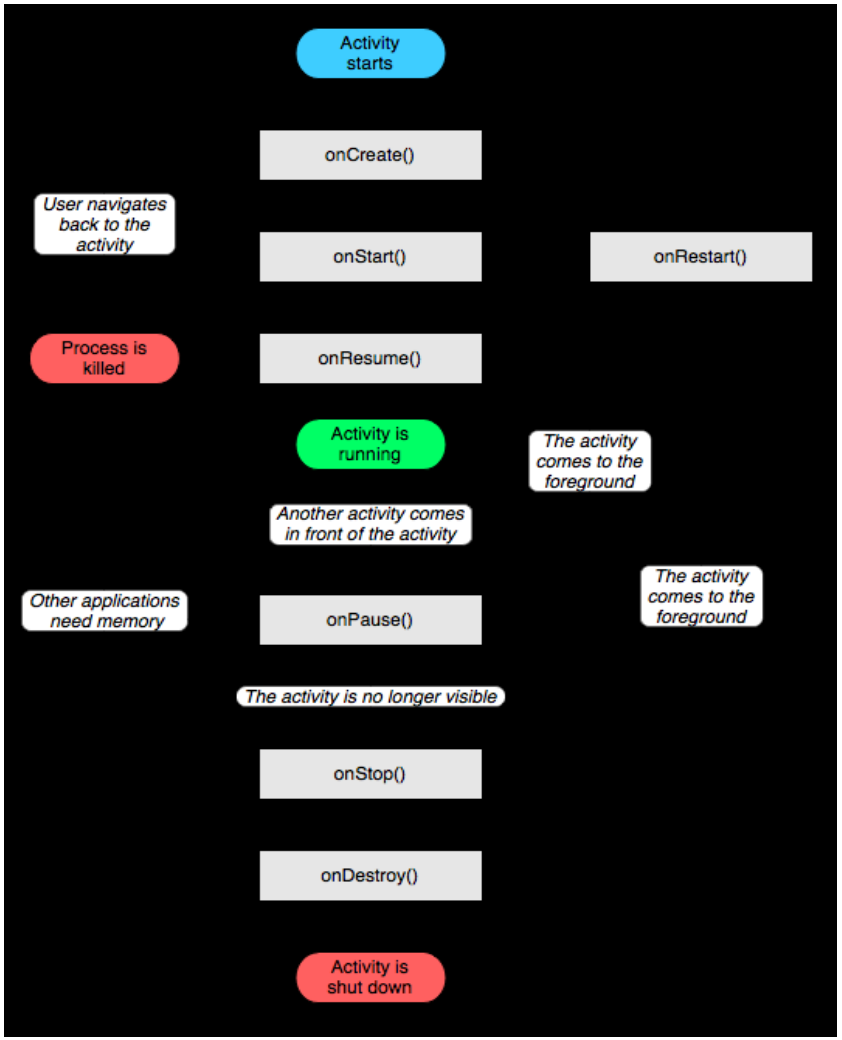
Activating components: intents

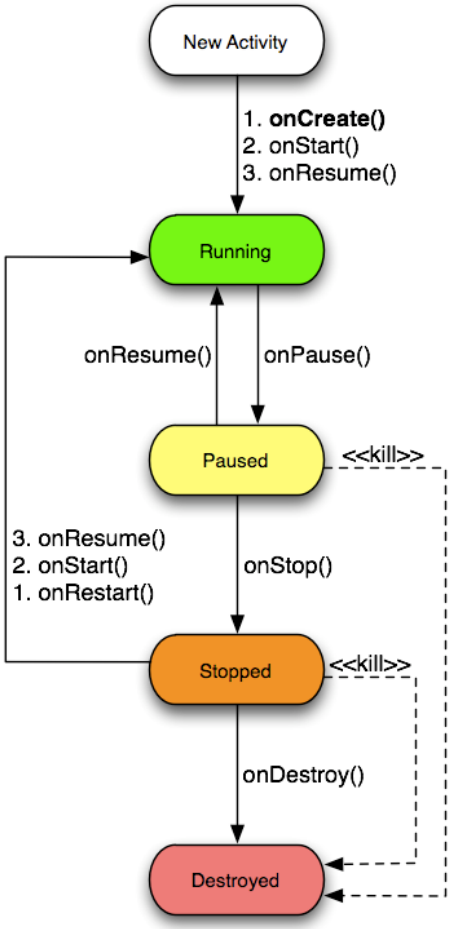
- Activity ➔ send Intent object
  - Context.startActivity() or Activity.startActivityForResult()
- Service ➔ send Intent object
  - Context.startService()
- Service ➔ send a bind request
  - Context.bindService()
- Broadcast Receivers ➔ send Intent object
  - Context.sendBroadcast()
- Content Provider only active when receiving request

Explicitly deactivating components

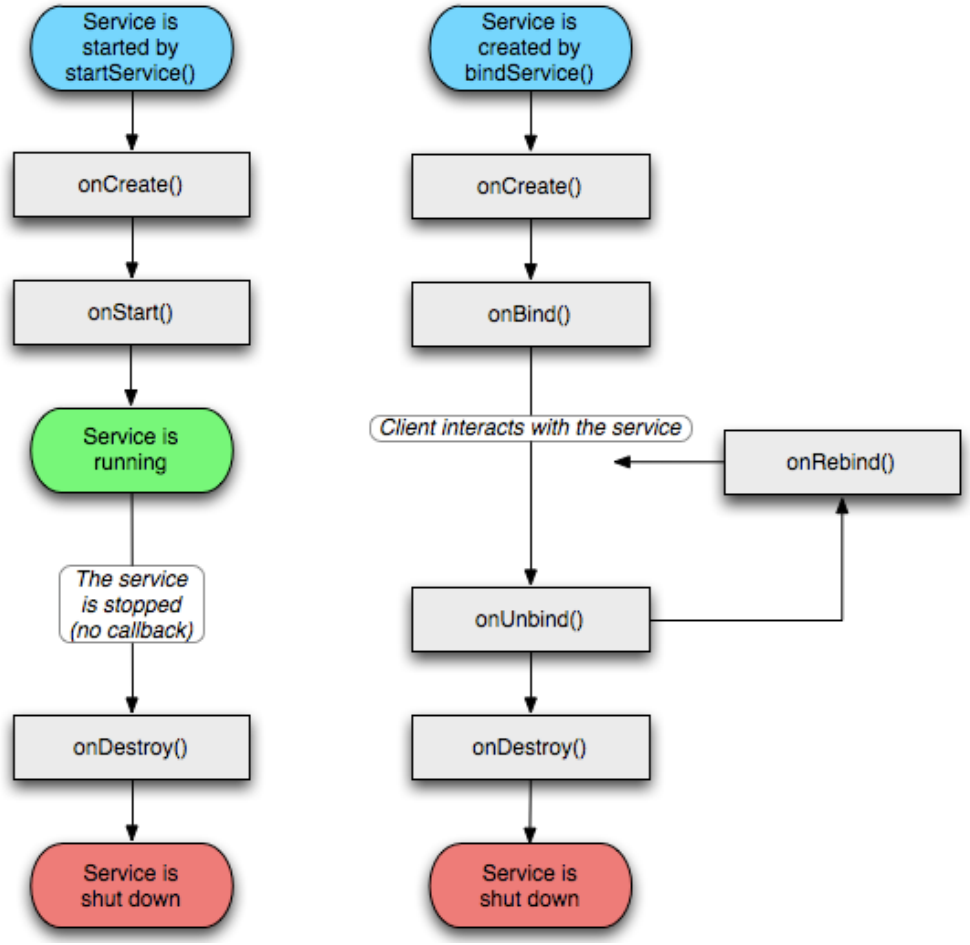
- Activity ➔ finish()
  - May finish another activity (started by yourself) finishActivity()
- Service ➔ stopSelf(), Context.stopService()

Activity Lifecycle





Service Lifecycle

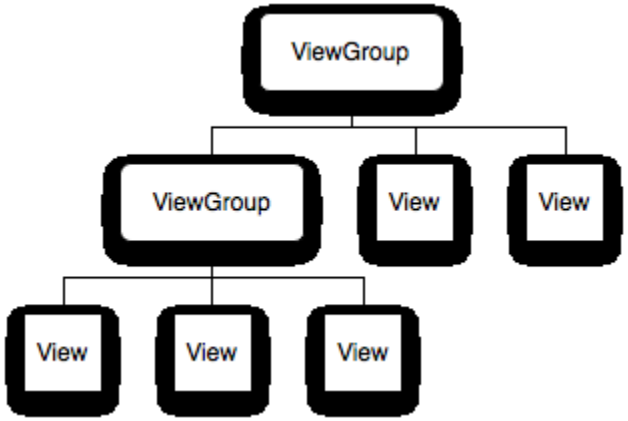


Broadcast Receiver

- One callback
  - onReceive()
- Boadcast Receiver lifecycle
  - A broadcast receiver is considered active only when onReceive() is executing

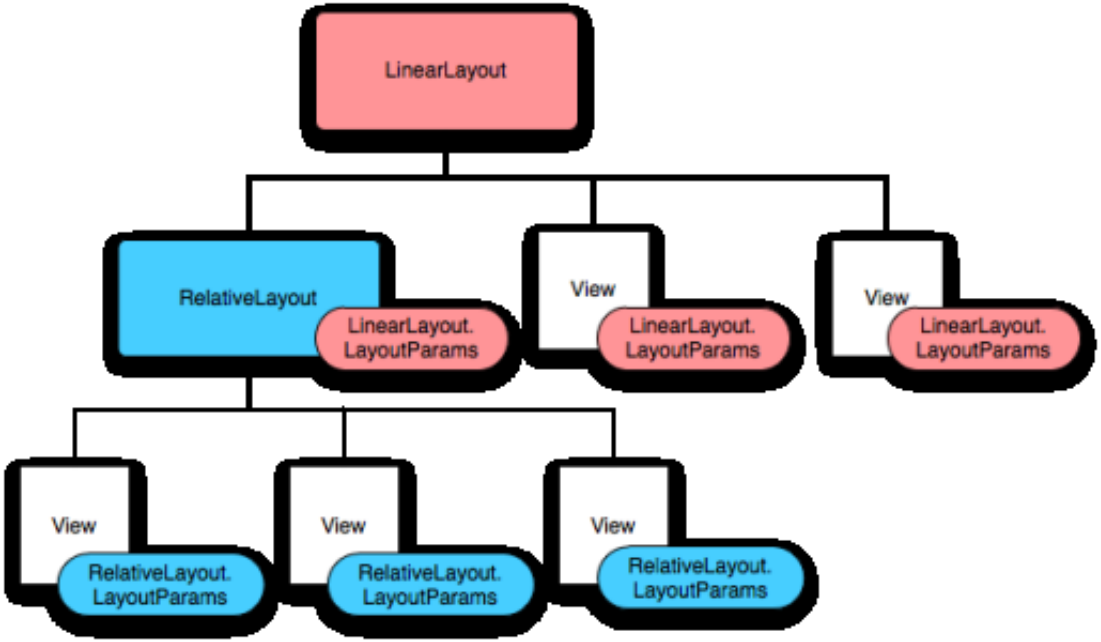
User Interface with Android

- User interface = one activity
- One activity = one class
- One activity ➔ hierarchy of views objects derived from the base View class
  - setContentView(View v) or setContentView(ViewGroup vg)

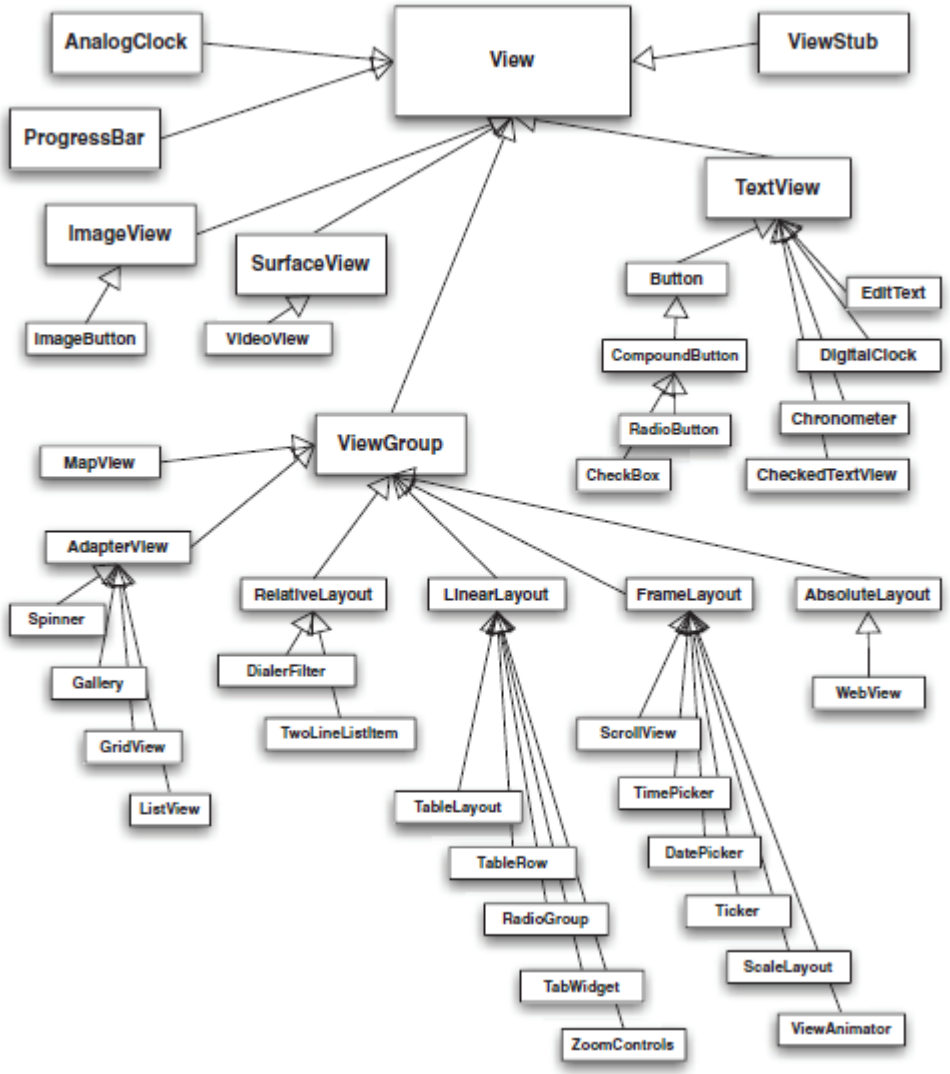


Layout or ViewGroup

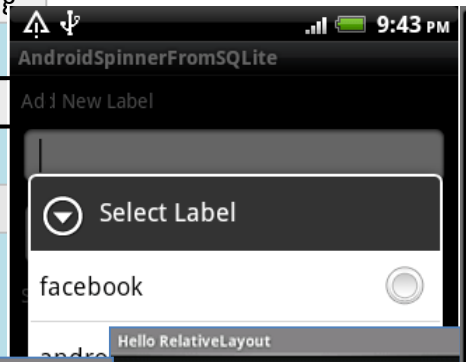
- Set the way to arrange sub-Views and sub-Layouts
- Hierarchical layouts
  - One layout composed of view and layout

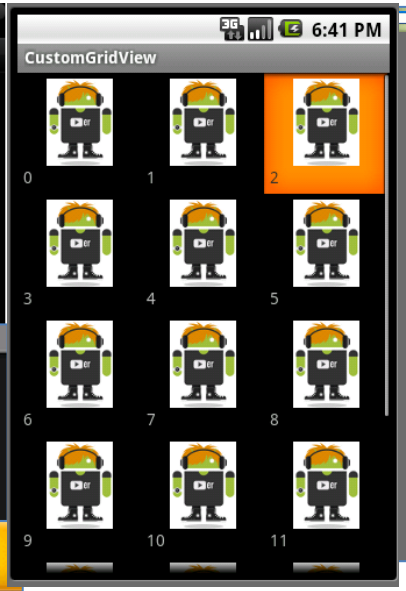


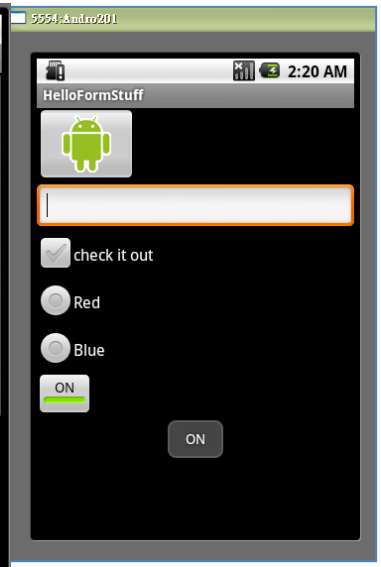
Views – region of the screen

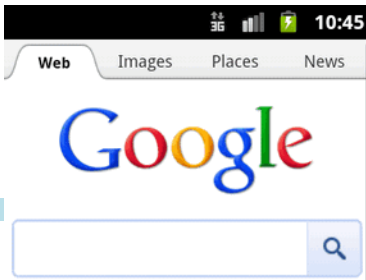


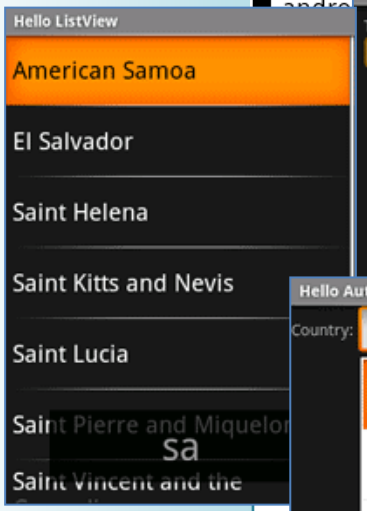
View examples

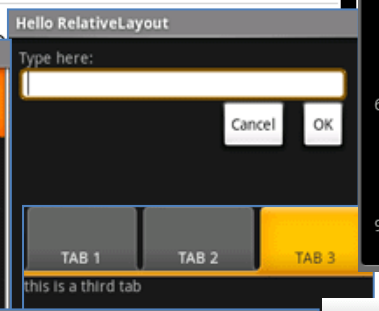


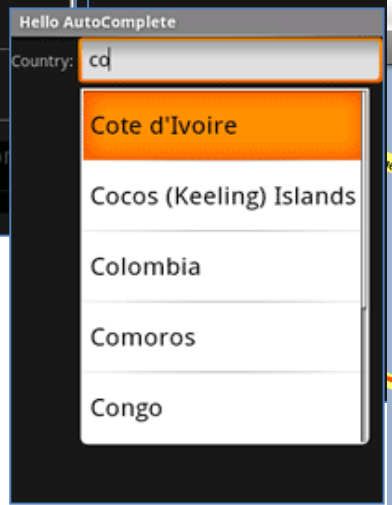


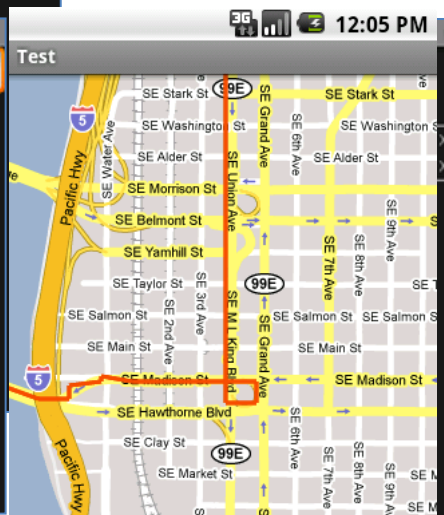


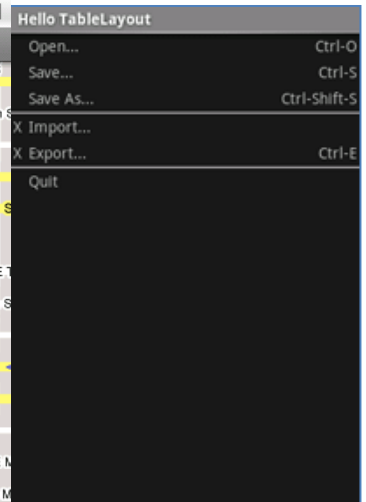


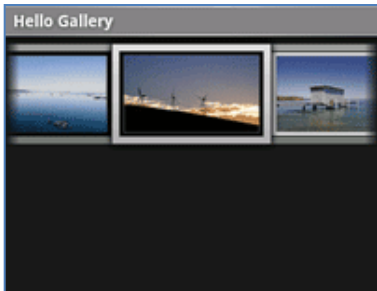












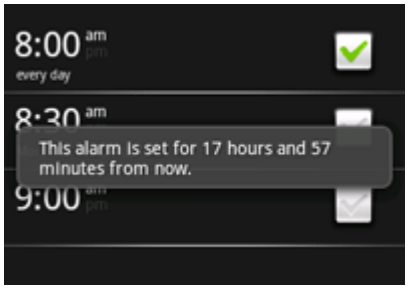
ECOLE SUPERIEURE D'INGENIEURS DE RENNES
Johann Bourcier

Handling UI Events

- UI Events = actions from the user on the interface
  - Touch, click, gesture …
- Event Listeners = single callback on specific user actions
  - onClick(), onLongClick(), onFocusChange(), onKey(), onTouch(), onCreateContextMenu().
- Register listener for a specific view
  - button.setOnClickListener(myListener)

Notifying user

- Toast notifications
  - pops up on the surface of the window



- Status Bar Notification
  - adds an icon to the system's status bar

XML based definition of UI

It is possible to define your UI in an XML file

/res/layout/myUI.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:text="@string/hello"/>
```

/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, Android! I am a string resource!</string>
    <string name="app_name">Hello, Android</string>
</resources>
```

XML based definition of UI

- Load the XML file

```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

XML based UI

- How to reference View defined in the XML in your Java code?
  - FindViewById(ID)

```
<Button android:id="@+id/buttonSearch"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/buttonSearch" android:width="100px"/>
```

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button button = (Button)findViewById(R.id.buttonSearch);
    button.setOnClickListener(this);
}
```

Anonymous Class

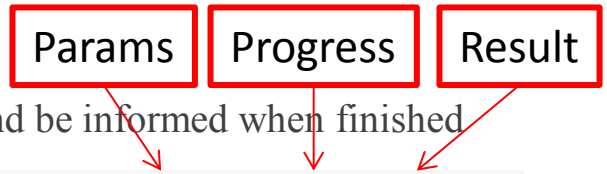
- **Callback:** your code needs to be notified when something in the UI changes.
  - Ex. a button is pushed and we need to change state, new data has arrived from the network and it needs to be displayed
- Java provides idiom to pass blocks in code
- Anonymous classes are a handy tool for expressing many kinds of code blocks.

Without anonymous class	With anonymous class

public class myDataModel{
//Callback class
private class keyHandler implements View.onKeyListener {
public boolean onKey(View v, int keyCode, KeyEvent event) {
handleKey(v, keyCode, event);
}
}
/* @param view in the view we model */
public myDataModel(View view){
view.setOnKeyListener(new KeyHandler())
}
/** Handle a key event **/
void handleKey(View v, int keyCode, KeyEvent event){
// key handling code goes here …
}
}
	public class myDataModel{
	
	/* @param view in the view we model */
	public myDataModel(View view) {
	view.setOnKeyListener(
	// this is an anonymous class!!
	new View.OnKeyListener() {
	public boolean onKey(View v, int keyCode, KeyEvent event) {
	handleKey(v, keyCode, event);
	}
	});
	/** Handle a key event **/
	void handleKey(View v, int keyCode, KeyEvent event){
	// key handling code goes here …
	}
	}

Asynchronous task

- Async task
  - Used to launch a background task and be informed when finished




```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            totalSize += Downloader.downloadFile(urls[i]);
            publishProgress((int) ((i / (float) count) * 100));
            // Escape early if cancel() is called
            if (isCancelled()) break;
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
        showDialog("Downloaded " + result + " bytes");
    }
}
```

Asynchronous task

- Async task
  - Usage :
  - Steps :
    - onPreExecute()
    - doInBackground(Params...)
    - onProgressUpdate(Progress...)
    - onPostExecute(Result)
  - You can use publishProgress(progress) in the doInBackground(Params...) method to publish the progress of the background task.

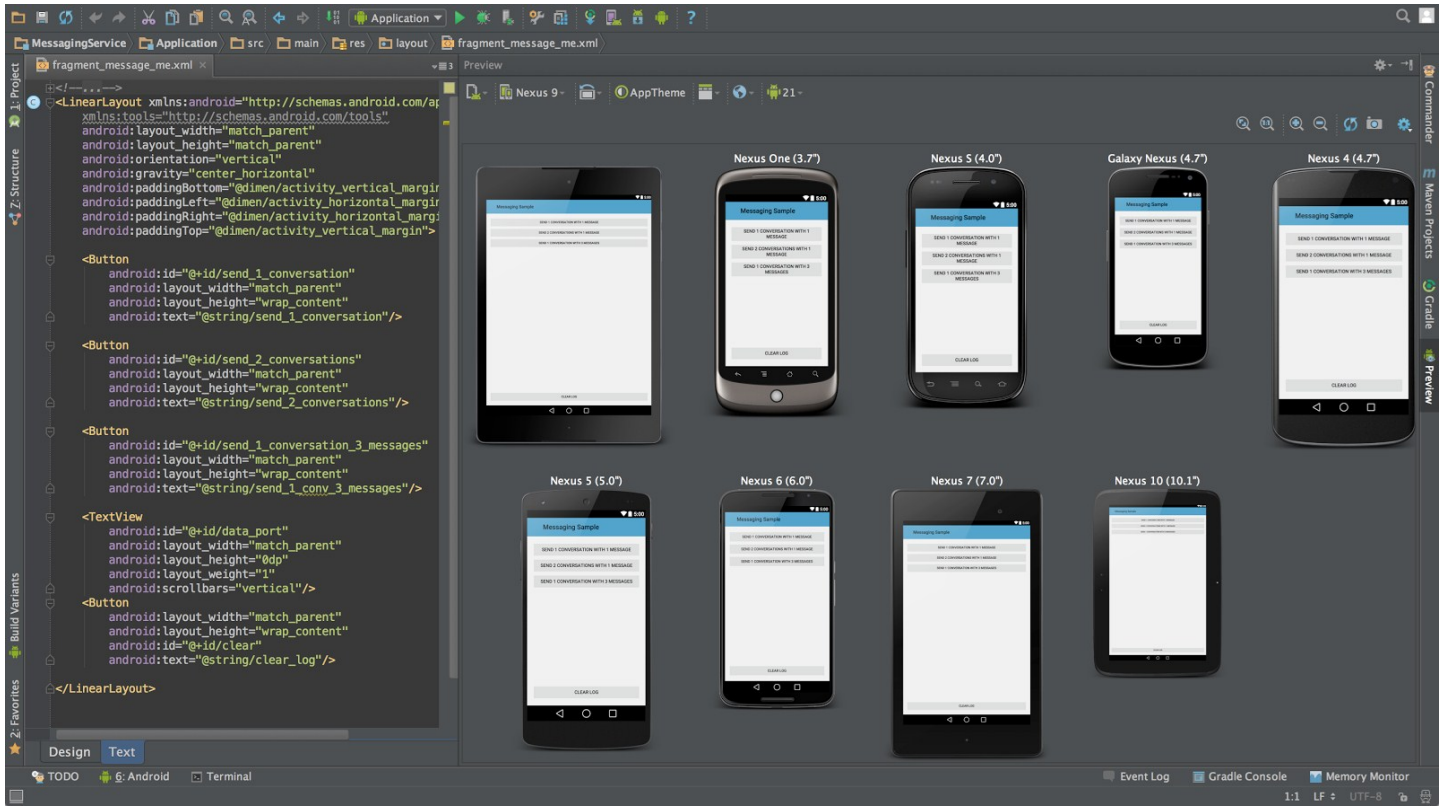
```
new DownloadFilesTask().execute(url1, url2, url3);
```

Android SDK

- Android SDK
  - Debugger
  - Libraries
  - Handset Emulator
  - Documentation
  - Sample Code
- Android Virtual Device (AVD)
  - Support the definition of Virtual devices to test your applications
- Fully integrated with IntelliJ

Android developer's website :
http://developer.android.com/index.html

ANDROID Studio



Android Emulator