




Programming with Android:
First Steps

Marco Di Felice

Dipartimento di Informatica – Scienze e Ingegneria

Università di Bologna

Android Applications Design

TUTORIAL in ENGLISH

http://www.cs.unibo.it/projects/android/2014/

http://developer.android.com/training/index.html

TUTORIAL in ITALIAN

http://www.html.it/guide/guida-android/

http://www.mrwebmaster.it/android/guide/guida-sviluppo-apps-android/

Android … What?

Android … What?

Android … What?

SMART FRIDGE

ANDROID MICROWAVE

SMARTPHONES

TABLETS

EREADERS

ANDROID TV

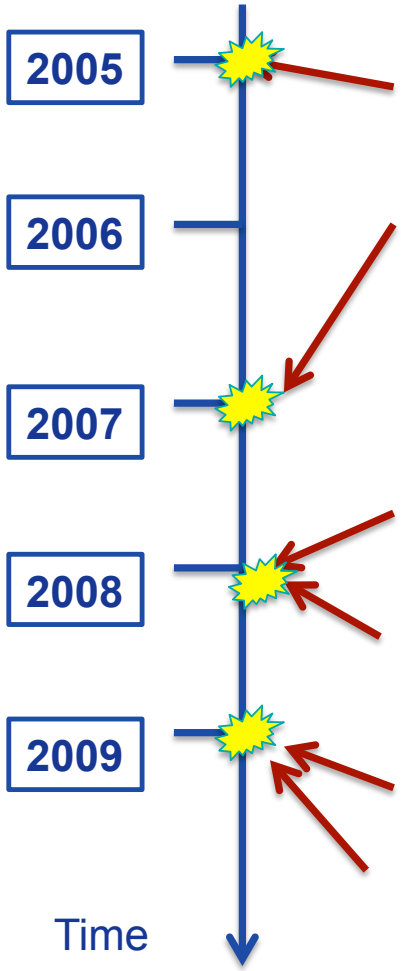
GOOGLE GLASSES

?


Android ... What?
50:50
CHI VUOL ESSERE
MILIONARIO
CHI VUOL ESSERE
Q. Who made Android ?
A. Microsoft
B. Apple
1
C. Google
D. Samsung

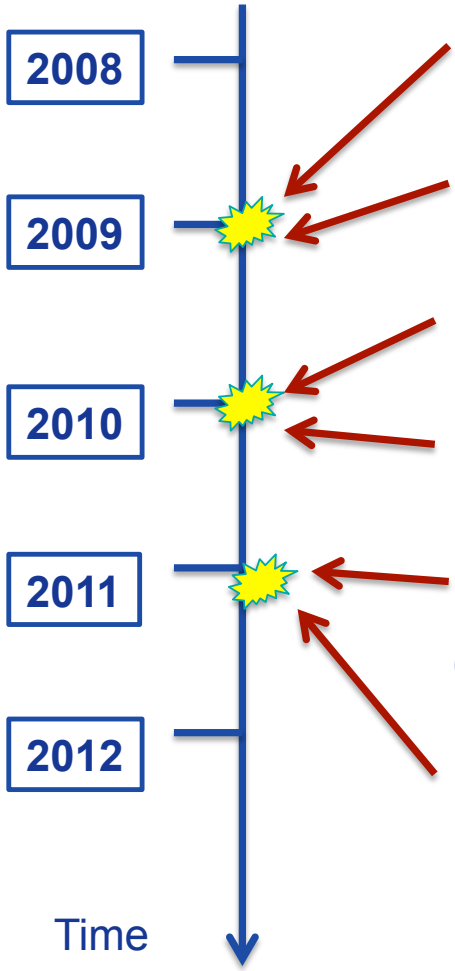

Android ... What?
50:50
CHI VUOL ESSERE
MILIONARIO
CHI VUOL ESSERE
Q. Who made Android ?
A. Microsoft
B. Apple
1
C. Google
D. Samsung

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?



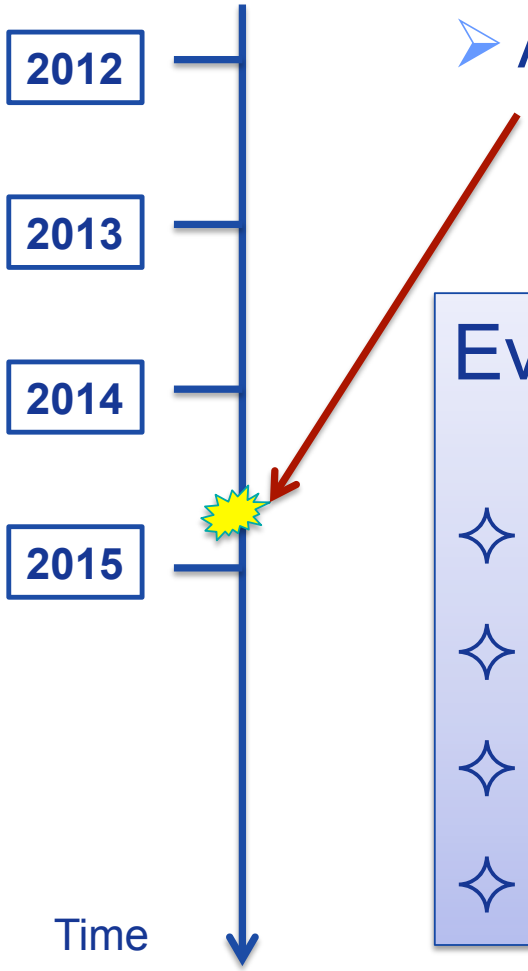
- Android **1.6** (**Donut**) Released
- Android **2.0** (**Eclair**) Released

- 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?

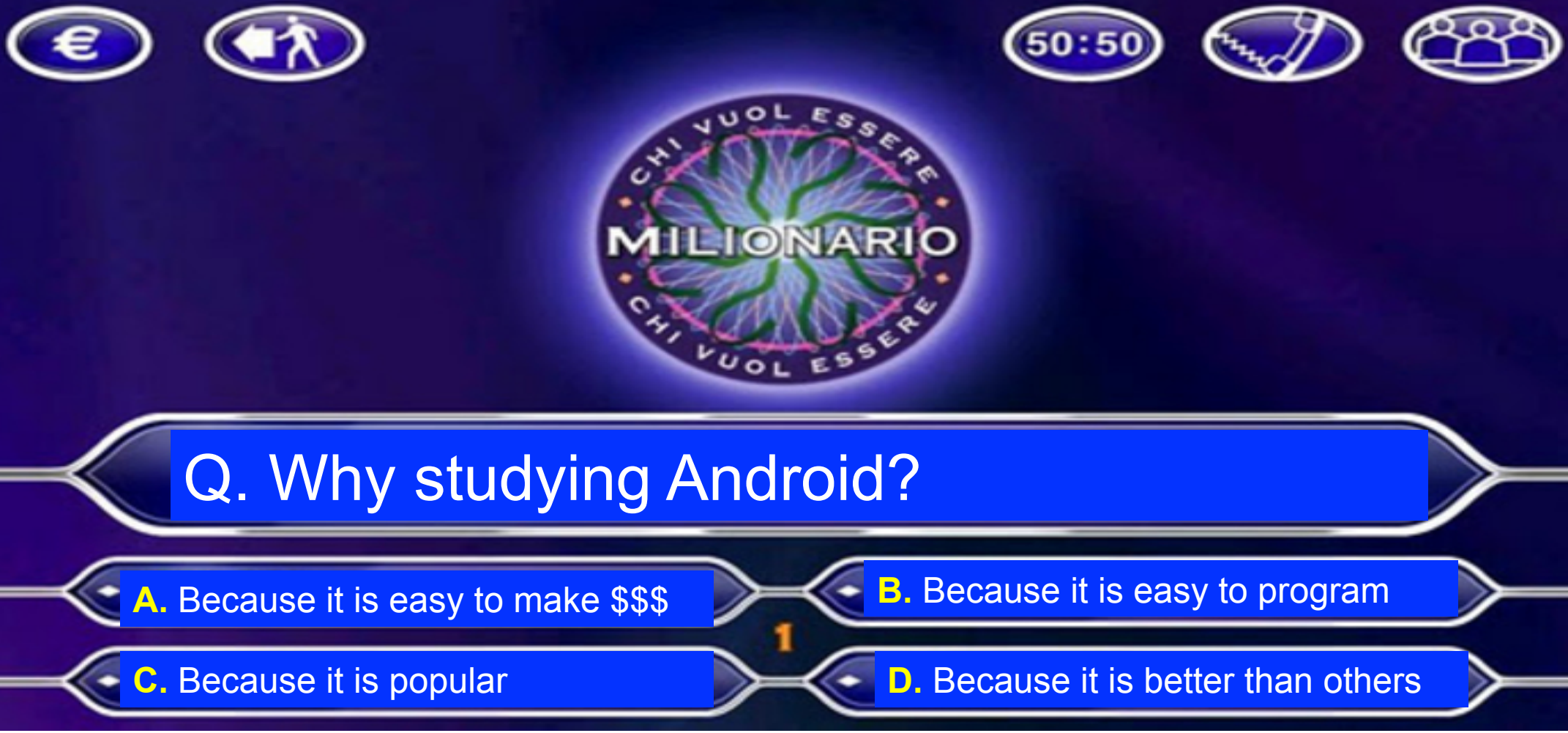


- Android **5.0** (**Lollipop**) Released

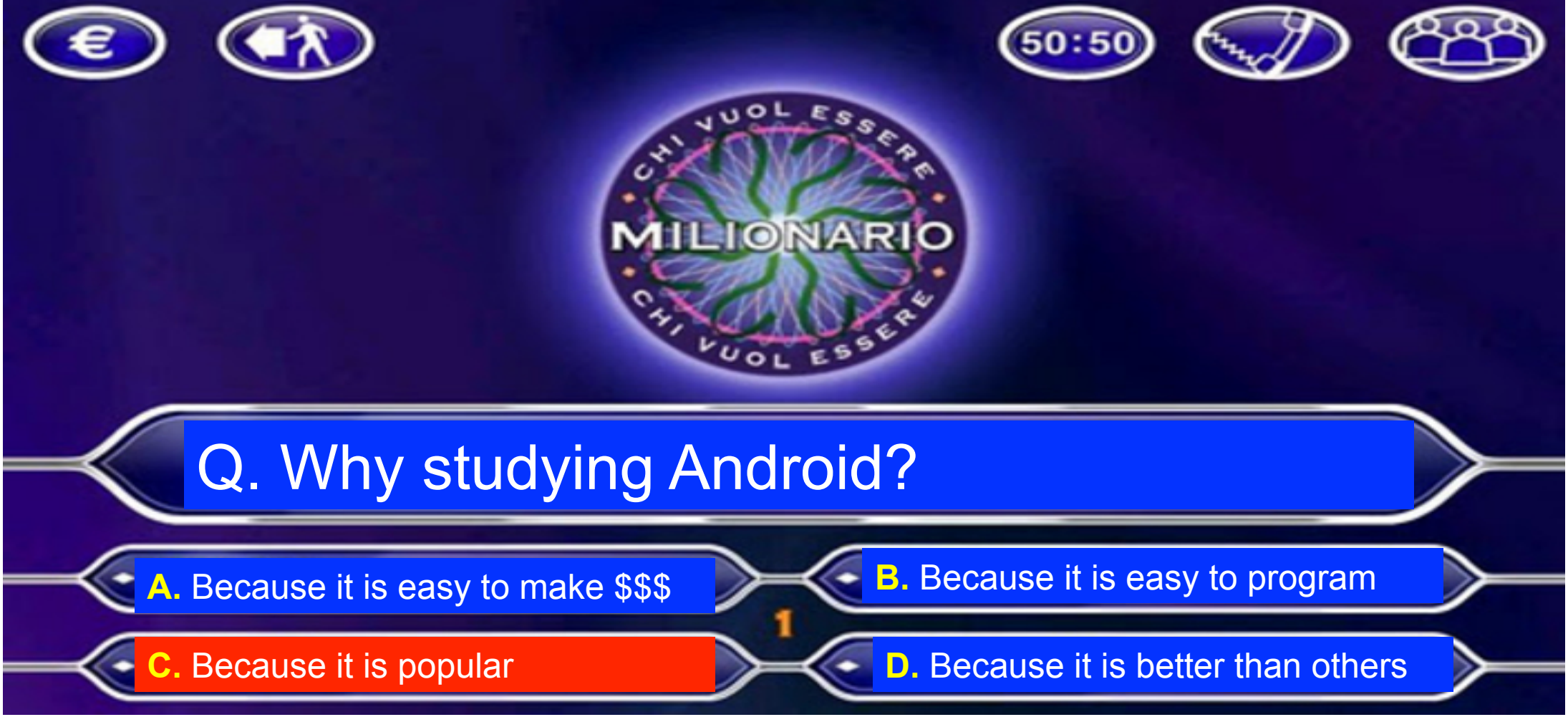
Every Android version has:

- A **name** (Lollipop)
- A **logo**
- New **features** of the operating system
- New **API** (Application Programming Interface)

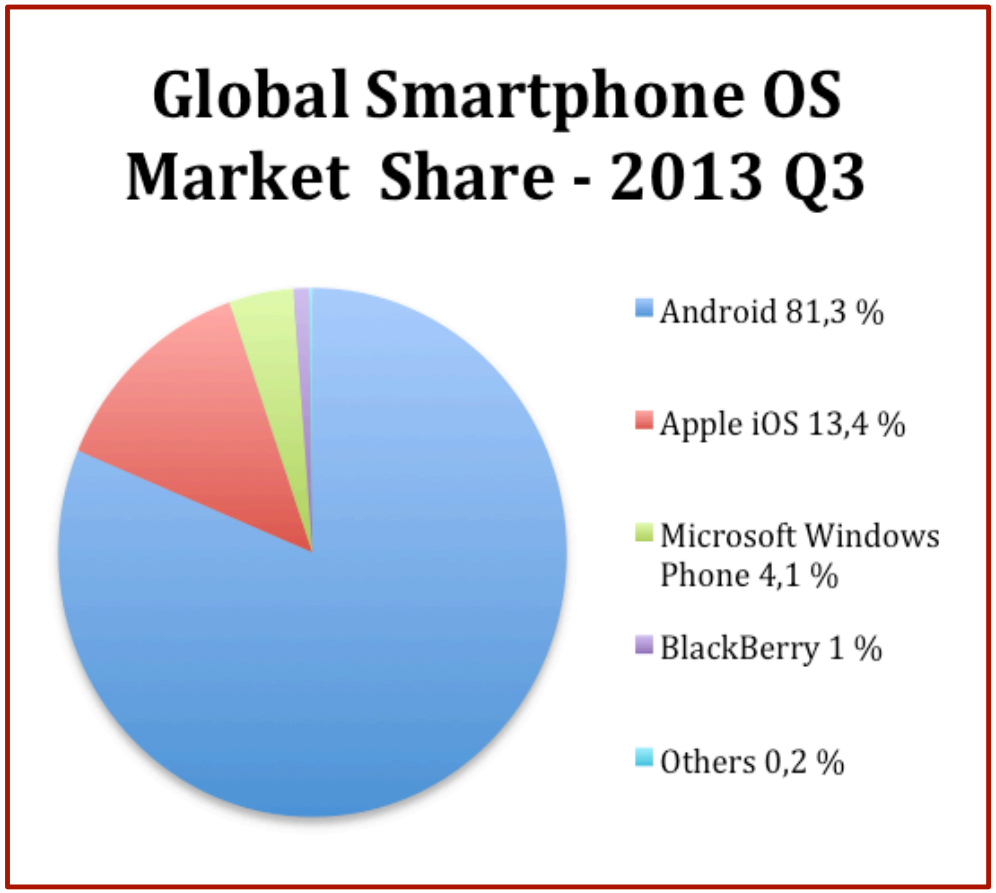
Android … Why?

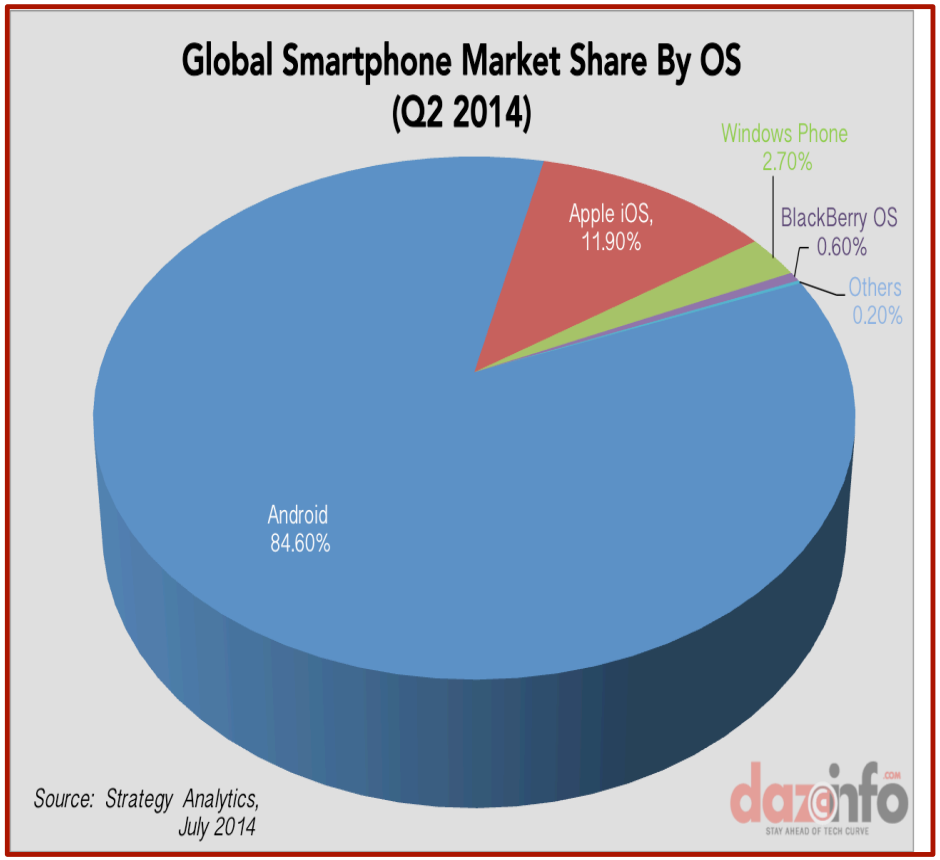

50:50
CHI VUOL ESSERE
MILIONARIO
CHI VUOL ESSERE
Q. Why studying Android?
A. Because it is easy to make $$$
B. Because it is easy to program
1
C. Because it is popular
D. Because it is better than others

Android … Why?


50:50
CHI VUOL ESSERE
MILIONARIO
CHI VUOL ESSERE
Q. Why studying Android?
A. Because it is easy to make $$$
B. Because it is easy to program
1
C. Because it is popular
D. Because it is better than others

Android ... Why?



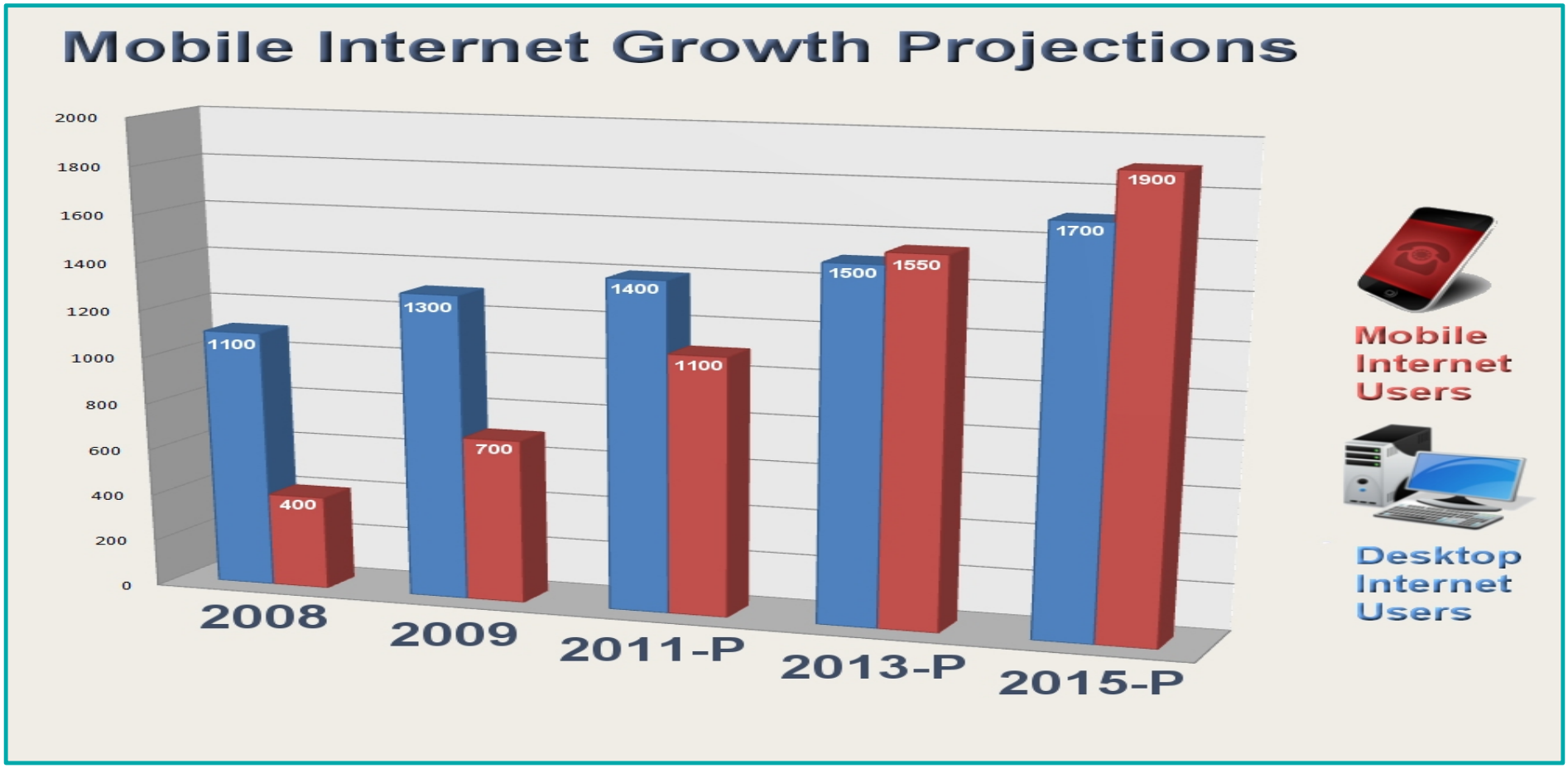


2013 Market Share

www.gartner.com

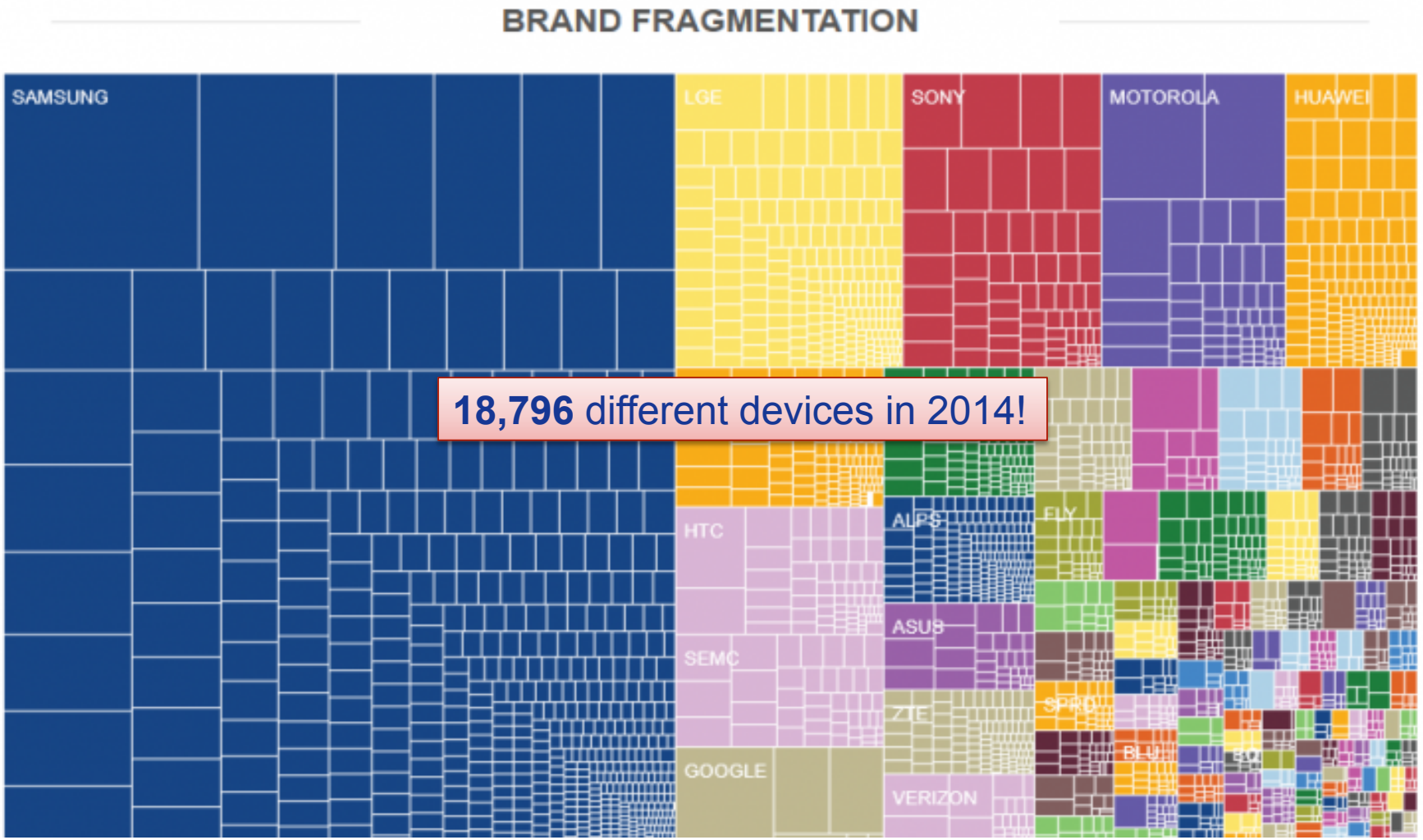
2014 Market Share

Android ... Why?



Sorgente: http://jeffalangray.com

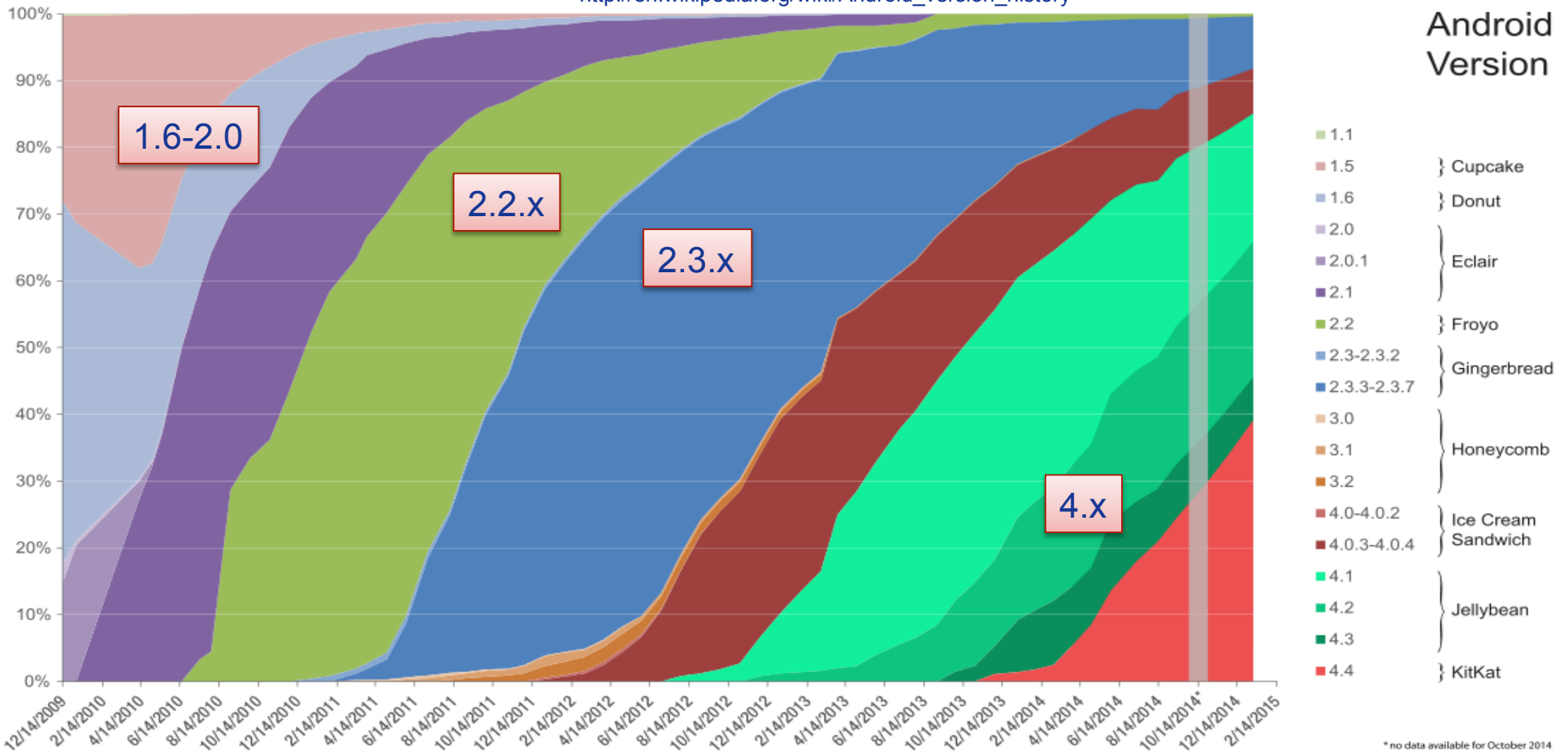
Android … Why?



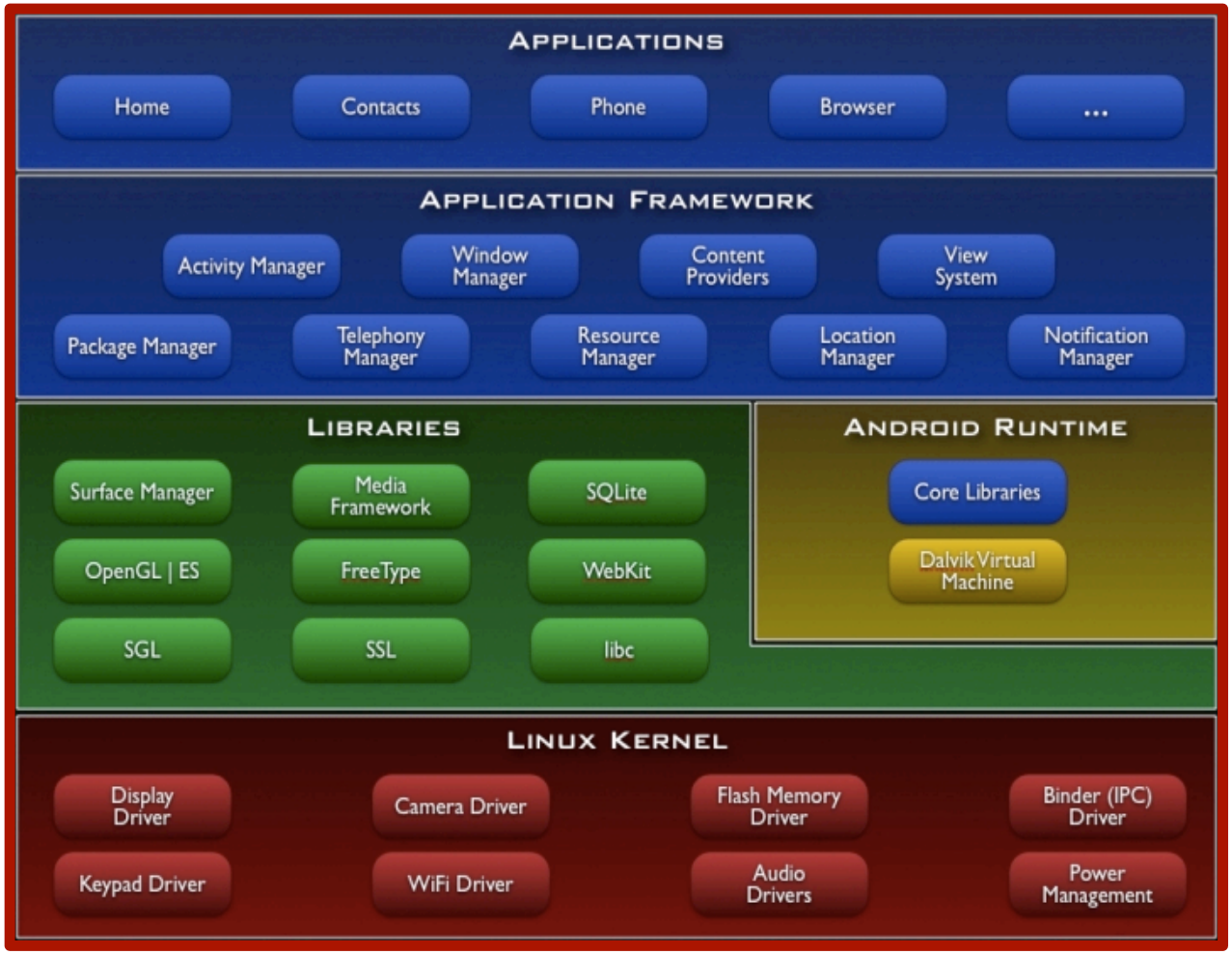
http://thenextweb.com/mobile/2014/08/21/18796-different-android-devices-according-opensignals-latest-fragmentation-report/

Android ... How?

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



The Android Architecture

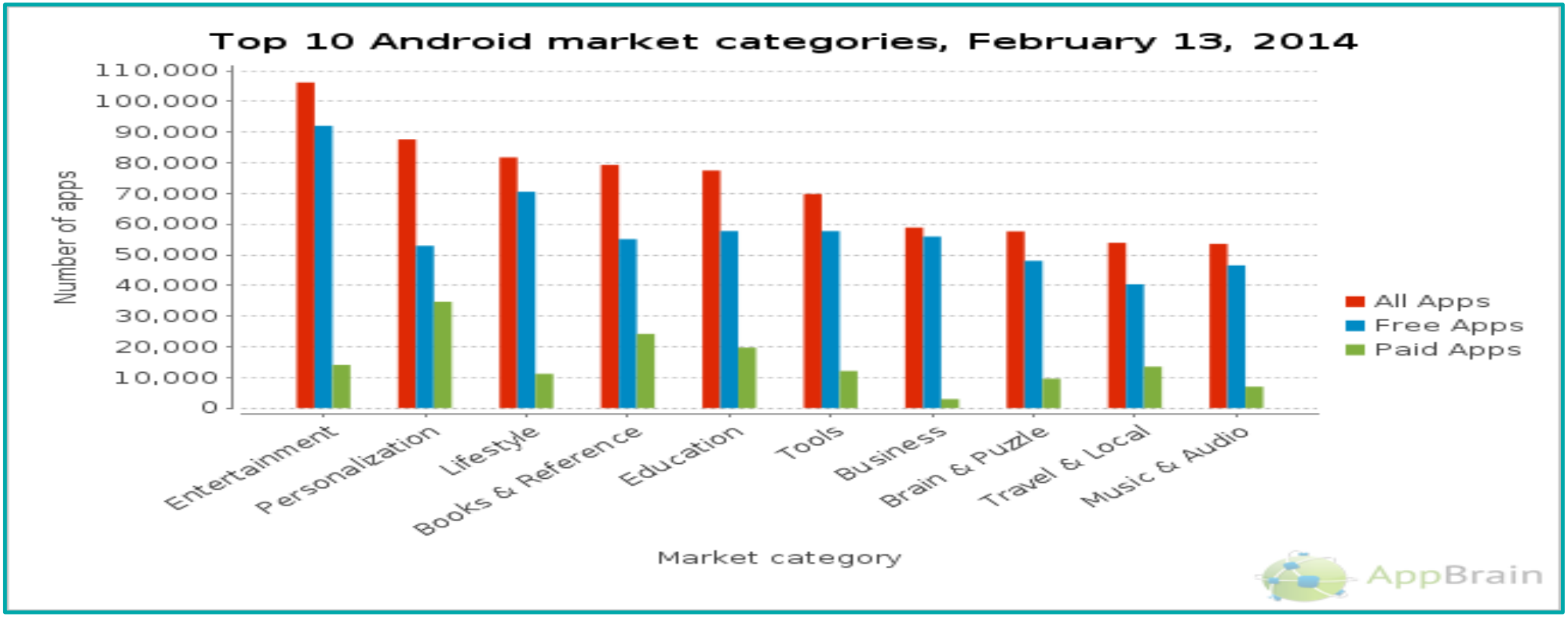


We are **here**!

Android *Architecture*

Android Applications

ANDROID APP CATEGORIES



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

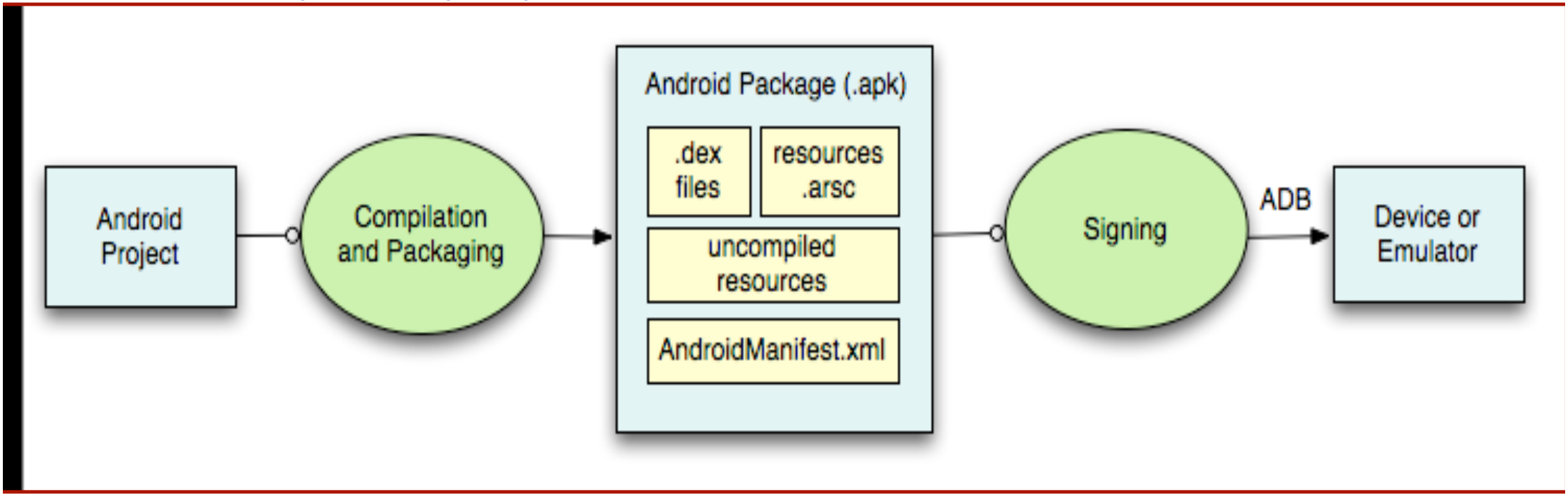
Android Applications Design

APPLICATION **DESIGN**:

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

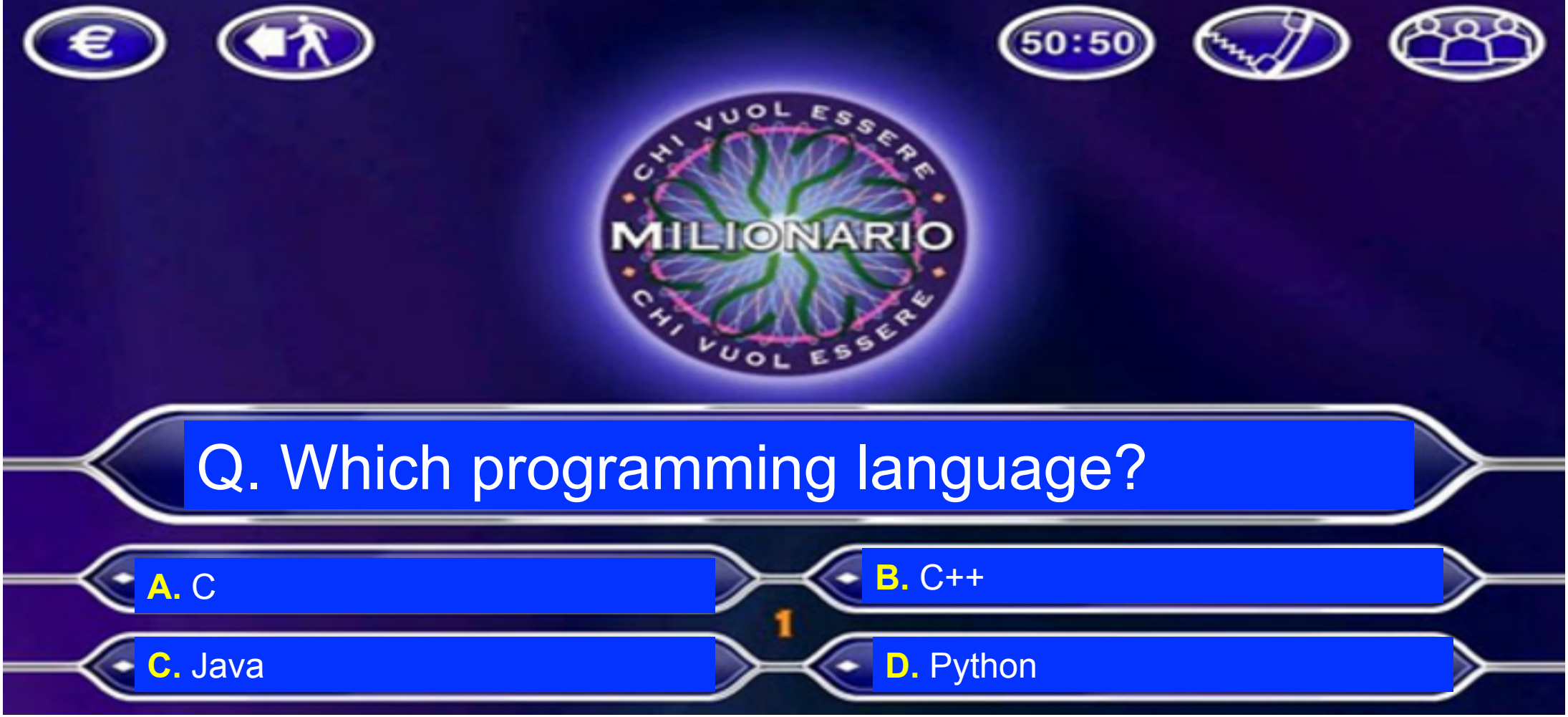
Android Applications: Development

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



✧An IDE like **Android Studio** handles the entire **development process**

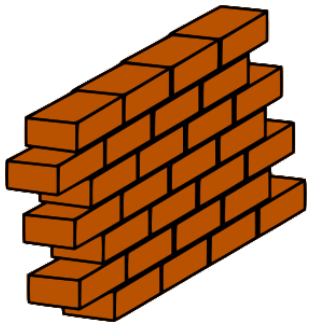
Android … What?

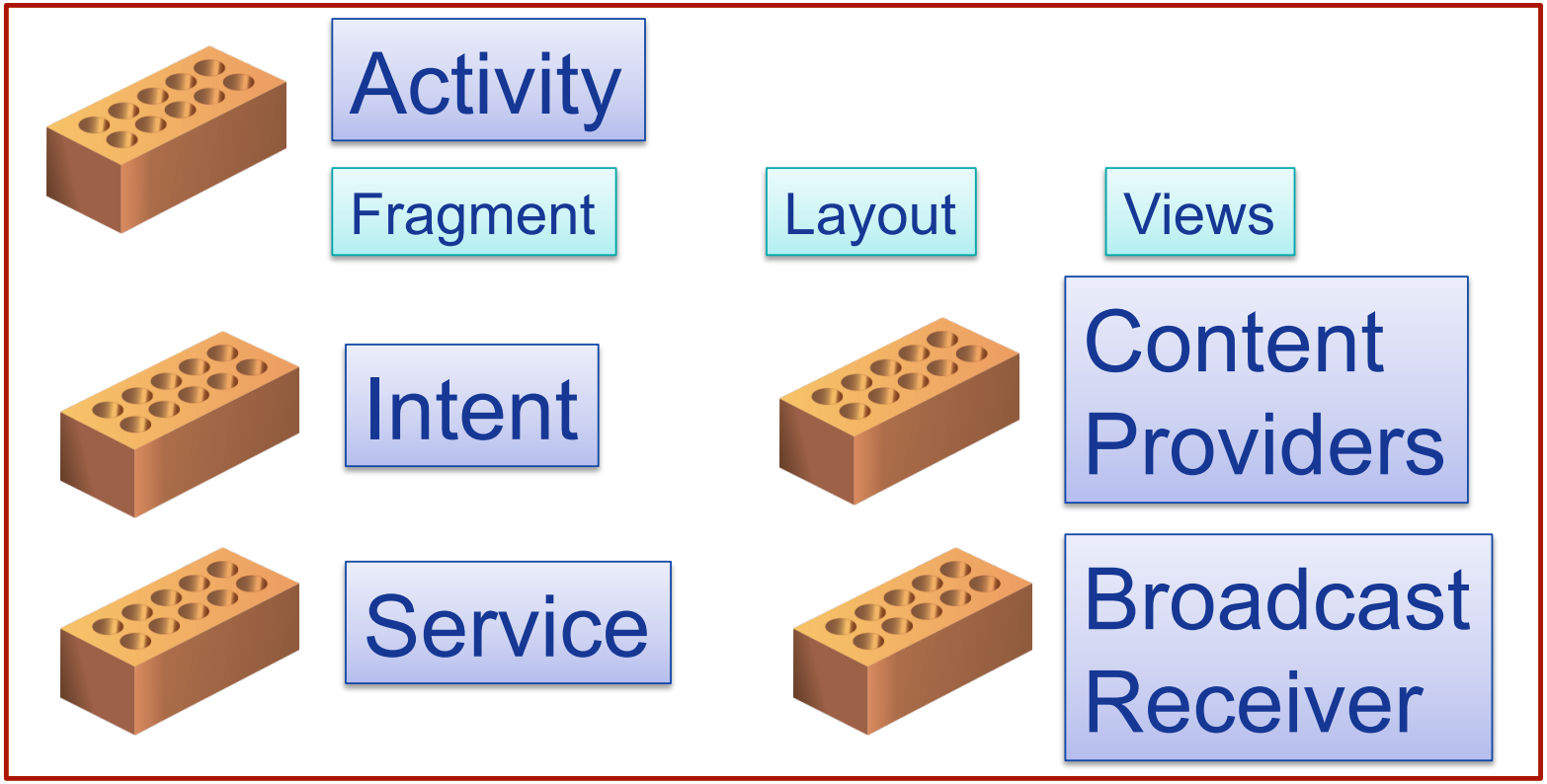



Android ... What?
50:50
CHI VUOL ESSERE
MILIONARIO
CHI VUOL ESSERE
Q. Which programming language?
A. C
B. C++
C. Java
D. Python

Android Applications Design

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





Android Applications Design

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

Android System Services

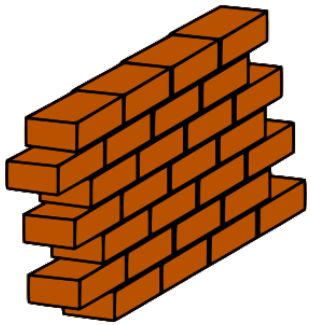
- WiFi Service
- Embedded Sensor Service
- Notification Manager Service
- …

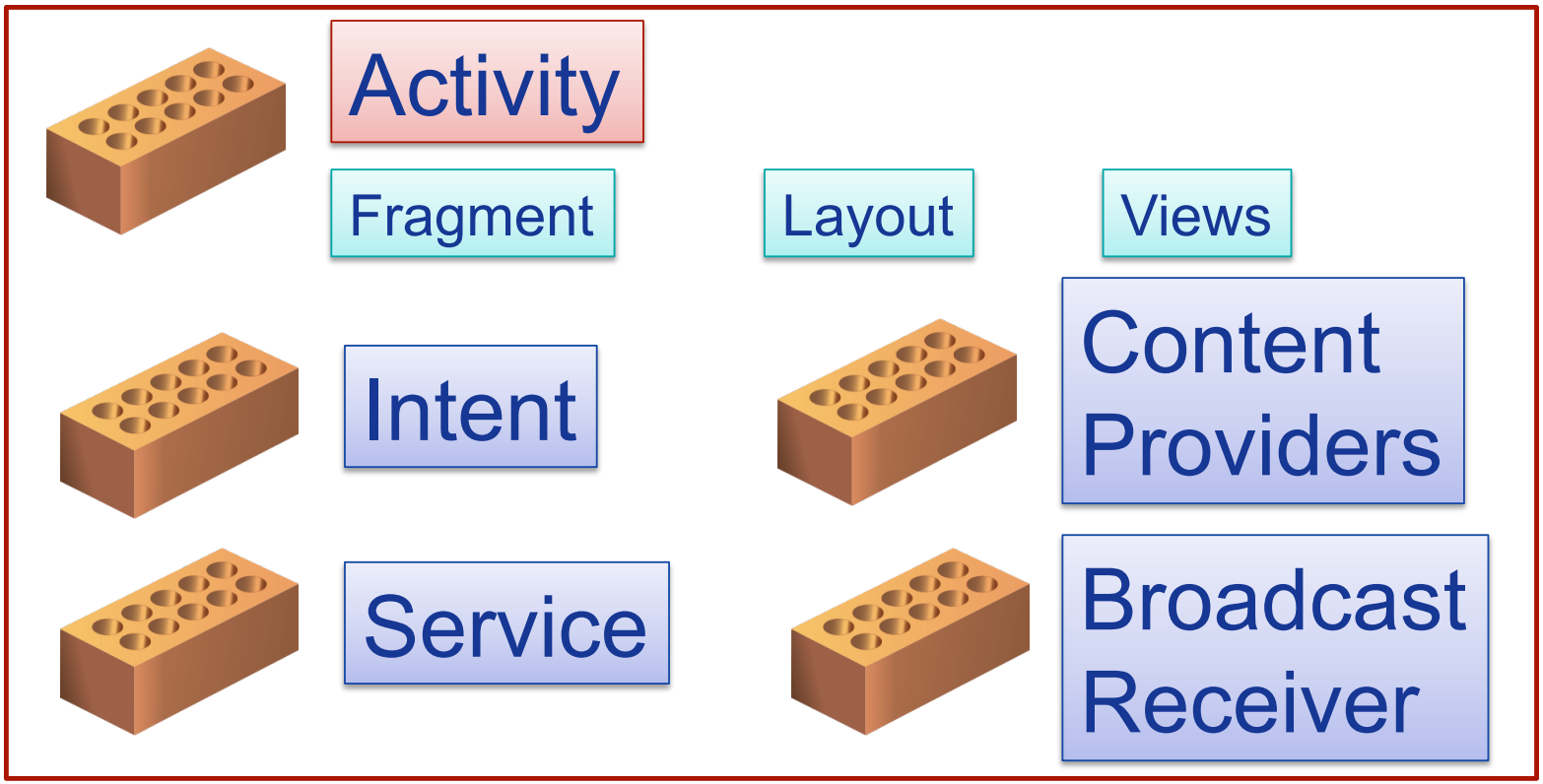
Google Play Libraries

- 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** …



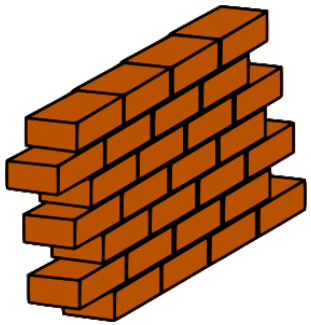


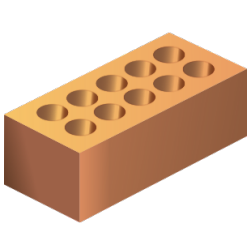
Android Components: Activities

- 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 Applications Design

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



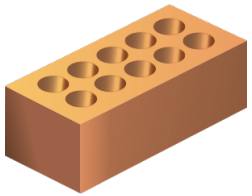


Activity

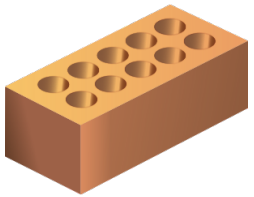
Fragment

Layout

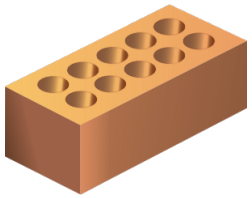
Views



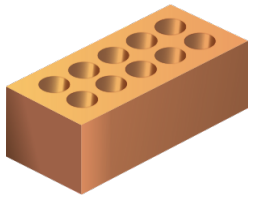
Intent



Content Providers



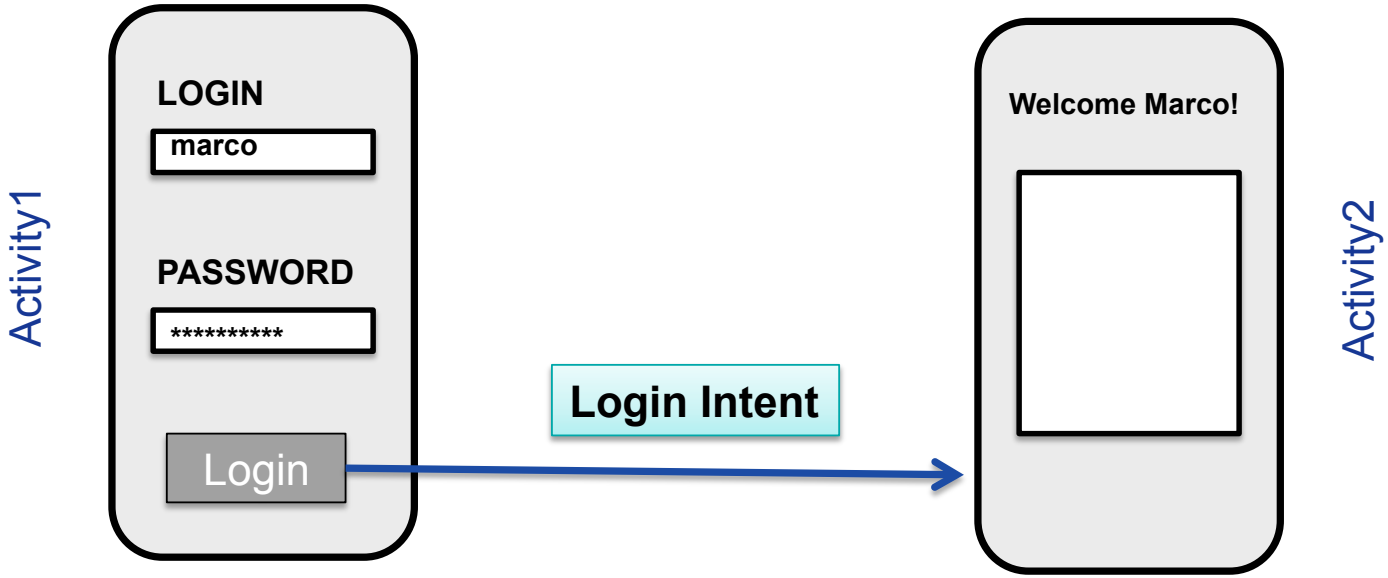
Service



Broadcast Receiver

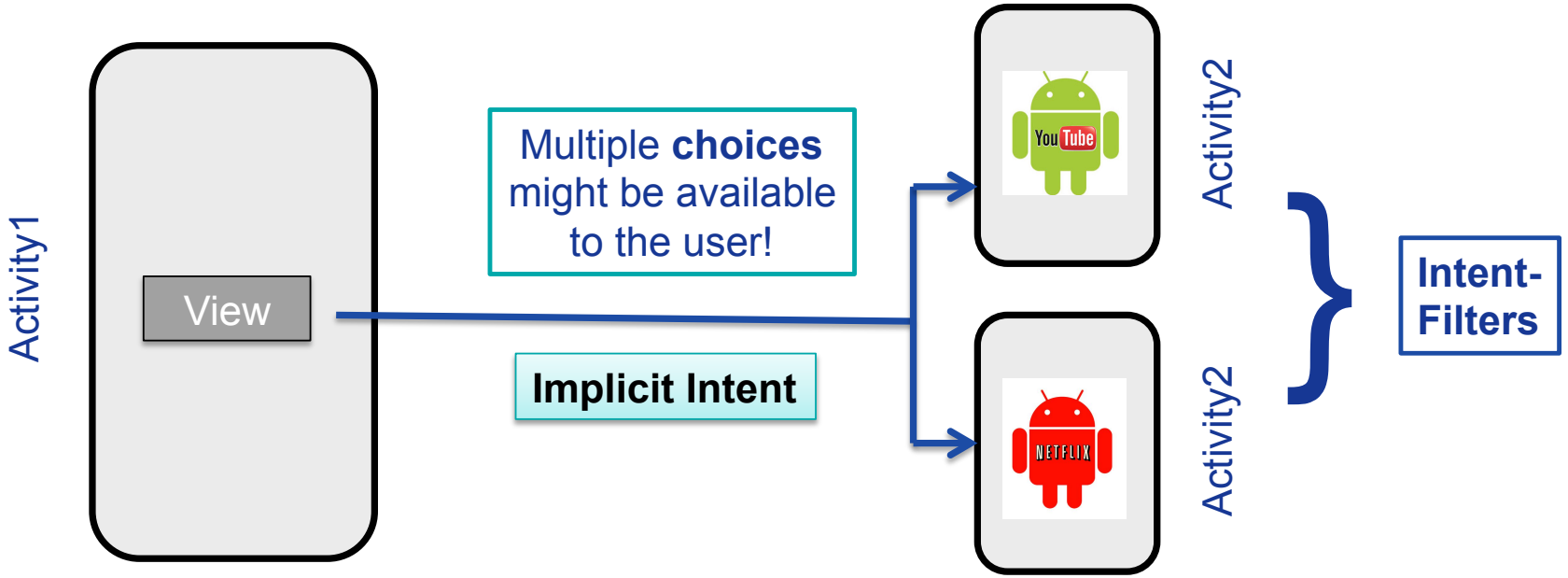
Android Components: Intents

- **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)*.



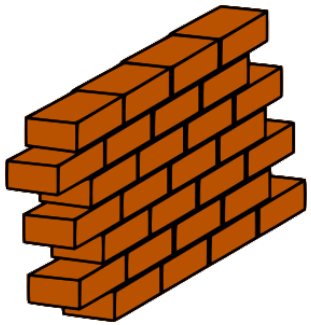
Android Components: Intents

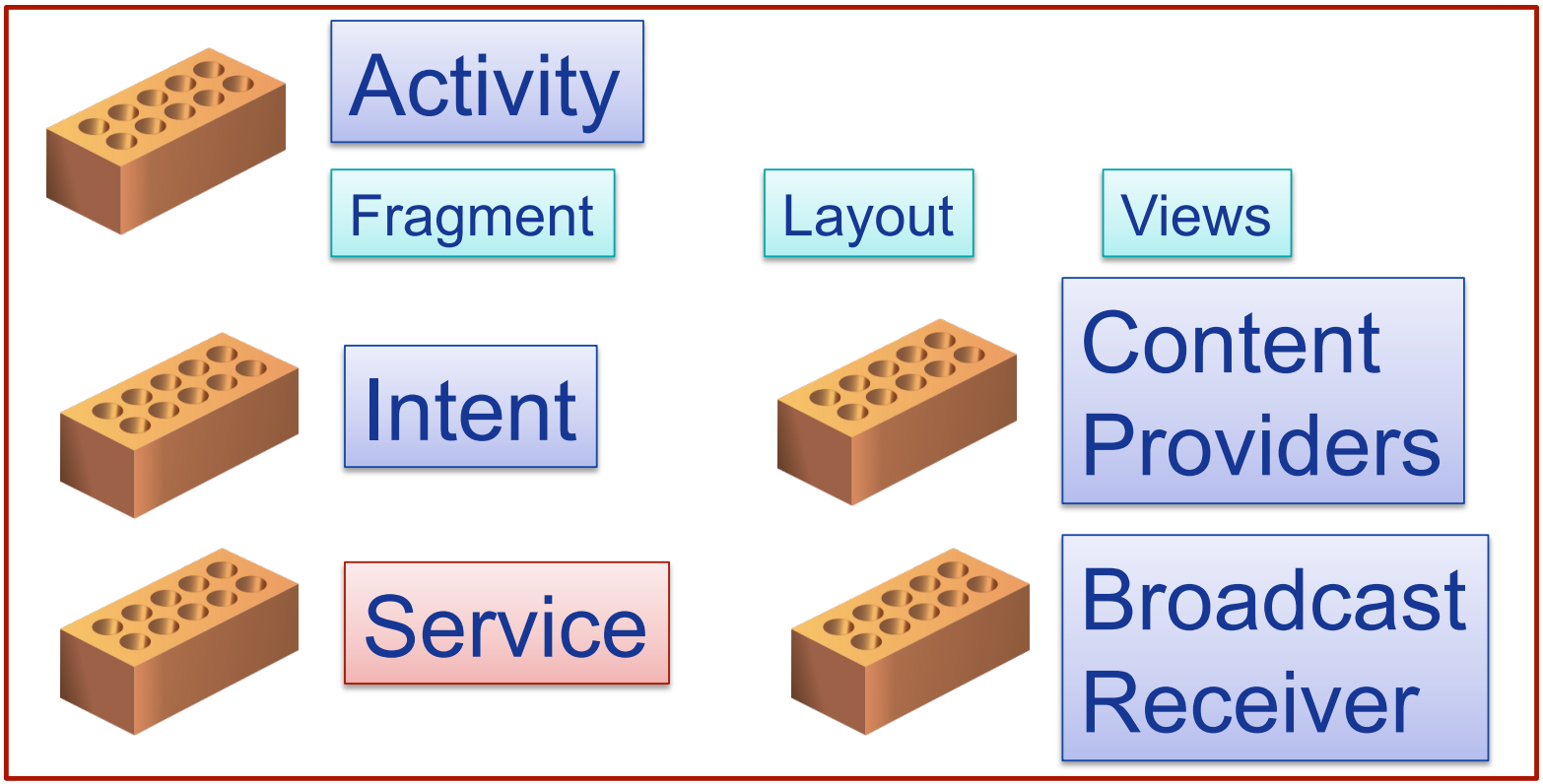
- **Intents**: asynchronous **messages** to activate core Android components (e.g. Activities).
- **Implicit** Intent → The component *(e.g. Activity1)* specifies the type of the intent *(e.g. "View a video")*.



Android Applications Design

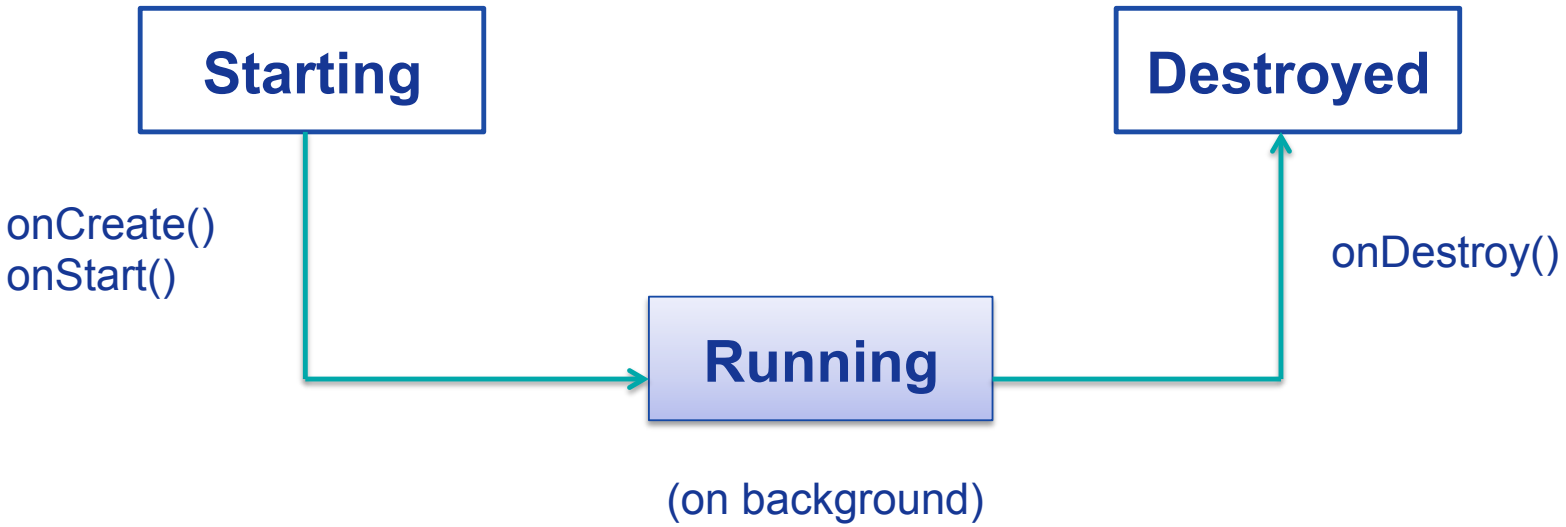
- Developing an Android Application means using in a proper way the **Android basic components** …





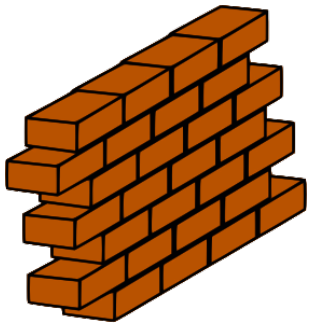
Android Components: Services

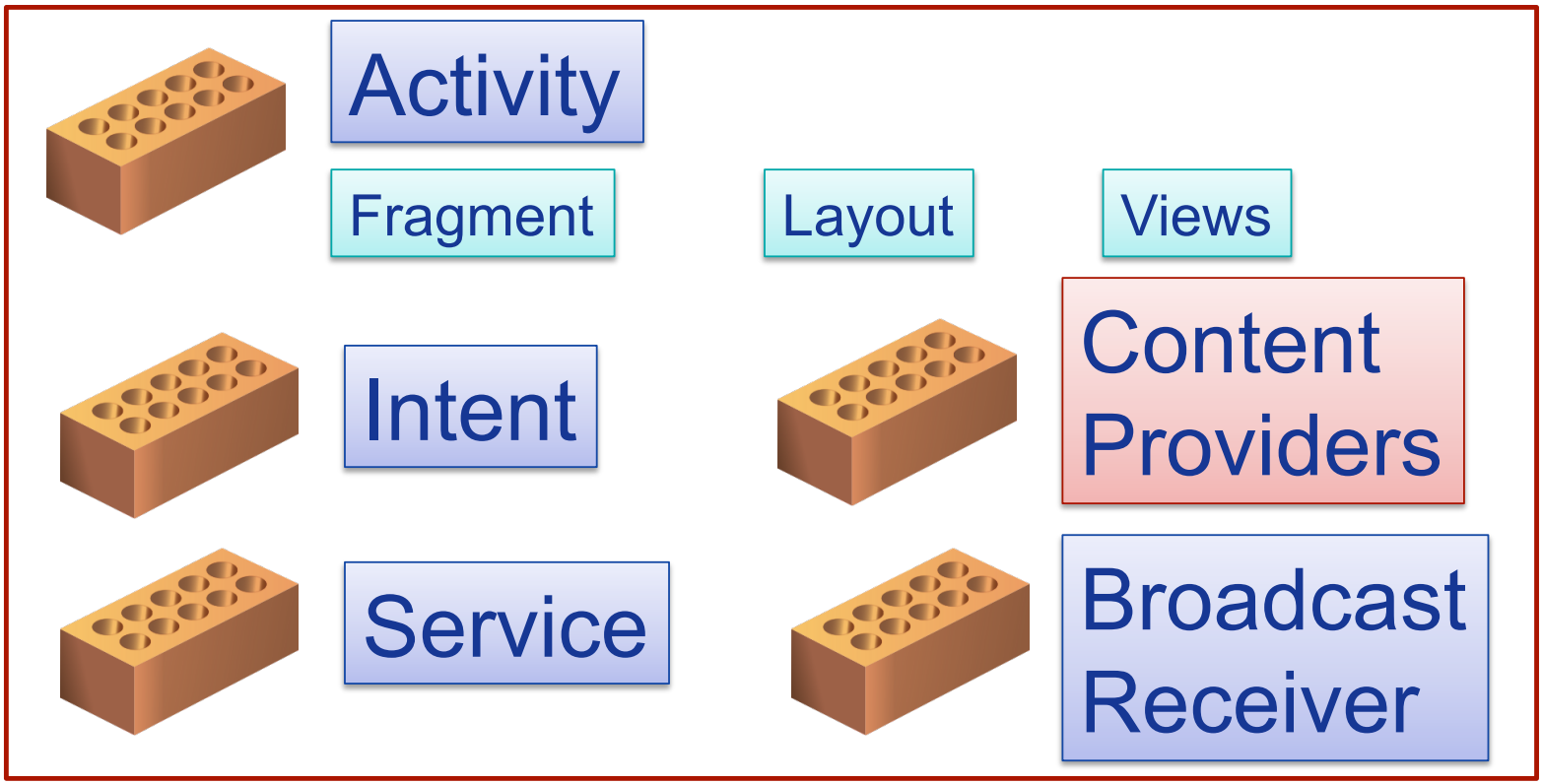
- **Services**: like Activities, but run in **background** and do not provide an user interface.
- Used for **non-interactive** tasks (e.g. networking).
- Service life-time composed of 3 states:



Android Applications Design

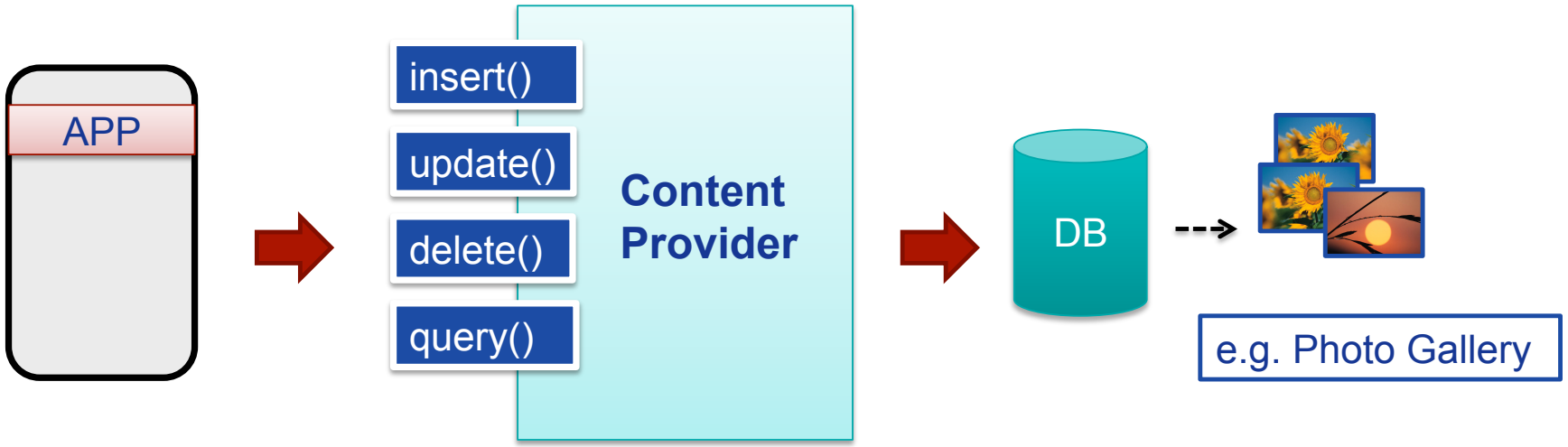
- Developing an Android Application means using in a proper way the **Android basic components** …





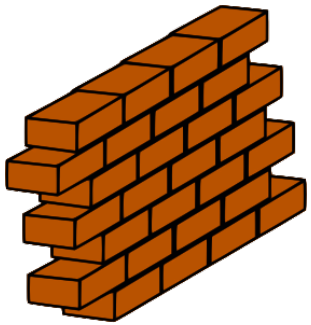
Android Components: Content Providers

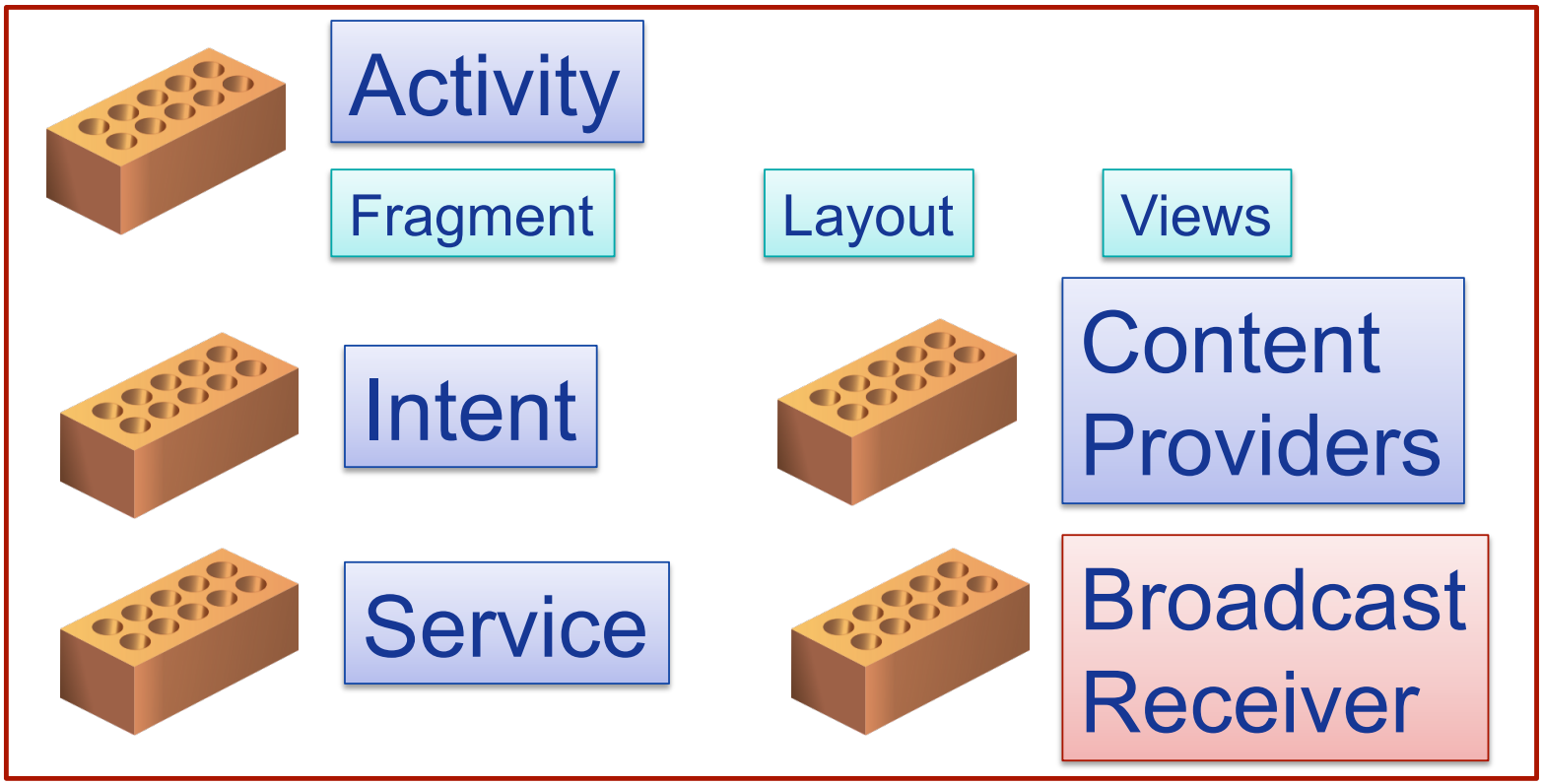
- Each Android **application** has its own **private** set of data (managed through *files* or through *SQLite* database).
- **Content Providers**: Standard **interface** to *access and share data among different applications*.



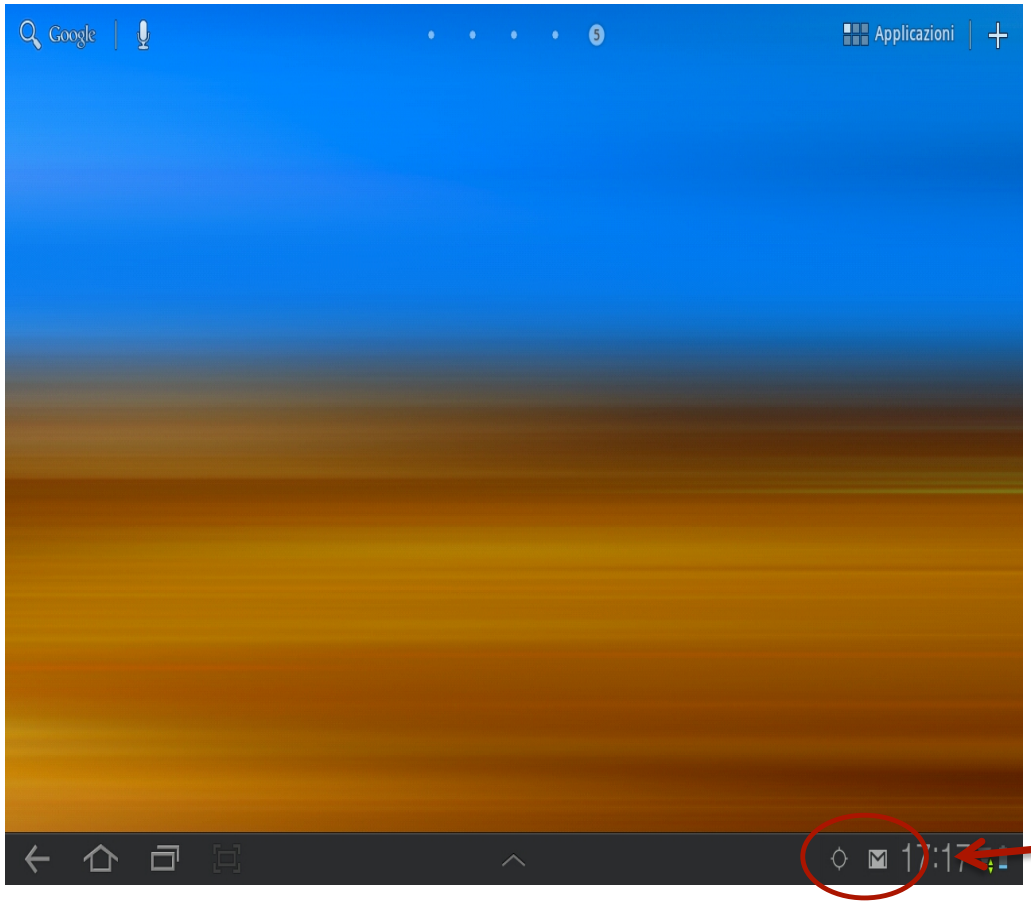
Android Applications Design

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



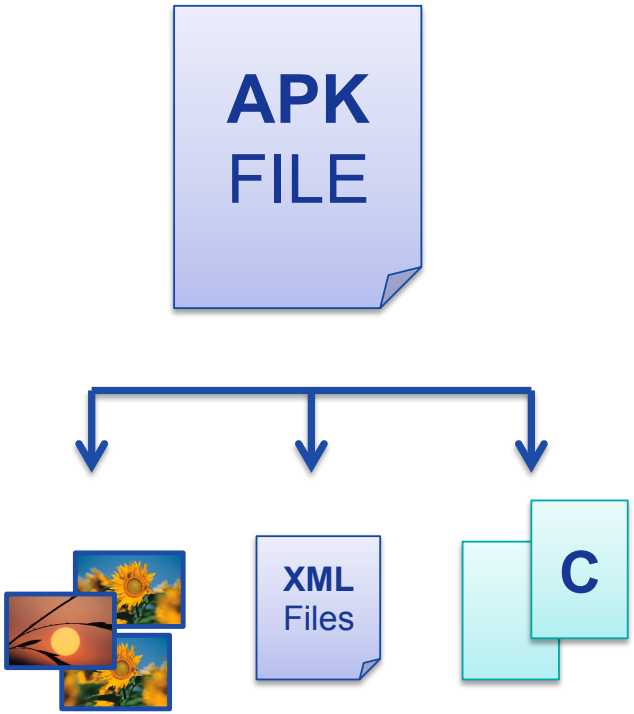


Android Components: Broadcast Receivers



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

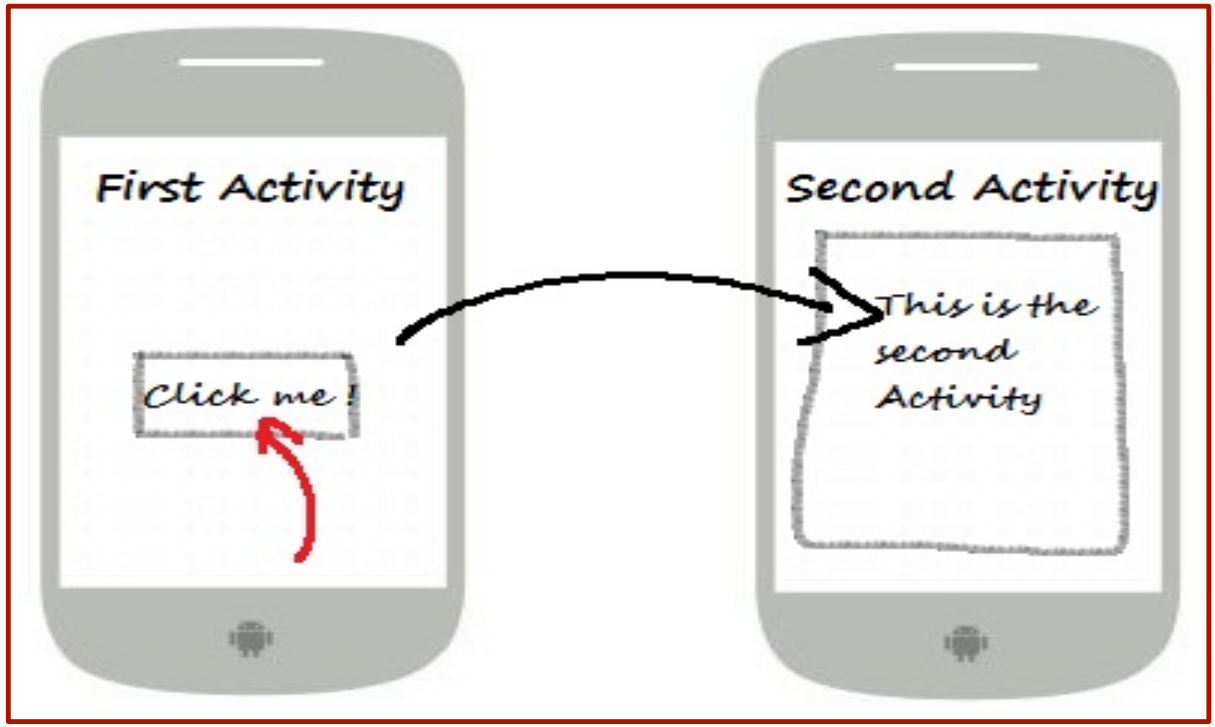
Android Application Distribution



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

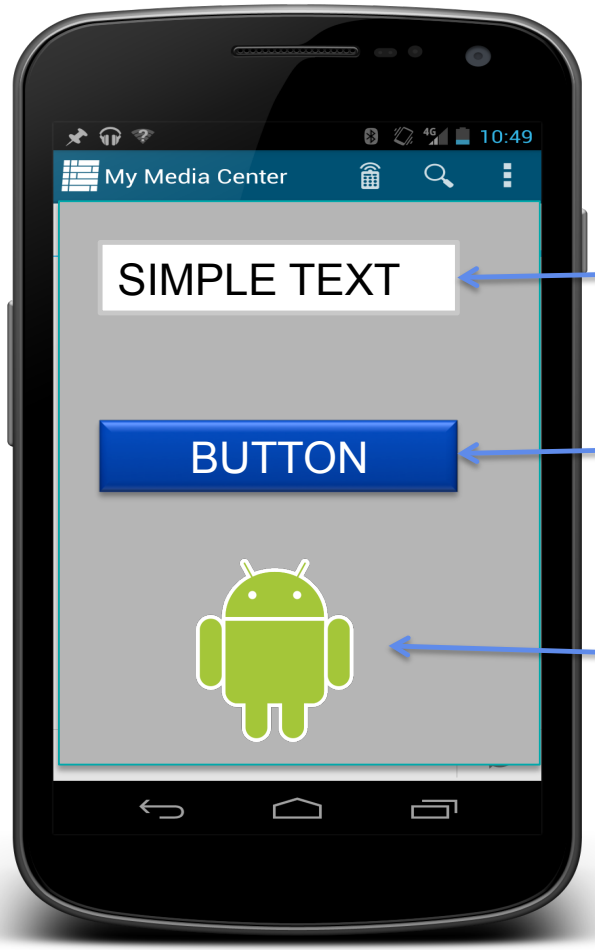
Android Activity

- An **Activity** is a *component* that provides a **screen** with which users can interact in order to do something …



Android Activity

Each **Activity** is composed of graphical components, named **Views**.



TextView — Plain text, users cannot interact with it

Button — Users can click on it

ImageView — Display a static image

Android Activity

Views can be placed on the **Activity**, based on a **Layout**

Vertical

Horizontal

Grid

Android Activity

- **Active** (or running)
  - Foreground of the screen
- **Paused**
  - Lost focus but still visible
  - Can be killed by the system in extreme situations
- **Stopped**
  - Completely obscured by another activity
  - Killed if memory is needed somewhere else

Android Activity

In order to **create** an **Activity** on your app:

- **Extend** the `Activity` class
- **Implement** the `onCreate()` and `onPause()` methods

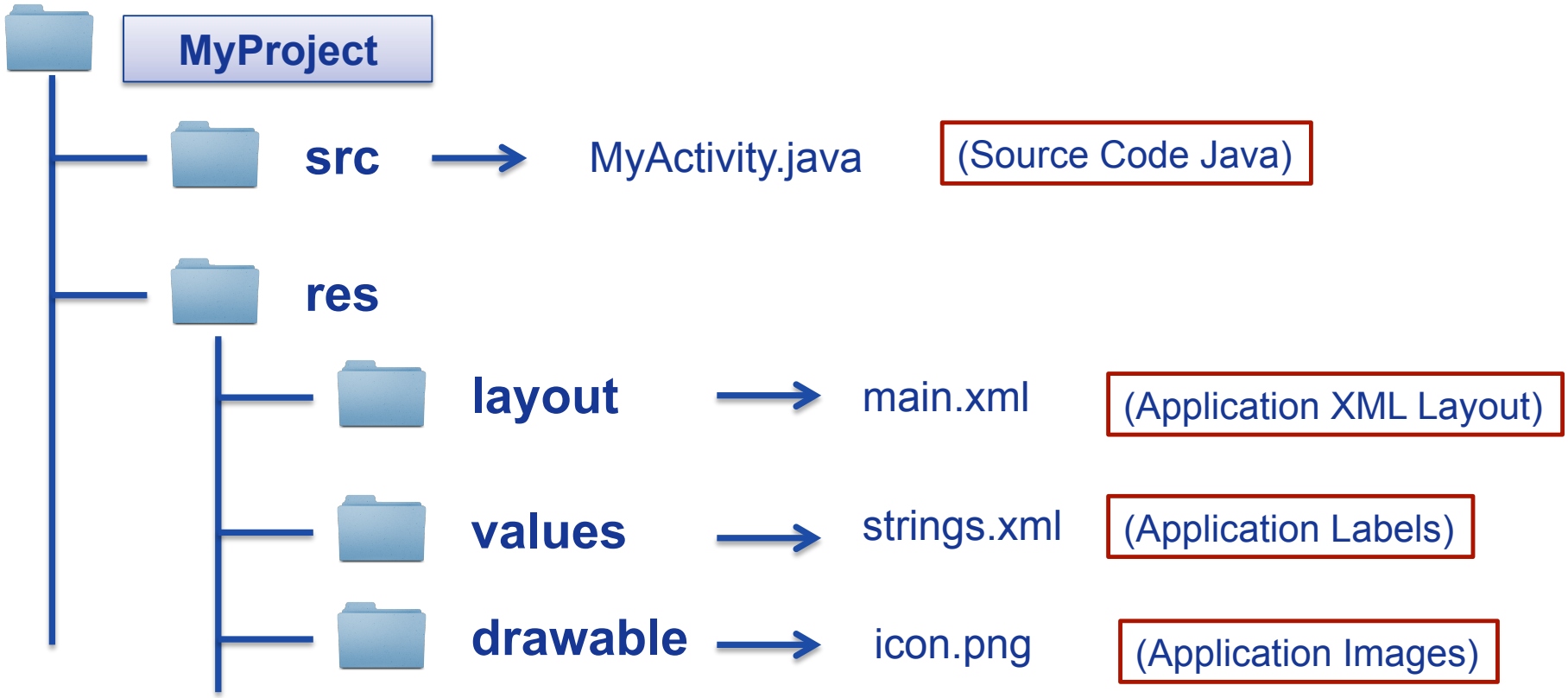
```
public class myActivity extends Activity {
…
  protected void onCreate() { …
  }
  protected void onPause() { …
  }
}
```

CALLED AUTOMATICALLY WHEN THE ACTIVITY **IS CREATED**

CALLED AUTOMATICALLY WHEN THE USER IS **LEAVING THE ACTIVITY**

Application Resources

- Resources are defined in the **res**/ folder of the project.



Application Resources Definition

DEF. Resources are everything that is not Java code (including: XML layout files, language packs, images, audio/video files, etc)

Utilization of **Resources**… why?

- **Separate** what the application does from how the application looks like!

- **Provide** alternative resources to support specific device configurations (e.g. different language)

Application Resources Definition

Create a resource of type: **String**

STRINGS.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

     <string name="app_name"> My First Android App</string>

     <string name="label" > Hello world!   </string>

</resources>
```

Application Resources Definition

Create a resource of type: **Integer**

STRINGS.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

     <integer name="val1">  1  </integer>

     <integer name="val2" > 2 </integer>

</resources>
```

Application Resources Definition

Create a resource of type: **Color**

STRINGS.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <color name="blue">  #ff3c2eff  </color>
    <color name="red">   #ffff0d21  </color>
    <color name="green">  #ff388138  </color>

</resources>
```

Application Resources Definition

Create a resource of type: **Layout** → Add a **Button**

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <TextView
            android:text=”Hello world"
            android:id="@+id/textViewName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content”
            android:background= … />
</resources>
```

Application Resources Definition

Create a resource of type: **Layout** → Add a **Button**

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <Button
            android:text=”PressHere"
            android:id="@+id/buttonName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content”
            android:background= … />
</resources>
```

Application Resources Definition

Create a resource of type: **Layout** → Add an **EditText**

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <EditText
            android:id="@+id/editTextName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content”
            android:background= … />
</resources>
```

Application Resources Definition

Create a resource of type: **Layout** → Add a **CheckBox** (ON/OFF Button)

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <CheckBox
            android:text=”Click Here"
            android:id="@+id/checkBoxName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content”
            android:background= … />
</resources>
```

Application Resources Definition

Create a resource of type: **Layout** → Add a **ToggleButton**

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <ToggleButton
            android:text=”Click Here"
            android:id="@+id/toggleButtonName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content”
            android:background= … />
</resources>
```

Application Resources Definition

Create a resource of type: **Layout** → Add a **RadioButton**

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <RadioButton
            android:text=”Select this option"
            android:id="@+id/radioButtonName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content”
            android:background= … />
</resources>
```

Application Resources Definition

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<RadioGroup
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical”>
        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/buttonRadio1"
            android:text="Option 1"
            android:checked="true” />
       <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/buttonRadio2"
            android:text="Option 2” />
</RadioGroup>
</resources>
```

Mutually **exclusive** Radio buttons

Application Resources Definition

Create a resource of type: **Layout** → Add an **ImageView**

ACTIVITY_MAIN.XML

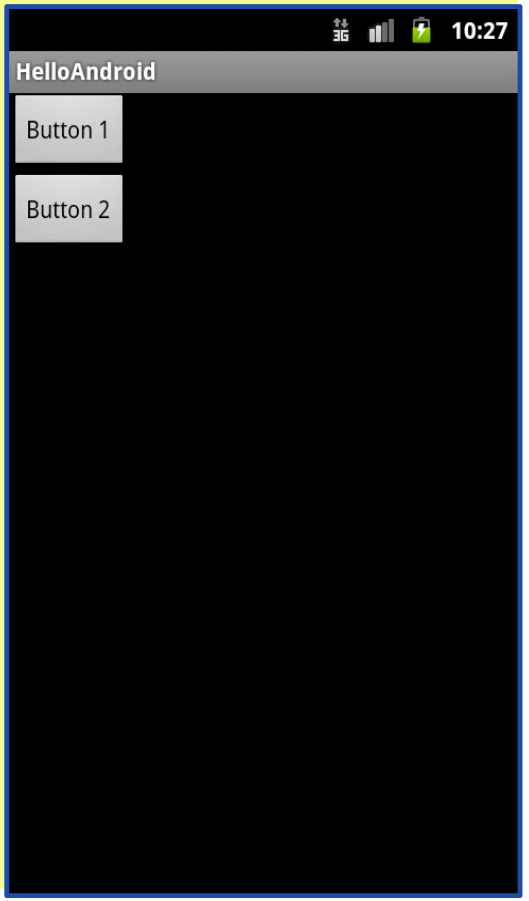
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content”
            android:src= … />
</resources>
```

Application Resources Definition

ACTIVITY_MAIN.XML

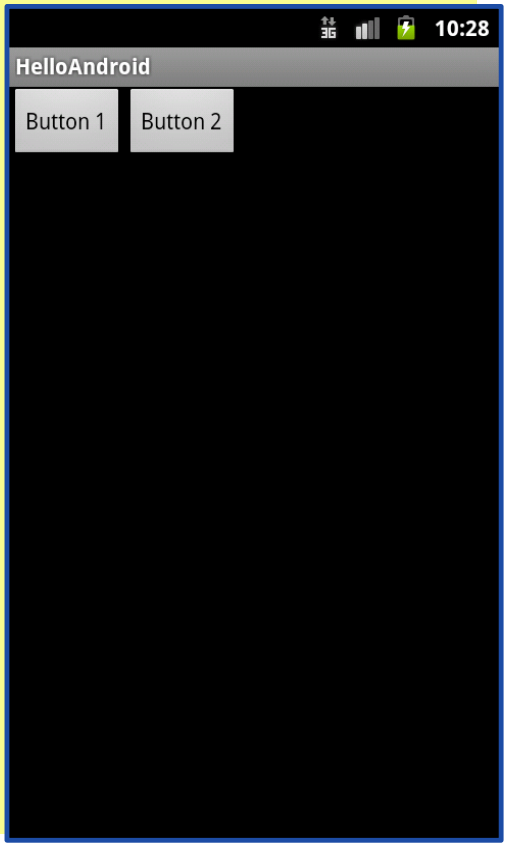
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString2" />
</LinearLayout>
```



Application Resources Definition

ACTIVITY_MAIN.XML

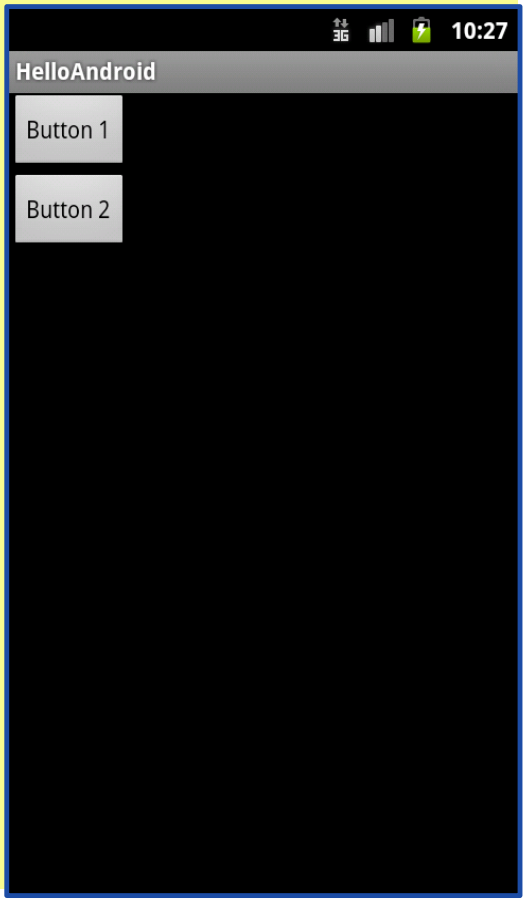
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString2" />
</LinearLayout>
```



Application Resources Definition

ACTIVITY_MAIN.XML

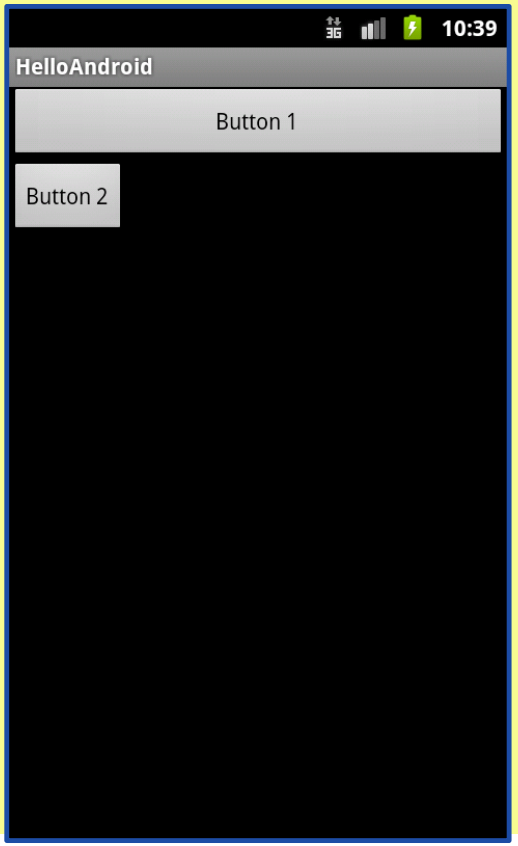
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString2" />
</LinearLayout>
```



Application Resources Definition

ACTIVITY_MAIN.XML

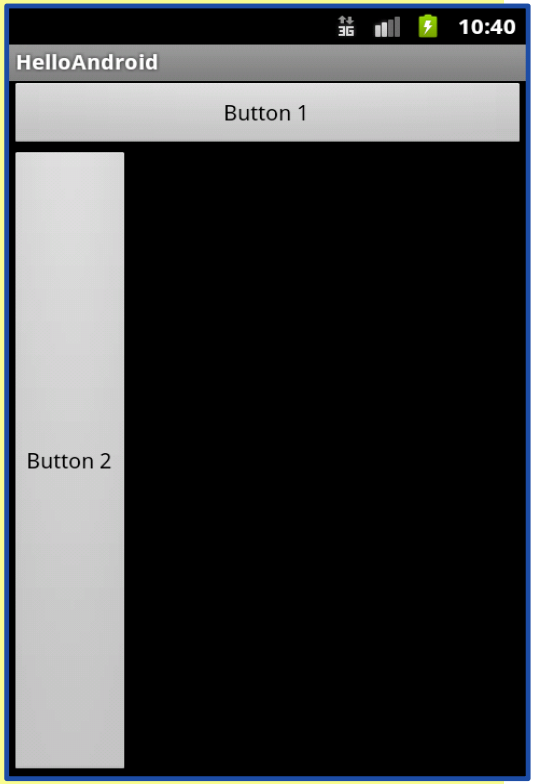
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation=”vertical" >
   <Button
      android:id="@+id/button1"
      android:layout_width=”match_parent"
      android:layout_height="wrap_content"
      android:text="@string/buttonString1" />
   <Button
      android:id="@+id/button2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/buttonString2" />
</LinearLayout>
```



Application Resources Definition

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
   <Button
      android:id="@+id/button1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="@string/buttonString1" />
   <Button
      android:id="@+id/button2"
      android:layout_width="wrap_content"
      android:layout_height="match_parent"
      android:text="@string/buttonString2" />
</LinearLayout>
```



Access to Application Resources

Q. How to access resources defined in another XML files?

@<resource_type>/<resource_name>

- <**resource_type**> is the the name of the resource type
- <**resource_name**> is either the resource filename without the extension or the android:name attribute value in the XML element.

Access to Application Resources

STRING.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name=”labelButton”> Submit </string>
    <string name=”labelText”> Hello world! </string>
</resources>
```

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <Textview android:id=“@+id/label1” android:text=“@string/
labelText” />
    <Button android:id=“@+id/button1” android:text=“@string/
labelButton”/>
  </resources>
```

Access to Application Resources

STRINGS.XML

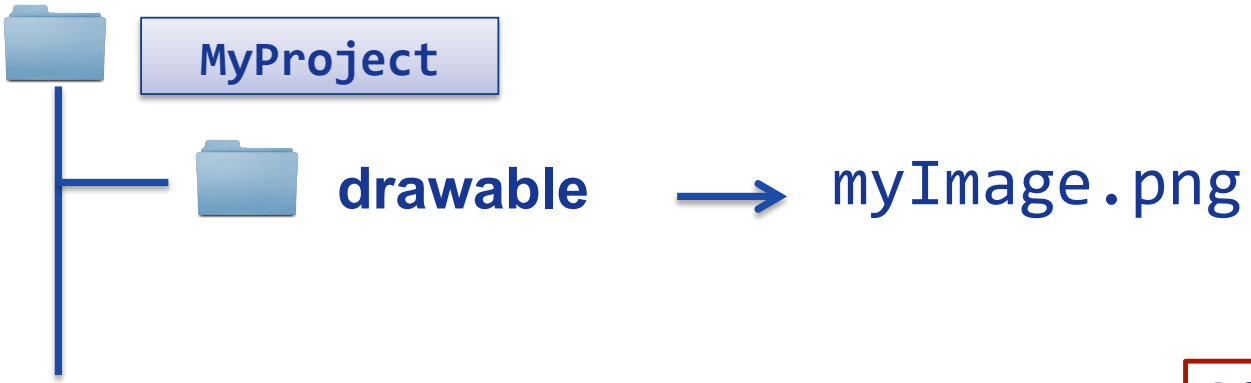
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name=”blue”> #ff3c2eff </string>
    <color name=”red”>#ffff0d21</string>
</resources>
```

ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <Textview android:id=“@+id/label1” android:text=“@string/
labelText” android:textColor=“@color/blue” />

    <Button android:id=“@+id/button1” android:text=“@string/
labelButton” android:textColor=“@color/red” />
  </resources>
```

Access to Application Resources



ACTIVITY_MAIN.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <ImageView android:id="@+id/img1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:src="@drawable/myImage" />
  </resources>
```

Resources Alternatives

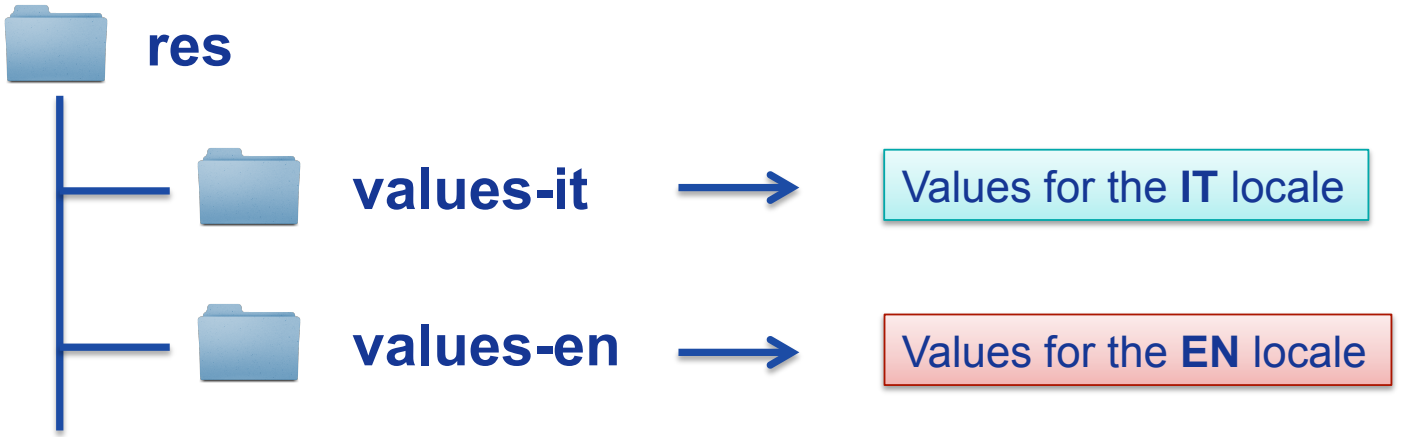
- Android applications might provide **alternative resources** to support specific device configurations (e.g. different languages).
- At runtime, Android **detects** the current device configuration and **loads** the appropriate resources for the application.
- To specify configuration-specific alternatives:

  1. Create a new directory in **res**/ named in the form *<resources_name>-<config_qualifier>*
  2. Save the respective alternative resources in this new directory

Resources Alternatives

Name of the folder: **<resources_name>-<config_qualifier>**.

- <*resources_name*> is the directory name of the corresponding default resources
- <*qualifier*> is a name that specifies an individual configuration for which these resources are to be used.



Interactive/Dynamic App Behaviours

Till now, we have worked on the (static) **User Interface** of the Mobile Application …

How to make our application more **interactive/dynamic**?

1. Manage **events** generated by Views
2. Access **resources** by Java code
3. Add **animations** to Views

Interactive/Dynamic App Behaviours

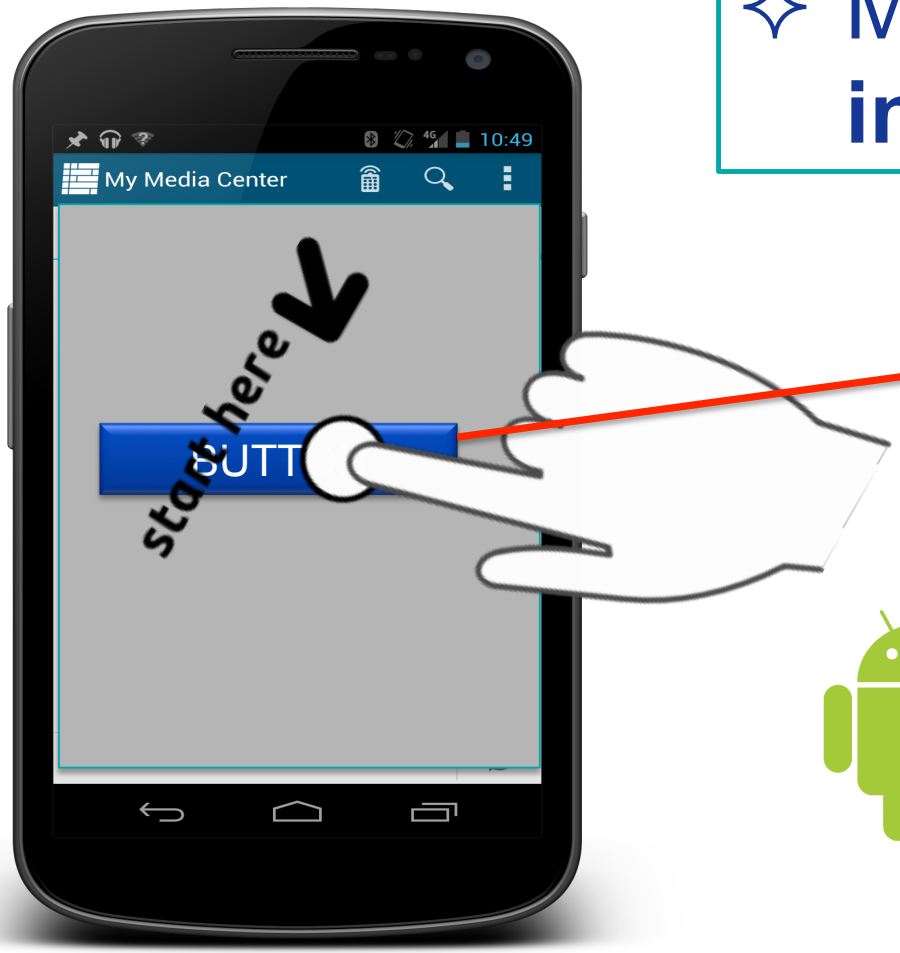
Till now, we have worked on the (static) **User Interface** of the Mobile Application …

How to make our application more **interactive/dynamic**?

1. Manage **events** generated by Views
2. Access **resources** by Java code
3. Add **animations** to Views

Handling Events from Views objects

- Most of the **Views** are **interactive** objects …



"What do you want to do after click?"

YOUR **APP**

Handling Events from Views objects

SOLUTION 1: Works only for **click** events on some **Views**

```
<Button
      android:text="@string/textButton"
      android:id="@+id/idButton"
      android:onClick=”doSomething”
/>
```

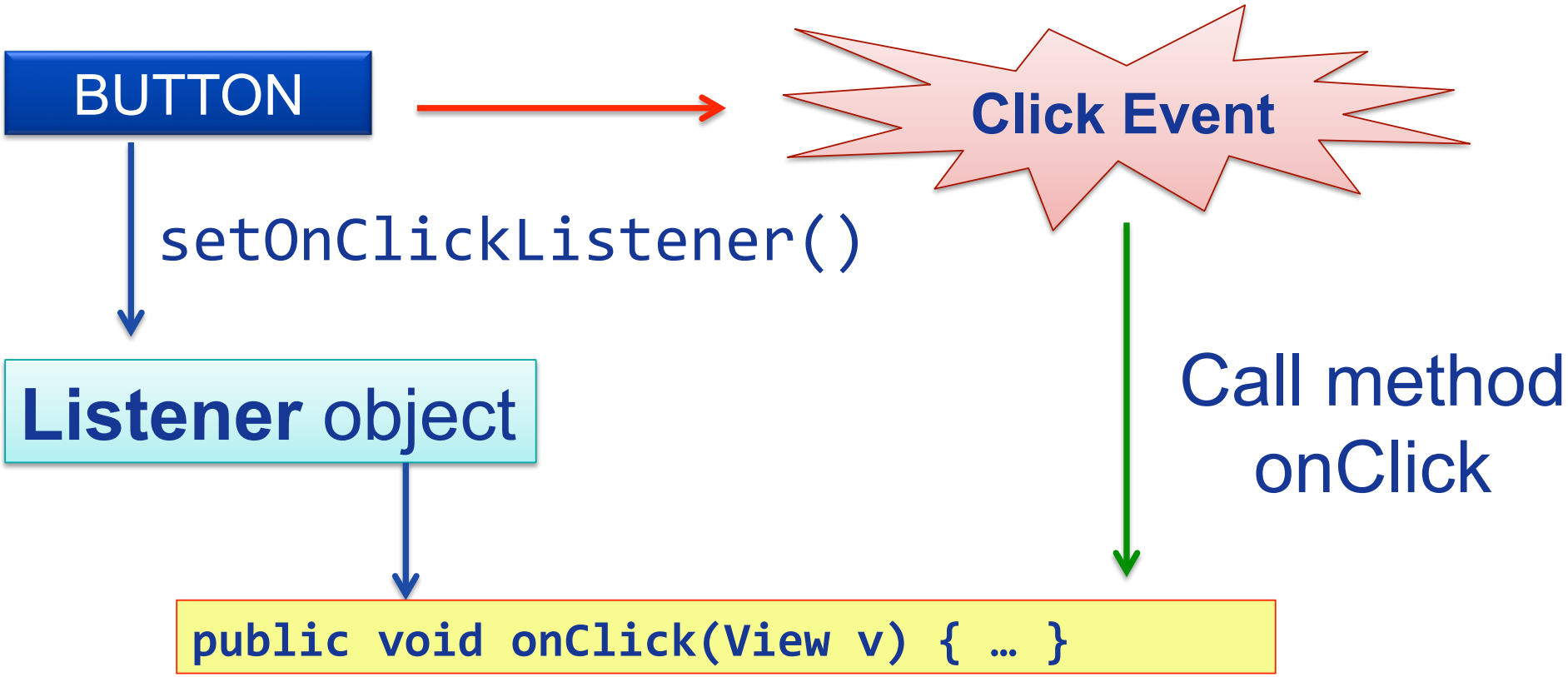
ACTIVITY_MAIN.XML

Java code

```
public void doSomething(View w) {
..
}
```

Handling Events from Views objects

SOLUTION 2: Works on all Views and for all kinds of events



Handling Events from Views objects

SOLUTION 2: Works on all Views and for all kinds of events

```
public class ExampleActivity extends Activity
implements OnClickListener {
 …
  button.setOnClickListener(this);
   …
  public void onClick(View v) { }
}
```

Access to Application Resources

```
public class ExampleActivity extends Activity
implements OnClickListener {
  protected void onCreate {
        …
        Button button=(Button) findViewById
(R.id.buttonName);
        button.setOnClickListener(this);
  }

}
```

Access to Application Resources

```
public class ExampleActivity extends Activity
implements OnClickListener {
   …

   public void onClick(View v) {
      Toast t=Toast.makeToast(this,”Bottone
premuto”, Toast.LENGTH_SHORT);
      t.show();
   }
}
```

Interactive/Dynamic App Behaviours

Till now, we have worked on the (static) **User Interface** of the Mobile Application …

How to make our application more **interactive/dynamic**?

1. Manage **events** generated by Views
2. Access **resources** by Java code
3. Add **animations** to Views

Access to Application Resources: Java

- Resource can be accessed in the **Java** code through the **R class**, that works as a **glue** between the world of java and the world of resources.
- **Automatically generated** file, no need to modify it.

```
public final class R {
    public static final class string {
         public static final int hello=0x7f040001;
          public static final int label1=0x7f040005;
    }
}
```

Access to Application Resources: Java

STEP0: ***Declare*** *resources in res/*

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

   <string name="hello"> Hello </string>
   <string name="label1"> Label  </string>
</resources>
```

XML-Based, Declarative Approach

STEP1: ***Compile*** *the project*

STEP2: **Access** *resources through R class*

```
public final class R {

    public static final class string {
        public static final int hello=0x7f040001;
         public static final int label1=0x7f040005;
    }

}
```

Java Code, Programmatic Approach

Access to Application Resources: Java

Q. How to access resources from Java code?

R.<*resource_type*>.<*resource_name*>

- <**resource_type**> is the the name of the resource type
- <**resource_name**> is either the resource filename without the extension or the android:name attribute value in the XML element.

Access to Application Resources: Java

ACCESSING **RESOURCES** OF TYPE: STRING or COLOR

getResources().getString(ID) → Return a String with a given ID
getResources().getColor(ID) → Return a Color with a given ID

```
// Get a string resource from the string.xml file
hello=getResources().getString(R.string.hello);

// Get a color resource from the string.xml file
color=getResources().getColor(R.color. red);
```

Access to Application Resources: Java

ACCESSING **RESOURCES** OF TYPE: LAYOUT

findViewById(id) → Return a **View** with a given ID

```
/ / Set the text on a TextView object

TextView msgTextView = (TextView)
     findViewById(R.id.label1);

msgTextView.setText(getResources().getString(R.string
.stringText));
```

Access to Application Resources: Java

ACCESSING **RESOURCES** OF TYPE: LAYOUT

findViewById(id) → Return a **View** with a given ID

```
/ / Set the image on a ImageView object

ImageView myImgView = (ImageView)
     findViewById(R.id.imageview1);

myImgView.setDrawable(getResources().getDrawable(R.dr
awable.stringText2));
```

Interactive/Dynamic App Behaviours

Till now, we have worked on the (static) **User Interface** of the Mobile Application …

How to make our application more **interactive/dynamic**?

1. Manage **events** generated by Views
2. Access **resources** by Java code
3. Add **animations** to Views

Interactive/Dynamic App Behaviours

Animations allow to add dynamic effects to the User Interface of our application … HOW?

1. Create an **`anim/`** folder in the `res/` folder
2. Create an **`animation.xml`** file
3. Define the type of **effect** you want to produce (in XML)
4. **Apply** the effect to the View (in XML or JAVA)

Interactive/Dynamic App Behaviours

Alpha animation: adjust transparency

MYANIMATION.XML

```
<set
    xmlns:android="http://schemas.android.com/apk/res/
android">
    <alpha
        android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="1500"
         />
</set>
```

Interactive/Dynamic App Behaviours

Apply the animation to a **TextView**

ANIMATION.XML

```
protected void onCreate() {
  ….
    ….
     TextView text = (TextView)findViewById(R.id.mytext);
     Animation alphaAnim = AnimationUtils.loadAnimation(this,
R.anim.myanimation);
     text.startAnimation(alphaAnim);
   }
```

Interactive/Dynamic App Behaviours

Translate animation: perform traslation

MYANIMATION2.XML

```
<set xmlns:android="http://schemas.android.com/apk/res/
android">
<translate
        android:fromXDelta=”0"
        android:toXDelta=”200"
        android:fromYDelta=”0"
        android:toYDelta=”600”
        android:duration="1500” />
</set>
```

Interactive/Dynamic App Behaviours

Apply the animation to a **TextView**

ANIMATION.XML

```
protected void onCreate() {
  ….
    ….
     TextView text = (TextView)findViewById(R.id.mytext);
     Animation alphaAnim = AnimationUtils.loadAnimation(this,
R.anim.myanimation2);
     text.startAnimation(alphaAnim);
   }
```

Interactive/Dynamic App Behaviours

Rotate animation: perform **rotation**

MYANIMATION3.XML

```
<set xmlns:android="http://schemas.android.com/apk/res/
android">
<rotate
        android:fromDegrees=”0"
        android:toDegrees=”180"
        android:pivotX=”50%"
        android:pivotY=”50%"
       android:duration="1500” />
</set>
```

Interactive/Dynamic App Behaviours

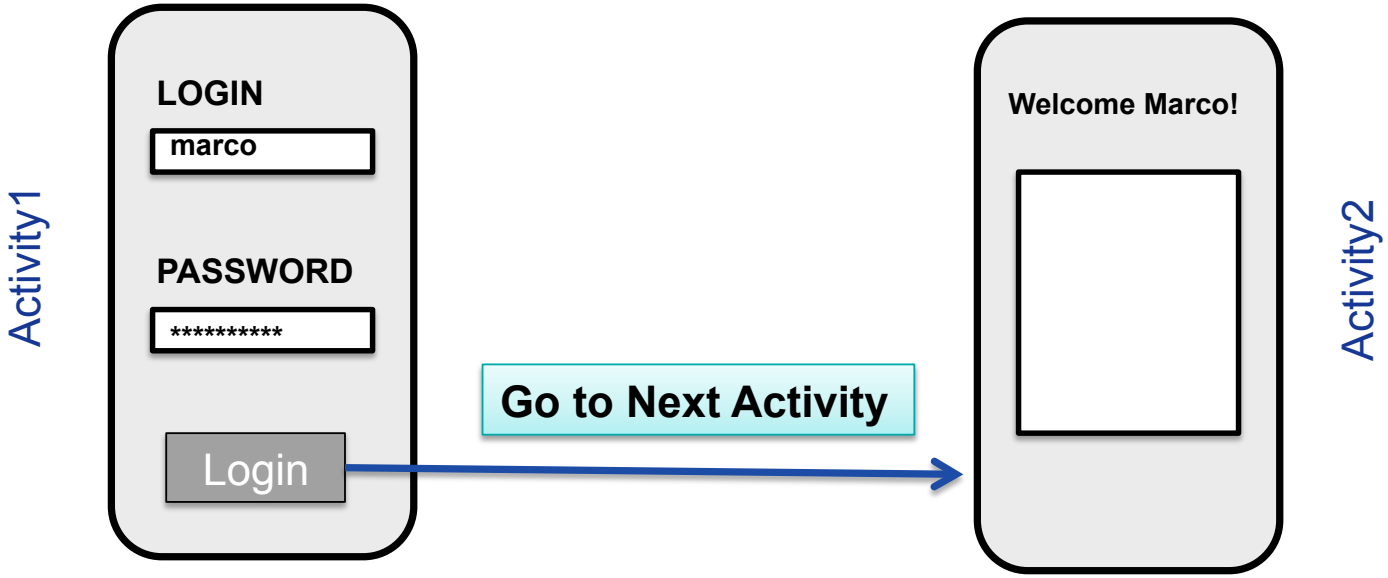
Scale animation: perform scaling

MYANIMATION4.XML

```
<set xmlns:android="http://schemas.android.com/apk/res/
android">
<scale
        android:fromXScale="1.4"
        android:toXScale="0.0"
        android:fromYScale="0.6"
        android:toYScale="0.0"
         android:duration="1500” />
</set>
```

Handling multiple Activities: Intents

In most cases, an Android Application is composed of **multiple Activities, not just one …**



Handling multiple Activities: Intents

- **Each Activity** has its own:
  - **Java** implementation → `MyActivity.java`
  - XML **Layout** file → `activity_main.xml`
  - **Lifecycle** with different states: ACTIVE, PAUSED, STOPPED
  - **XML Tag** in `AndroidManifest.xml`

```
<application>
        <activity android:name=".MyActivity" />
</application>
```

Handling multiple Activities: Intents

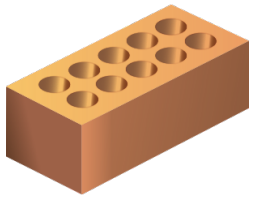
Each activity has its own **Java** code and **layout** file.

```
public class FirstActivity extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_first);
}
```

```
public class SecondActivity extends Activity {
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_two);
}
```

Handling multiple Activities: Intents

How to pass **control** from an Activity to another?



Through Android **Intent**!

Android Intent → similar to the concept of **Message**

Explicit Intent

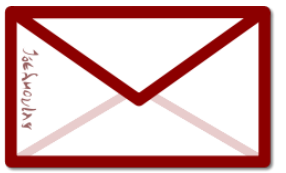
(Define the **name** of next Activity to be lanched)

Implicit Intent

(Define only the **action** to be performed)

Intent Definition

We can think to an **Intent** object as a **message** containing a bundle of information.



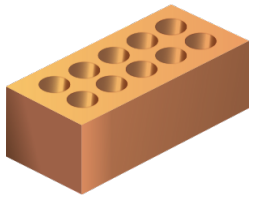
Structure of an **Intent**

- **Component Name**
- **Action Name**
- **Data**
- **Category**
- Extra
- Flags

✧ Intent is sent from current Activity to a receiver Activity which is then **activated** and **executed**.

Handling multiple Activities: Intents

How to pass **control** from an Activity to another?



Through Android **Intent**!

Android Intent → similar to the concept of **Message**

Explicit Intent

(Define the **name** of next Activity to be lanched)

Implicit Intent

(Define only the **action** to be performed)

Handling multiple Activities: Intents

1. Build a new `Intent` message
2. Specify the Activity who will **receive** the Intent
3. Fire the Intent through the `startActivity()`

NAME OF THE ACTIVITY TO START

```
Intent intent=new Intent(this, SecondActivity.class);

startActivity(intent);
```

Handling multiple Activities: Intents

(OPTIONAL) Insert parameters to be sent to the called Activity in the the **Extra** field of the Intent.

```
intent.putExtra("KEY", VALUE);
```

Set an argument named "MyValue" and equal to 5.

```
Intent intent=new Intent(this, SecondActivity.class);
intent.putExtra("myValue",5);
startActivity(intent);
```

Handling multiple Activities: Intents

(OPTIONAL) From the called Activity, **retrieve the parameters** inserted from the calling Activity

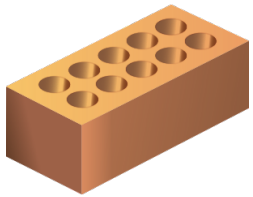
```
intent.getExtras)().getTYPE(“KEY”);
```

Get an argument of type int with key equal to “myValue”

```
intent.getExtras().getInt(“myValue”);
intent.getExtras().getString(“myString”);
intent.getExtras().getBoolean(“myBoolean”);
```

Handling multiple Activities: Intents

How to pass **control** from an Activity to another?



Through Android **Intent**!

Android Intent → similar to the concept of **Message**

Explicit Intent

(Define the **name** of next Activity to be lanched)

Implicit Intent

(Define only the **action** to be performed)

Handling multiple Activities: Intents

1. Build a new `Intent` message
2. Specify only the **Action** you want to perform
3. Fire the Intent through the `startActivity()`

ACTION NAME

```
Intent intent=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

startActivity(intent);
```

Handling multiple Activities: Intents

Special actions (http://developer.android.com/reference/android/content/Intent.html)

Action Name	Description
ACTION_IMAGE_CAPTURE	Open the camera and receive a photo
ACTION_VIDEO_CAPTURE	Open the camera and receive a video
ACTION_DIAL	Open the phone app and dial a phone number
ACTION_SENDTO	Send an email (email data contained in the extra)
ACTION_SETTINGS	Open the system setting
ACTION_WIRELESS_SETTINGS	Open the system setting of the wireless interfaces
ACTION_DISPLAY_SETTINGS	Open the system setting of the display

Intent types: Explicit Intents

Generic actions (http://developer.android.com/reference/android/content/Intent.html)

Action Name	Description
ACTION_EDIT	Display data to edit
ACTION_MAIN	Start as a main entry point, does not expect to receive data.
ACTION_PICK	Pick an item from the data, returning what was selected.
ACTION_VIEW	Display the data to the user
ACTION_SEARCH	Perform a search

- **Action Defined** by the programmer

  it.example.projectpackage.FILL_DATA (package prefix + name action)

Handling multiple Activities: Intents

Some actions require **data in input** to be executed.

Use method **`setData(URI)`** to define the data input of an Implicit Intent

```
Intent intent=new Intent(Intent.ACTION_DIAL);

intent.setData(Uri.parse("tel:0123456789"));

startActivity(intent);
```

Handling multiple Activities: Intents

Some actions require **data in input** to be executed.

Use method **`setData(URI)`** to define the data input of an Implicit Intent

```
Intent intent=new Intent(Intent.ACTION_VIEW);

intent.setData(Uri.parse("content://contacts/people/1"));

startActivity(intent);
```

Handling multiple Activities: Intents

Some actions require **data in input** to be executed.

Use method **`setData(URI)`** to define the data input of an Implicit Intent

```
Intent intent=new Intent(Intent.ACTION_VIEW);

intent.setData(Uri.parse(”http://www.cs.unibo.it"));

startActivity(intent);
```

Intent types: Explicit Intents

In an Intent, the **Data** is specified by a **name** and a **type**

NAME: Uniform Resource Identifier **(URI)**

```
scheme://host:port/path
```

EXAMPLEs

```
tel:003-232-134-126
content://contacts/people/1
http://www.cs.unibo.it
```

Intent Components

In some cases, you would like to specify only the **type of the target Activity**, given by the `Category` field

Use method **`addCategory(String)`** to define the receiver

```
Intent intent=new Intent();
Intent.setAction(Intent.ACTION_MAIN);
intent.addCategory(Intent.CATEGORY_HOME);

startActivity(intent);
```

Intent Components

Category → String describing the kind of component (Activity) that should handle the Intent.

Category Name	Description
CATEGORY_HOME	The activity displays the HOME screen.
CATEGORY_LAUNCHER	The activity is listed in the top-level application launcher, and can be displayed.
CATEGORY_PREFERENCE	The activity is a preference panel.
CATEGORY_BROWSABLE	The activity can be invoked by the browser to display data referenced by a link.

Handling multiple Activities: Intents

QUESTION: How can Android know which application to call after an Implicit Intent is fired?

ANSWER: Each application declares the **Intent** is able to handle in the `AndroidManifest.xml` file

If an Intent with Action name ACTION_ECHO is invoked, the Activity is lanched

```
<intent-filter>
        <action android:name="ACTION_ECHO" />
</intent-filter>
```

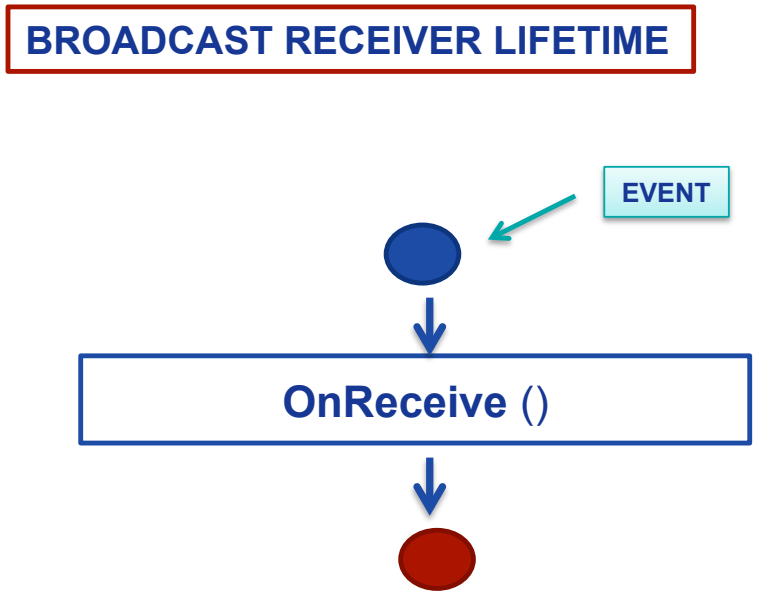
Handling events: Broadcast Receivers

A **Broadcast Receiver** is a component that is activated only when specific events occur (i.e. SMS arrival, phone call, etc).

- **Registration** of the Broadcast Receiver to the event …
  1. Event → **Intent**
  2. Register the event through **XML** code
- Extend the **`BroadcastReceiver`** class
- **Handling** of the event through the **`onReceive()`** method.

Handling events: Broadcast Receivers

A **Broadcast Receiver** is a component that is activated only when specific events occur (i.e. SMS arrival, phone call, etc).



- Single-state component …
- **onReceive**() is invoked when the registered event occurs
- After handling the event, the Broadcast Receiver is **destroyed**.

Handling events: Broadcast Receivers

Registration of the Broadcast Receiver to the event

XML Code: → modify the **AndroidManifest**.xml

```
<application>
    <receiver android:name="SMSReceiver">
        <intent-filter>
            <action
android:name="android.provider.Telephony.SMS_RECEIVED" />
        </intent-filter>
    </receiver>
</application>
```

Handling events: Broadcast Receivers

Implement the **action** to be performed when the Intent is handled

```
public class SMSReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent
intent) {
        Toast tt=Toast.makeText(context, "Ciao",
Toast.LENGTH_SHORT);
        tt.show();
    }
}
```

Managing the Data of an Application

How to store the **data** produced by a Mobile Application?

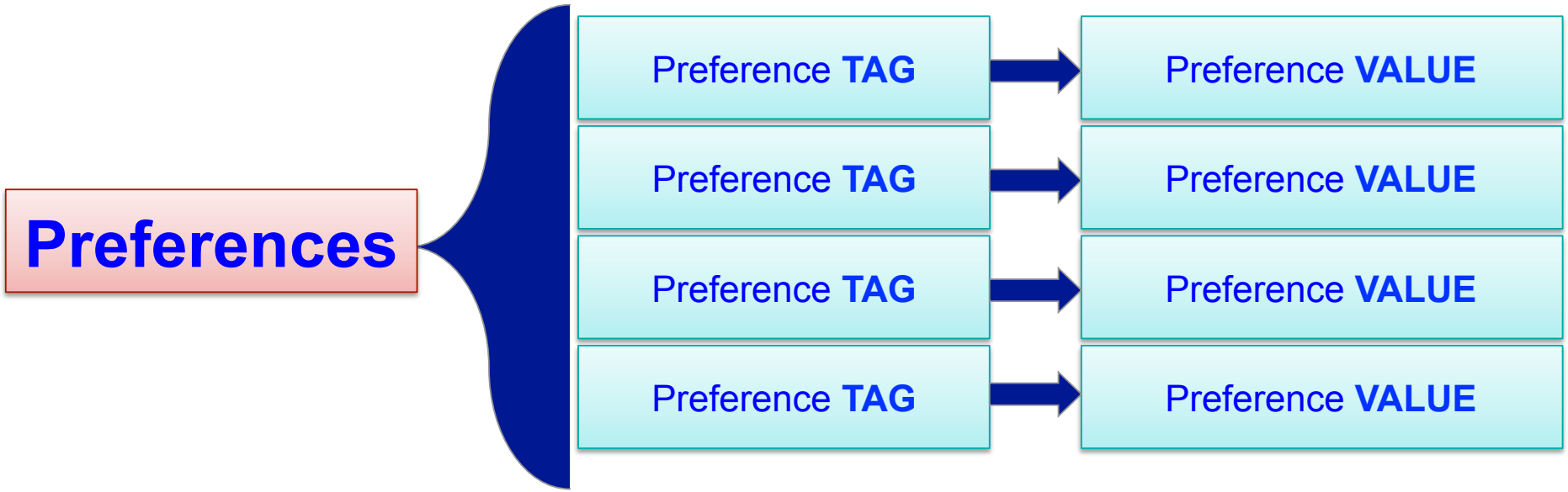
Three major alternatives:

1. Through the **Shared Preferences** (PUBLIC-PRIVATE)
2. Through the **File Systems** (PUBLIC-PRIVATE)
3. Through a **Database SQLite**

Managing the Data of an Application

Shared Preferences are a common way to edit/store the configuration parameters of a mobile application

PREFERENCES → Sets of **key-values** couples



Managing the Data of an Application

1. Get a SharedPreference object

```
getSharedPreferences(String, Context.MODE_WORLD_READABLE);
getSharedPreferences(String, Context.MODE_WORLD_WRITABLE);
getSharedPreferences(String, Context.MODE_PRIVATE);
```

2. Get a value from a key through **getTYPE(KEY,def)**

```
SharedPreferences pref=getSharedPreferences("MY_TAG", Context.
                                Context.MODE_PRIVATE);
pref.getString("KEY_STRING","Hello");
```

Default value whether the key does not exist

Managing the Data of an Application

3. Edit Preferences through putTYPE(KEY) method

```
SharedPreferences pref=getSharedPreferences(“MY_TAG”, Context.
                              Context.MODE_PRIVATE);

SharedPreferences.Editor editor = pref.edit();
editor.putString("mydata", “Hello world”);
editor.commit();
```

Perform **commit** to write the value on the SharedPreferences

Managing the Data of an Application

How to store the **data** produced by a Mobile Application?

Three major alternatives:

1. Through the **Shared Preferences** (PUBLIC-PRIVATE)
2. Through the **File Systems** (PUBLIC-PRIVATE)
3. Through a **Database SQLite**

Managing the Data of an Application

Data can be saved on the **file-system of the device**

Two file **storage areas**:

INTERNAL STORAGE

- ✧ Always available
- ✧ Accessible only by the current app
- ✧ Deleted when the user uninstall the app

BEST if you need PRIVACY

EXTERNAL STORAGE

- ✧ Can be available or not
- ✧ Accessible by the current app or shared
- ✧ Not necessarily deleted with the app

BEST if you dont' need RESTRICTIONS

Managing the Data of an Application

In Android, files can be **read/write** by using the same methods used by java.IO package.

CREATE THE FILE

```
File myfile = new File(getFilesDir(), filename);
```

WRITE a STRING (TEXT FILE)

```
BufferedWriter writer = new BufferedWriter(new
                                    FileWriter(myfile));
String string="Hello world";
writer.write(string);
writer.close();
```

Managing the Data of an Application

In Android, files can be **read/write** by using the same methods used by java.IO package.

READ a STRING (TEXT FILE)

```
File myfile = new File(getFilesDir(), filename);
```

```
BufferedWriter reader= new BufferedWriter(new
                                        FileReader(myfile));
String line;
while ((line = reader.readLine()) != null) {
       // process the line.
}
```

Managing the Data of an Application

Q. How to display the **data** on the UI, when we are not sure about the **number of items** to be displayed?

A. Through a **ListView** = list of scrollable items.

```
<ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id=@+id/mylist
/>
```

Managing the Data of an Application

Adapter → adapts a set of data to be displayed on a ListView

ArrayAdapter → we need to display a set of **String**

STEP 1: Define the data to be displayed, and the View for each element

```
String mylist[]={“Ciao”, “Ciao”, “Ciao2”};
ArrayAdapter<String> adapter = new ArrayAdapter<String>
                               (this,
                                android.R.layout.simple_list_item_1,
                                mylist);
```

VIEW to be USED FOR EACH ITEM

LIST OF STRING

Managing the Data of an Application

Adapter → adapts a set of data to be displayed on a ListView

ArrayAdapter → we need to display a set of **String**

STEP 2: Apply the ArrayAdapter to the container (ListView)

```
ListView lv=(ListView) findViewById(R.id.mylist);

lv.setAdapter(adapter);
```

SET THE **NUMBER OF ITEMS** TO BE DISPLAYED ON THE LISTVIEW
AND THE **VIEW** TO BE USED FOR EACH ITEM