The fundamental components

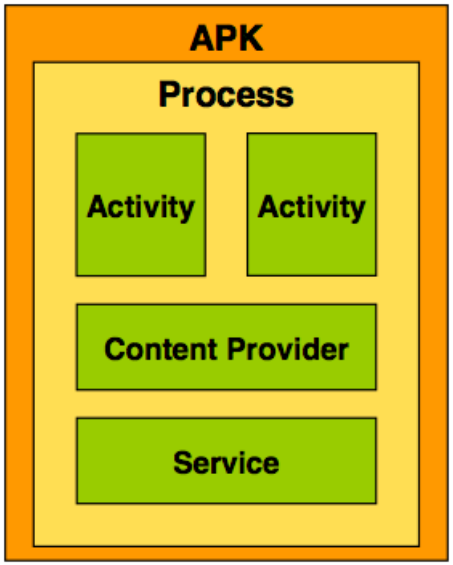
Marco Ronchetti
Università degli Studi di Trento

The fundamental components

- Activity
  - an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.
- Fragment (since 3.0)
  - a behavior or a portion of user interface in an Activity
- View
  - equivalent to Swing Component
- Service
  - an application component that can perform long-running operations in the background and does not provide a user interface
- Intent
  - a passive data structure holding an abstract description of an operation to be performed. It activates an activity or a service. It can also be (as often in the case of broadcasts) a description of something that has happened and is being announced.
- Broadcast receiver
  - component that enables an application to receive intents that are broadcast by the system or by other applications.
- Content Provider
  - component that manages access to a structured set of data.

Peeking into an application

Packaging: APK File (Android Package)
Collection of components



- Components share a set of resources
  - Preferences, Database, File space
- Components share a Linux process
  - By default, one process per APK
- APKs are isolated
  - Communication via Intents or AIDL (Android Interface Definition Language)
- Every component has a managed lifecycle

ONE APPLICATION, ONE PROCESS, MANY ACTIVITIES

Slide borrowed from Dominik Gruntz (and modified)

Activity

Not exactly what you might imagine…

Wordnet definitions:

- something that people do or cause to happen
- a process occurring in living organisms
- a process existing in or produced by nature (rather than by the intent of human beings)

Activity

From Wikipedia, the free encyclopedia

Activity may mean:

- Action (philosophy), in general
- The Aristotelian concept of energeia, Latinized as *actus*
- Physical exercise
- Activity (UML), a major task in Unified Modeling Language
- Activity diagram, a diagram representing activities in Unified Modeling Language
- *Activity*, an alternative name for the game charades
- *Activity*, the rate of catalytic activity, such as enzyme activity (enzyme assay), in physical chemistry and enzymology
- Activity (chemistry), the effective concentration of a solute for the purposes of mass action
- Activity (project management)
- Activity (radioactivity), radioactive decay#Radioactive decay rates, the number of radioactive decays per second
- Activity (software engineering)
- Activity (soil mechanics)
- *HMS Activity (D94)*, an aircraft carrier of the Royal Navy
- in military parlance, a military agency or unit (e.g. Intelligence Support Activity)
- Activity Theory , social constructivism (learning theory), Education

Activities

A rather misleading term… it's not a "computer activity", like a process.
It's rather an environment where a "user activity" is performed

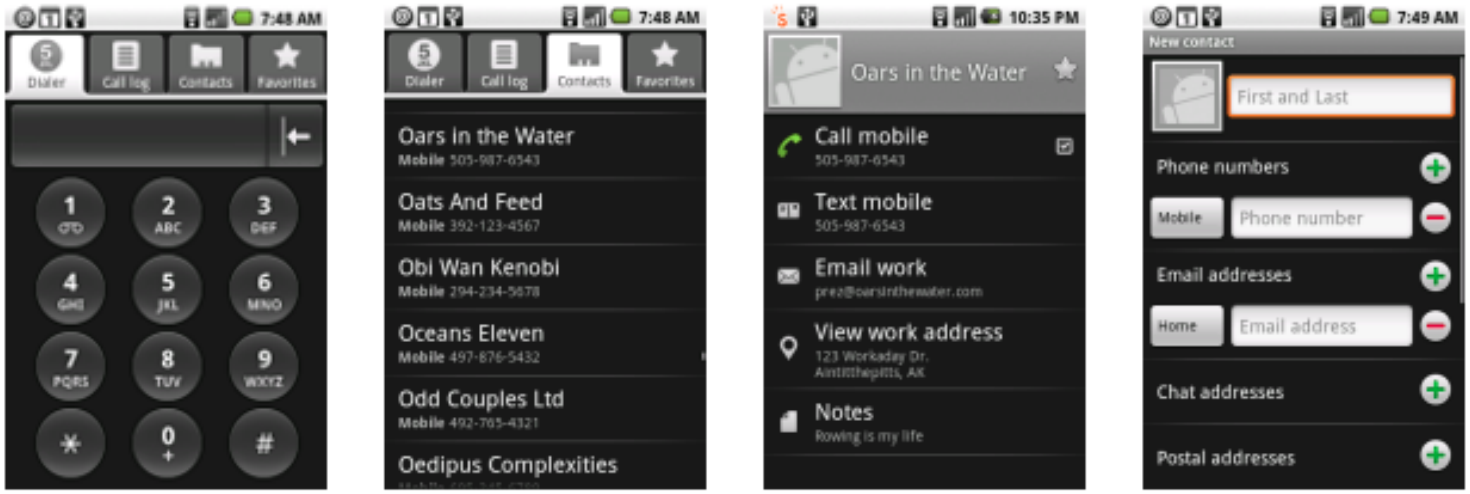
- "single" UI screens
- One visible at the time (Well. Almost…)
- One active at the time
- Stacked like a deck of cards

Activity

An application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.

Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows, or be embedded in another activity (activityGroup).

Activities of the dialer application

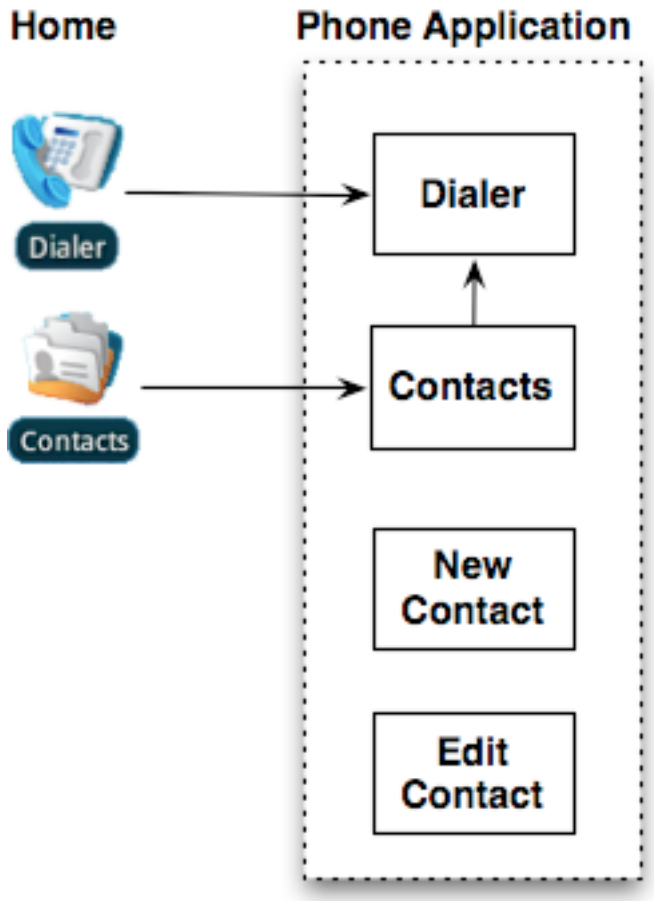


Multiple entry-point for an app

Typically, one activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time.

BUT

An application can have multiple entry points



Activity

Each activity can start another activity in order to perform different actions.

Each time a new activity starts, the previous activity is stopped.

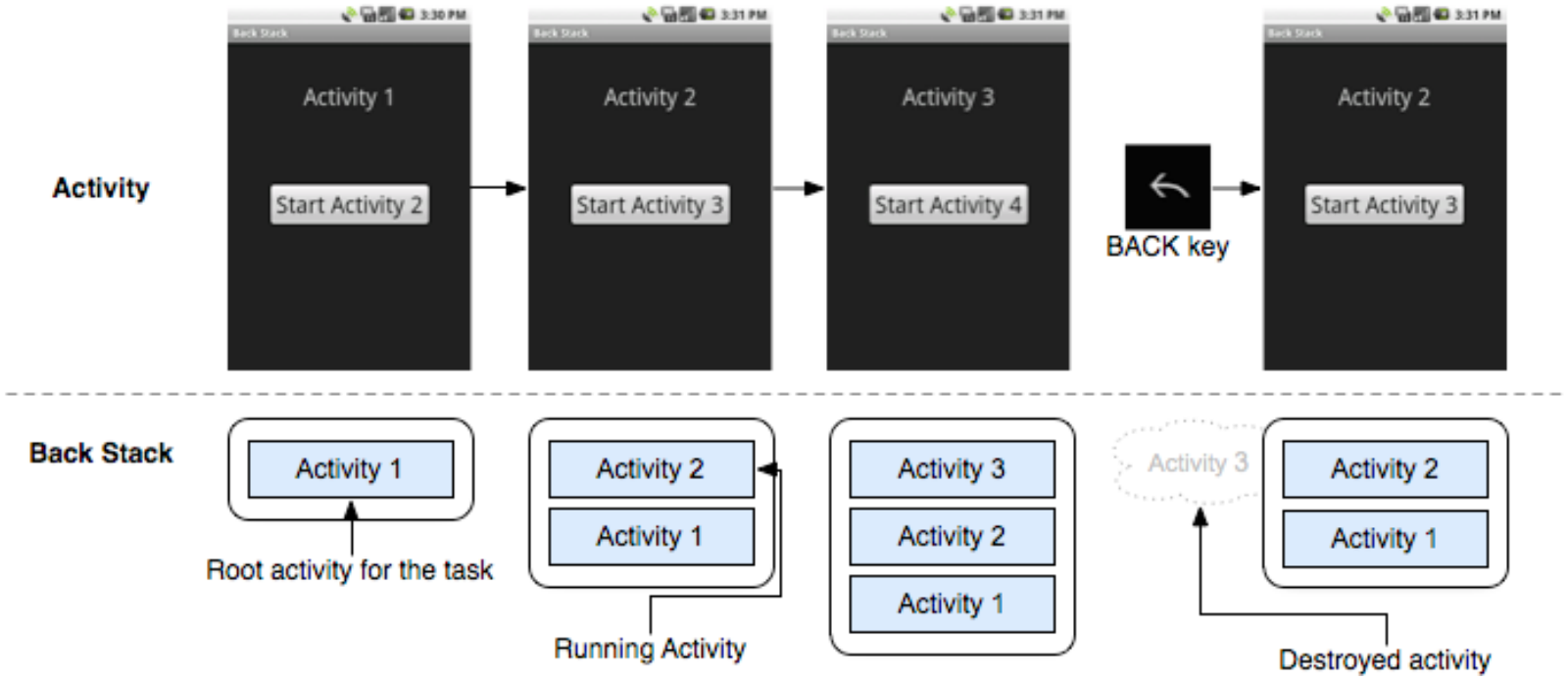
The system preserves the activity in a LIFO stack (the "activity stack" or "back stack").

The new activity it is pushed on top of the back stack and takes user focus.

When the user is done with the current activity and presses the BACK button, the current activity is popped from the stack (and destroyed) and the previous activity resumes.

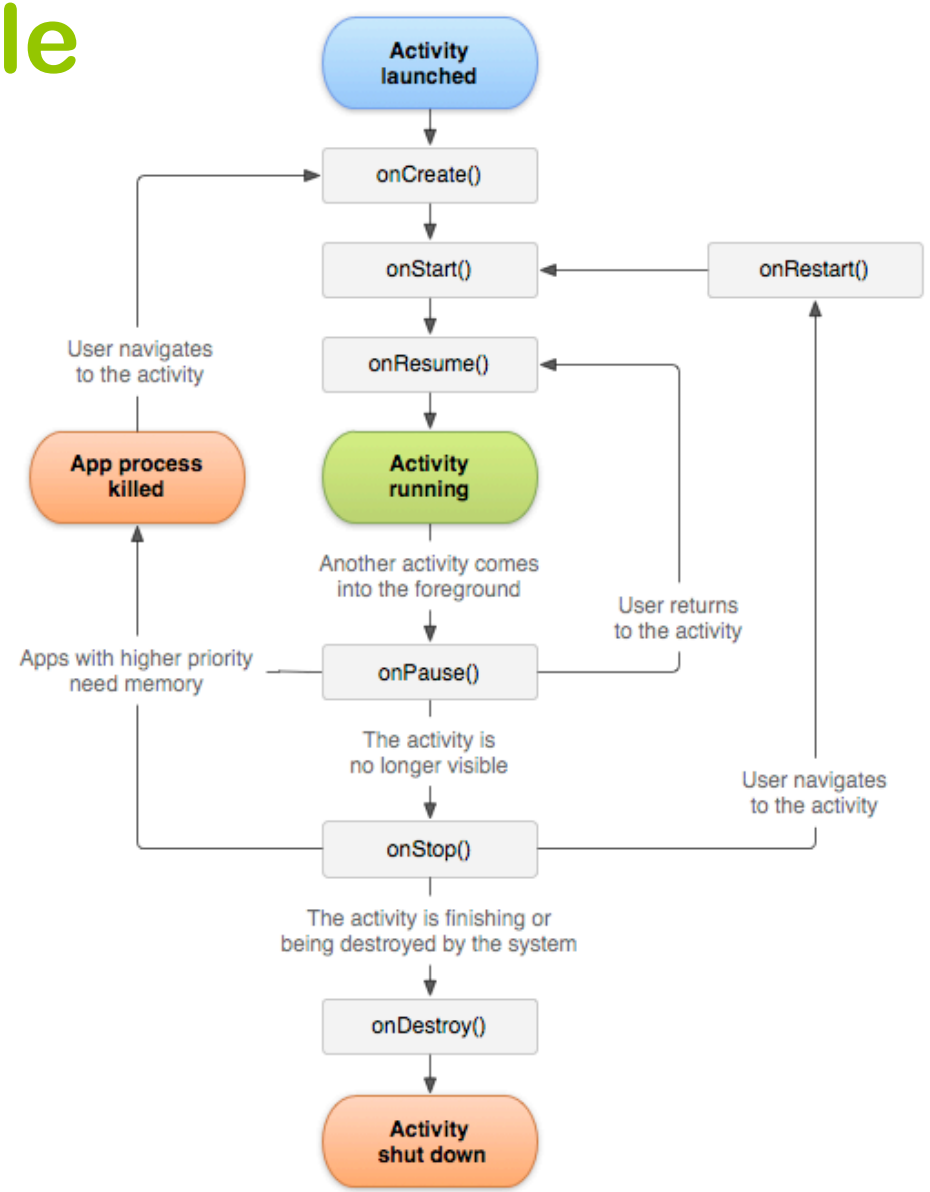
The activity stack

It's similar to the function stack in ordinary programming, with some difference



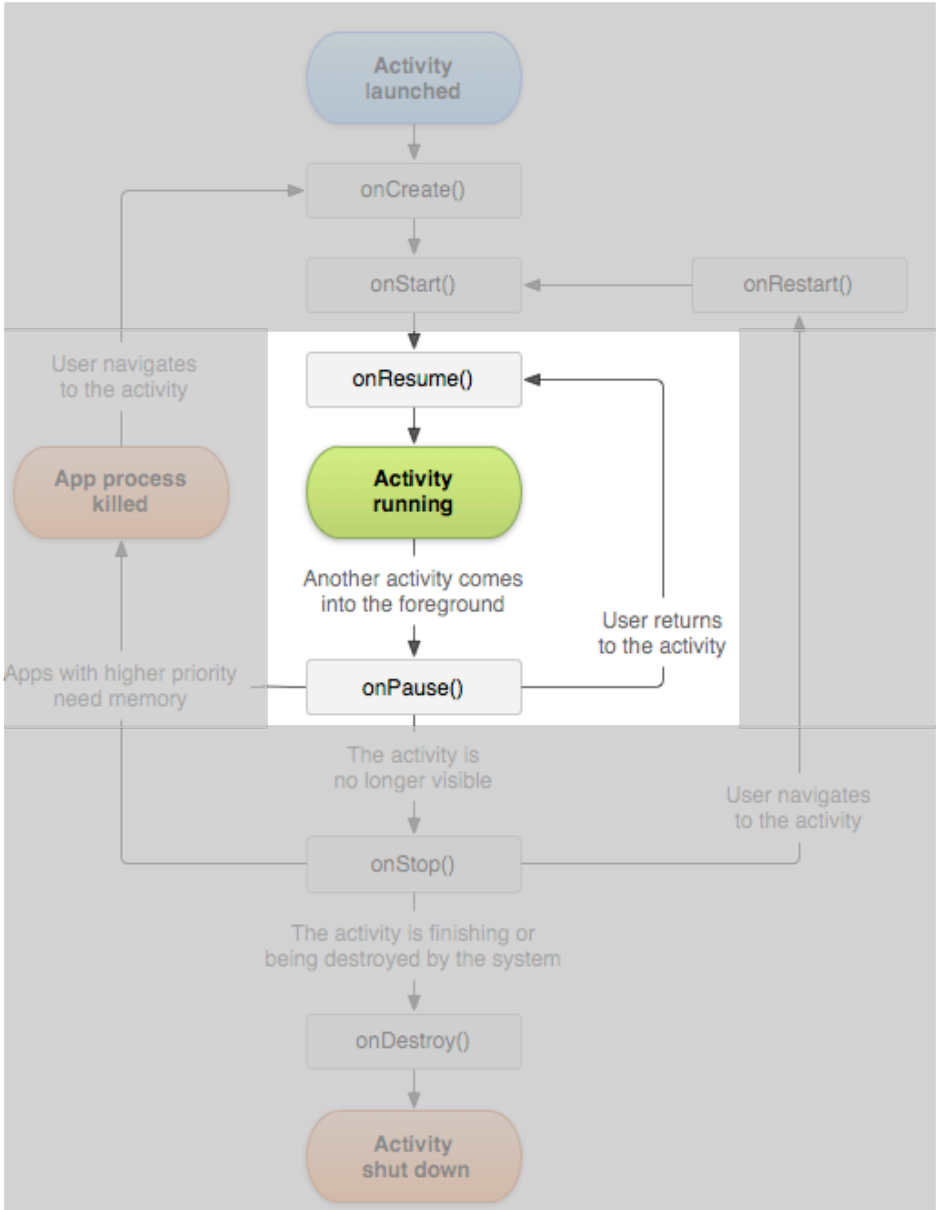
Activity lifecycle

States (colored),
and
Callbacks (gray)



Activity lifecycle

The FOREGROUND lifetime

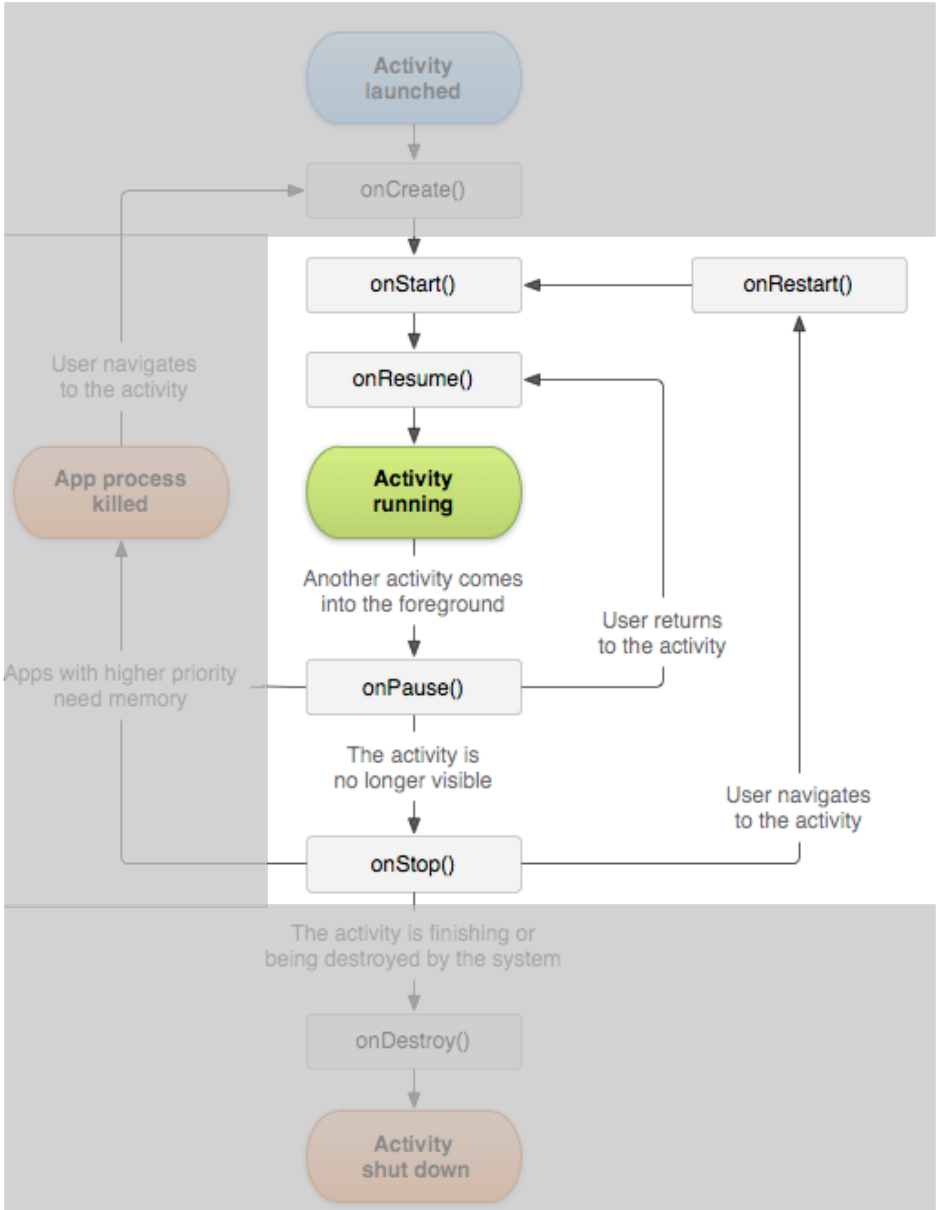


Activity lifecycle

The VISIBLE lifetime

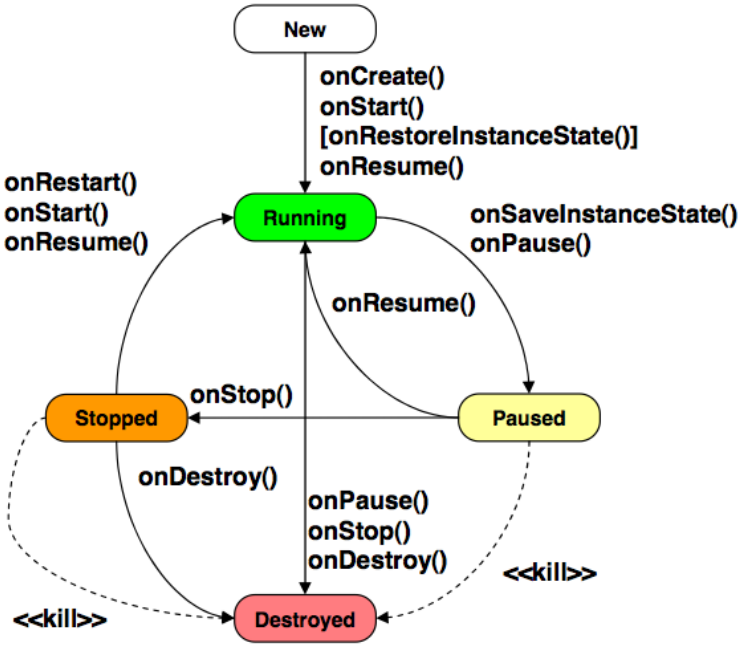
When stopped, your activity should release costly resources, such as network or database connections.

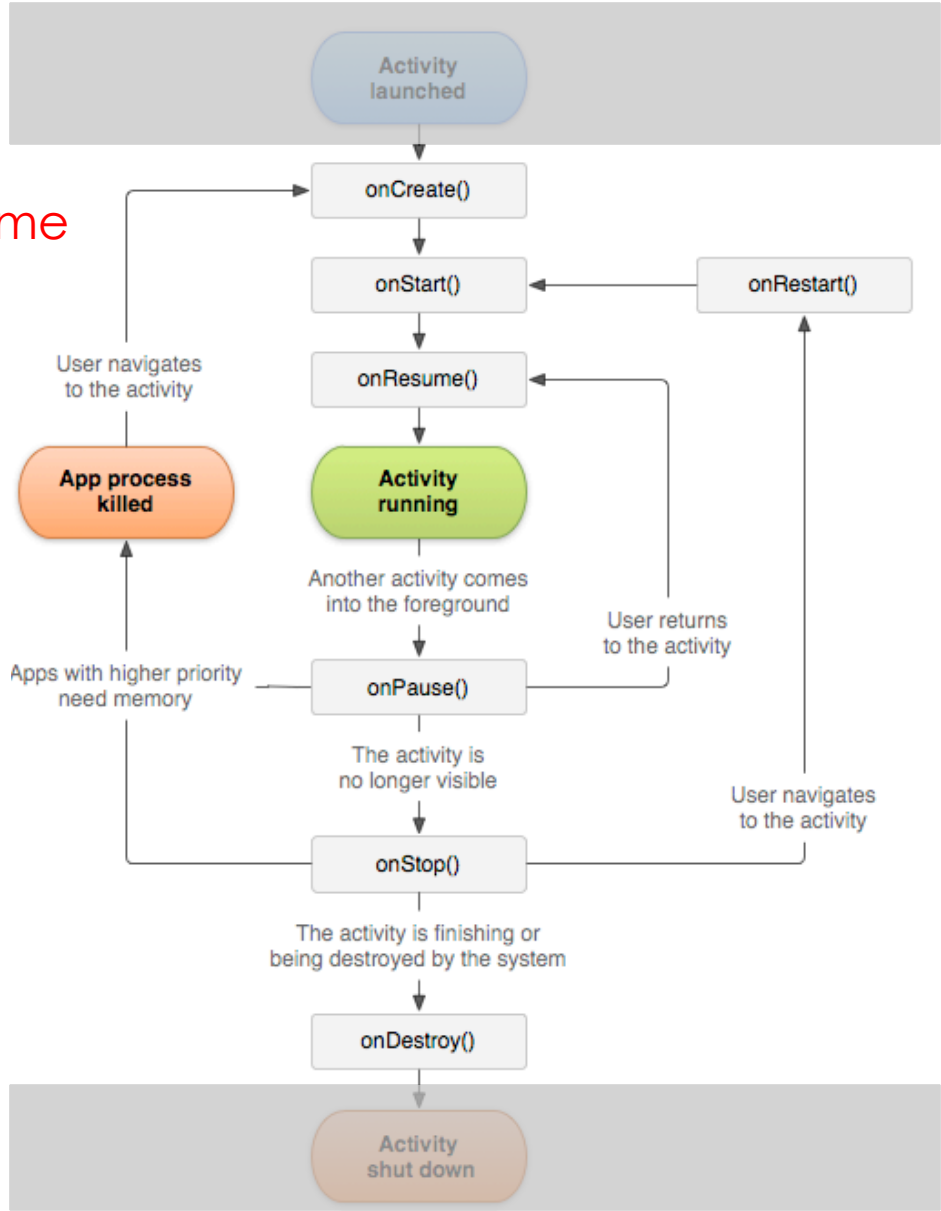
When the activity resumes, you can reacquire the necessary resources and resume actions that were interrupted.



Activity lifecycle

The ENTIRE lifetime



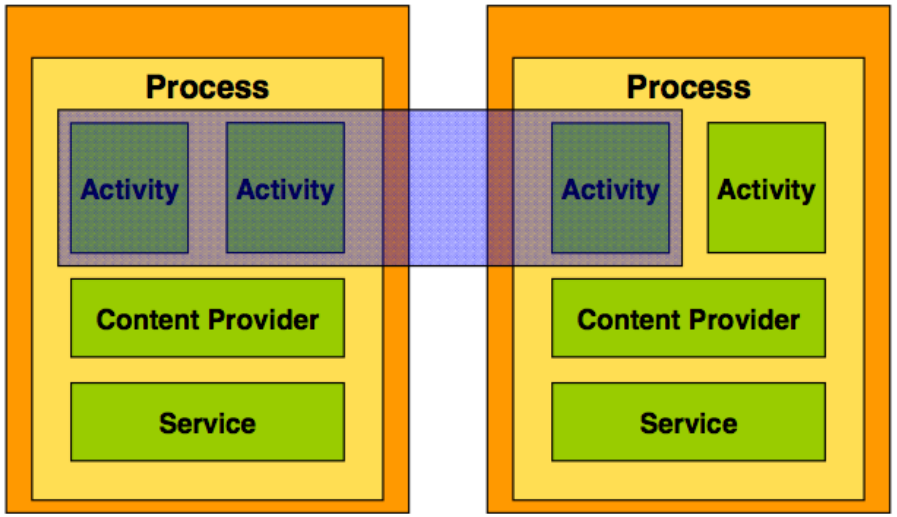


The shocking news…

An activity can start
a second activity in
a DIFFERENT application!
(and hence in a different process…)

We need a name
for this "thing":

We'll call it
"a task"



Task

Not exactly what you might imagine…

Wordnet definitions:

- activity directed toward making or doing something
- work that you are obliged to perform for moral or legal reasons

Task (computing)

From Wikipedia, the free encyclopedia

This article **needs additional cita**
reliable sources. Unsourced mate

A **task** is an execution path through address space.[1] In other words, a set of program instructions that are loaded in memory. The address registers have been loaded with the initial address of the program. At the next clock cycle, the CPU will start execution, in accord with the program. The sense is that some part of 'a plan is being accomplished'. As long as the program remains in this part of the address space, the task can continue, in principle, indefinitely, unless the program instructions contain a `halt`, `exit`, or `return`.

- In the computer field, "task" has the sense of a real-time application, as distinguished from **process**, which takes up space (memory), and execution time. See operating system.
  - Both "task" and "process" should be distinguished from event, which takes place at a **specific** time and **place**, and which can be planned for in a computer program.
    - In a computer graphical user interface (GUI), an event can be as simple as a mouse click or keystroke.

See also [edit]

- Thread
- Process states
- Process
- Computer multitasking

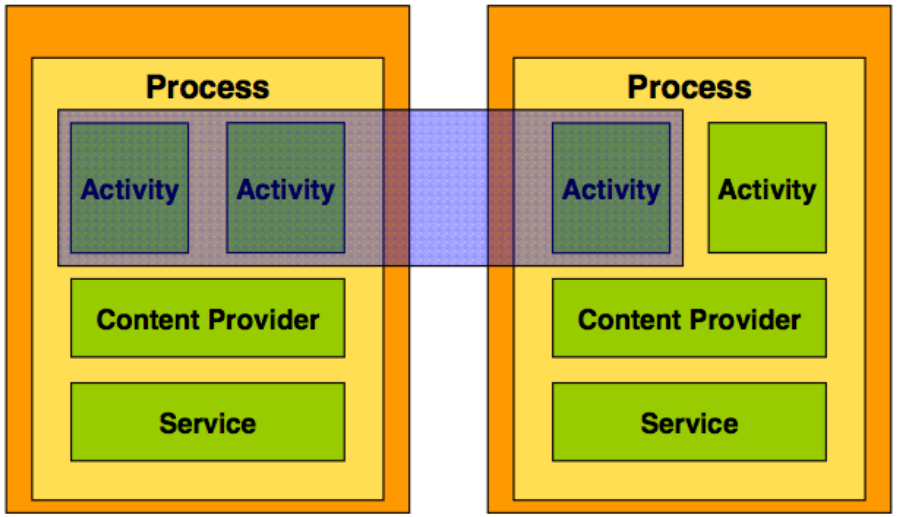
Notes [edit]

1. ^ Data General, *RDOS Reference Manual*

Tasks

Task (what users view as application)

- Collection of related activities
- Capable of spanning multiple processes
- Associated with its own UI history stack



Slide borrowed from Dominik Gruntz

Tasks

An App defines at least one task, may define more.

Activities may come from different applications (favoring reuse).

Android maintains a seamless user experience by keeping the activities in the same task.

Tasks may be moved in the background.

Tasks

The Home screen is the starting place for most tasks.

When the user touches an icon in the application launcher (or a shortcut on the Home screen), that application's task comes to the foreground.

If no task exists for the application (the application has not been used recently), then a new task is created and the "main" activity for that application opens as the root activity in the stack.

If the application has been used recently, its task is resumed (in general with its state preserved: more on this in the next lecture).

Switching among apps

To switching among apps:

long press the home button and you'll see a window of the 6 most recently used apps.

Tap the app you want to switch to.

Task Management

Default behavior:

New activity is added to the same task stack.

NOTE: Activity can have multiple instances, in different tasks or in the same task!

Google recommends:

"Let Android manage it for you. You do not need to bother with multitasking management!"

Process priorities

Active process	Critical priority
Visible process Started service process	High Priority
Background process Empty process	Low Priority

Task Managers ?

Several apps on the store offer a task manager functionality (to kill inactive apps). Are they needed?

Lots of services and applications constantly run in the background just like they do on Windows. However, and this is important, they do not have to use up a ton of resources. A service or app can be loaded, yet use almost no additional memory, and 0% CPU until it actually has to do something.

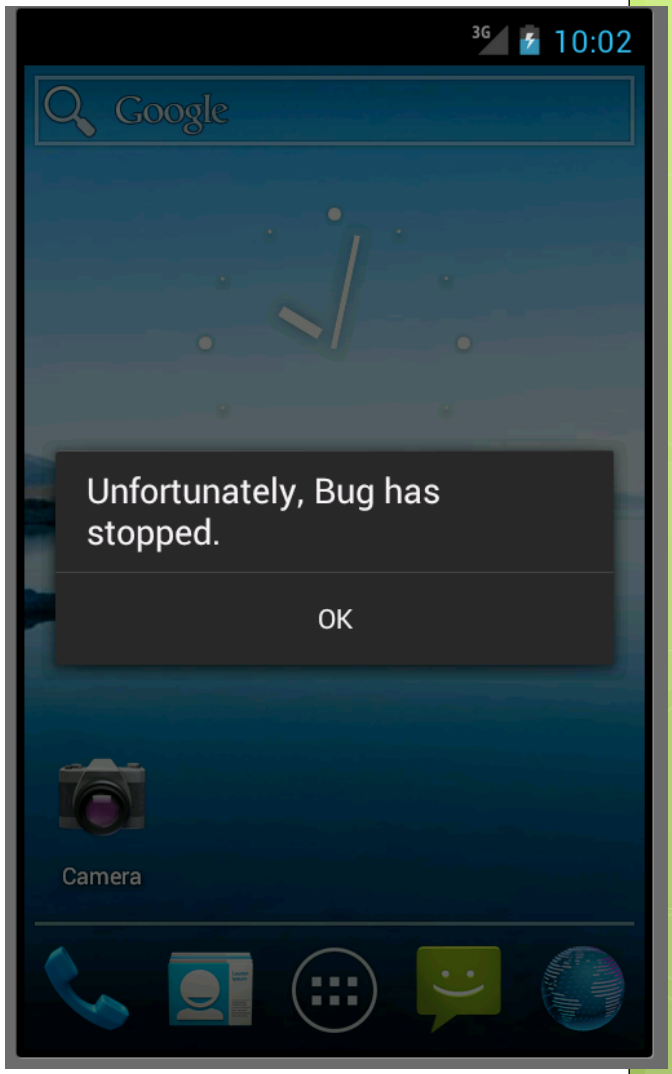
In general, killing off stuff is a waste of time. Android automatically asks apps to close when it needs more memory. Killing off processes also means it'll slow your phone down, as when you do need them again the system will need to reload them.

Basic tips: having troubles…

Marco Ronchetti
Università degli Studi di Trento

A bugged program

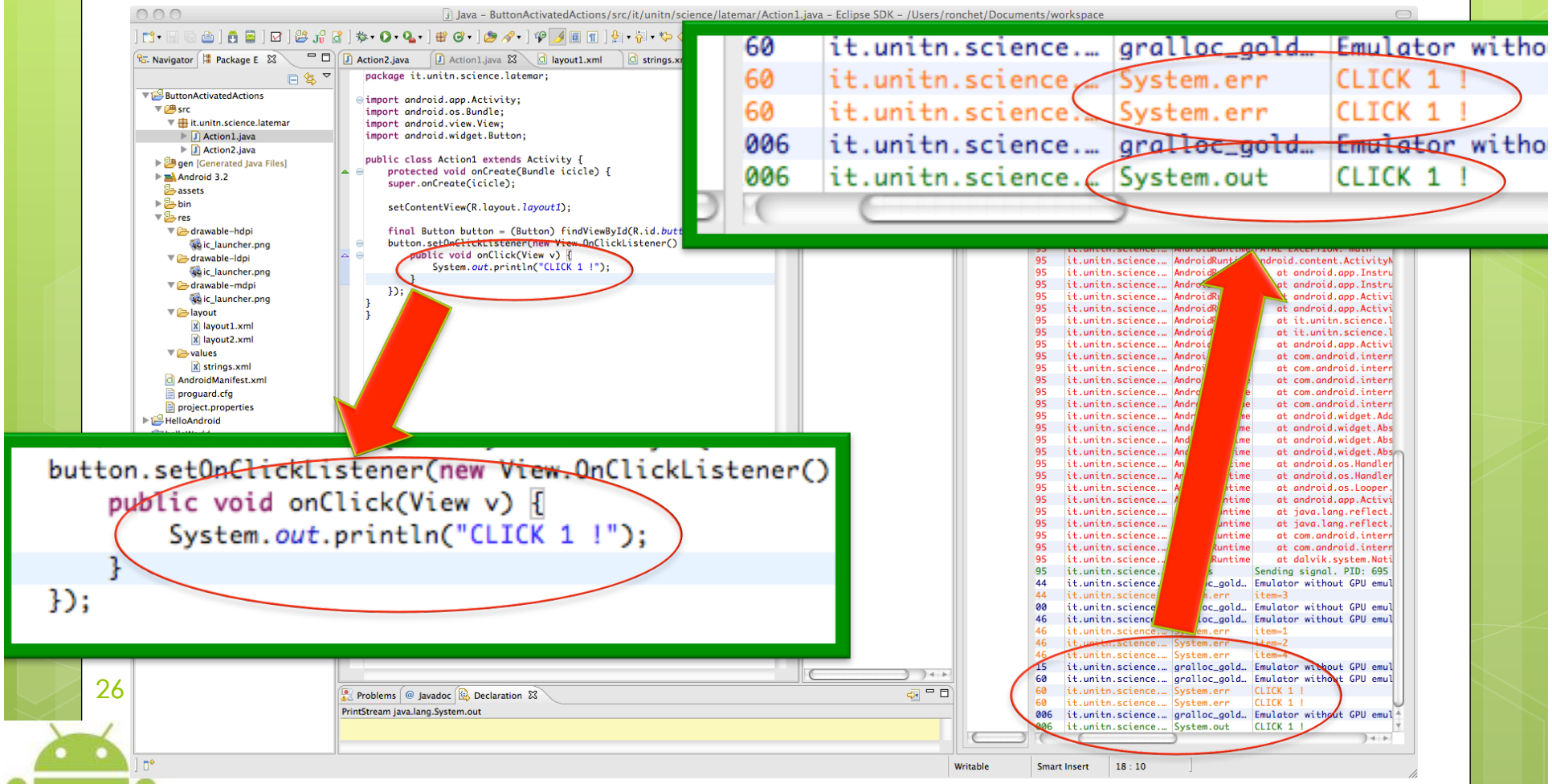
```
package com.example.helloandroid;

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

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



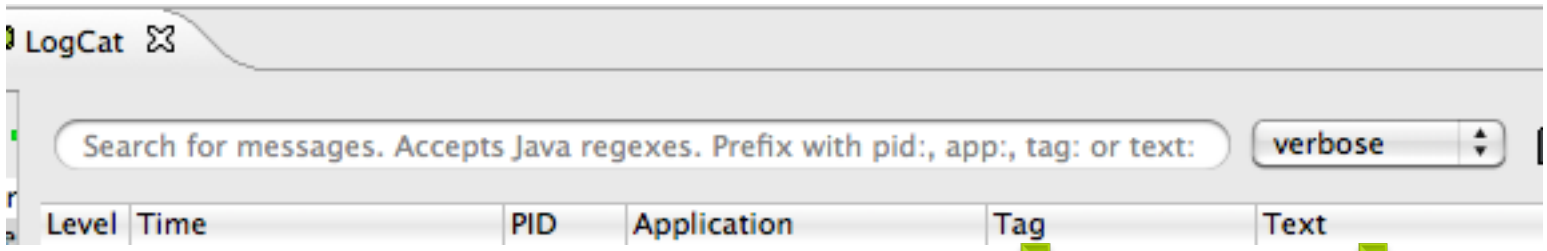
Basic tips: printing on the console

Marco Ronchetti
Università degli Studi di Trento

Printing in Eclipse



The Logger console



Log.*d("CalledActivity","OnCreate ");*

Basic UI elements: Android Buttons (Basics)

Marco Ronchetti
Università degli Studi di Trento

Let's work with the listener

```
Button button = …;
        button.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        Log.d("TRACE", "Button has been clicked ");
                }
        });
```

Anonymous Inner Class

In Swing it was

```
JButton button=…
button.addActionListener (new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        …;
                }
        });
```

public class
Button

extends TextView

java.lang.Object
↳android.view.View
↳android.widget.TextView
↳android.widget.Button

Known Direct Subclasses
CompoundButton

Known Indirect Subclasses
CheckBox, RadioButton, Switch, ToggleButton

Let's work with the listener

```
Button button = ...;
        button.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        Log.d("TRACE", "Button has been clicked ");
                }
        });
```

The event target Is passed

MAIN DIFFERENCE

The event is passed

In Swing it was

```
JButton button=...
button.addActionListener (new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        ...;
                }
        });
```


An alternative

The various View classes expose several public callback methods that are useful for UI events.

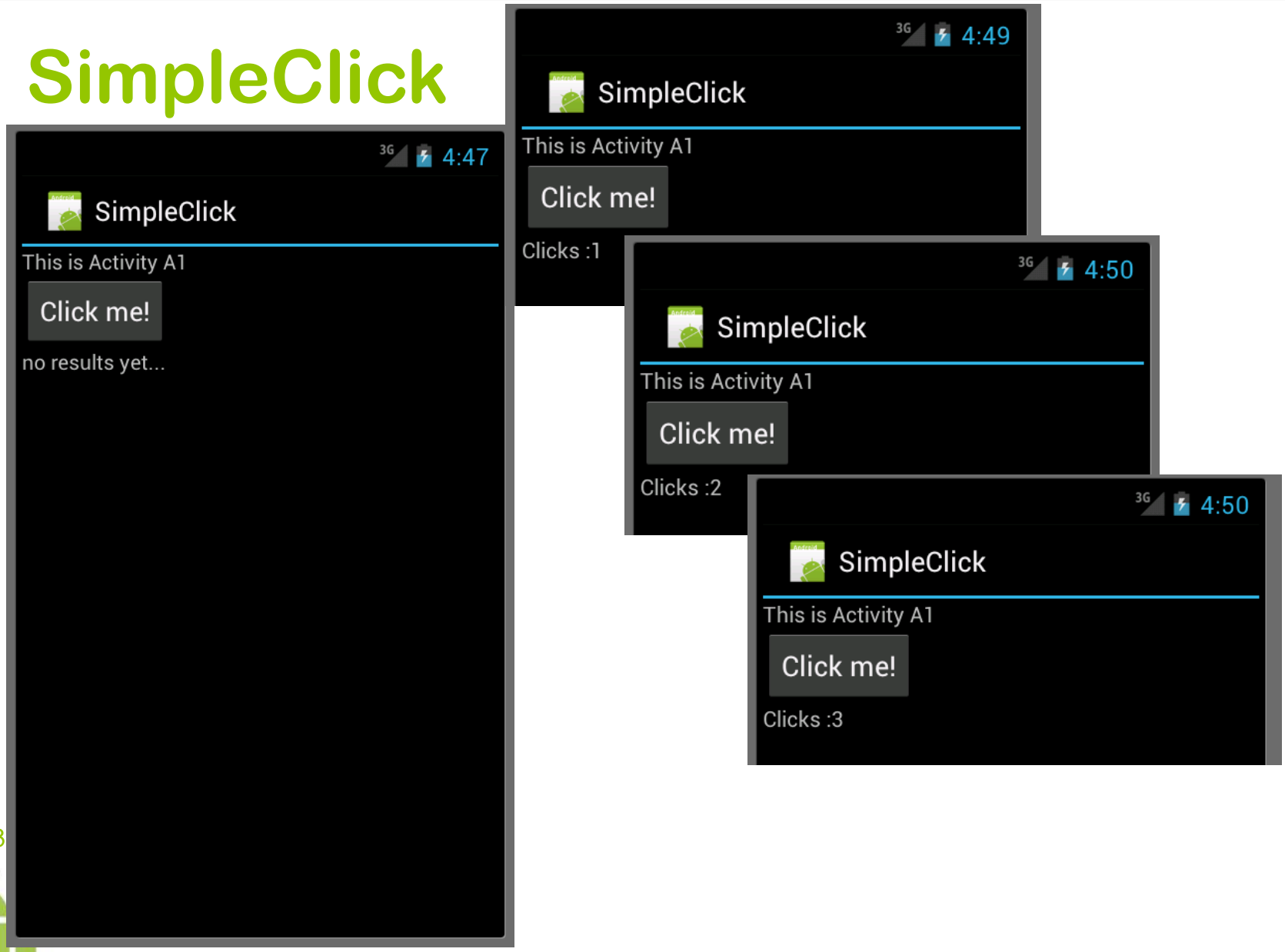
These methods are called by the Android framework when the respective action occurs on that object. For instance, when a View (such as a Button) is touched, the onTouchEvent() method is called on that object.

However, in order to intercept this, you must extend the class and override the method.

Extending Button to deal with events

```
class MyButton extends Button {
public boolean onTouchEvent(MotionEvent event) {
   int eventAction = event.getAction();
   switch (eventAction) {
      case MotionEvent.ACTION_DOWN:  // finger touches the screen
         …;
         break;

      case MotionEvent.ACTION_MOVE:  // finger moves on the screen
         …;
          break;

      case MotionEvent.ACTION_UP:   // finger leaves the screen
         …;
          break;
   }
// tell the system that we handled the event and no further processing is needed
   return true;
}
```

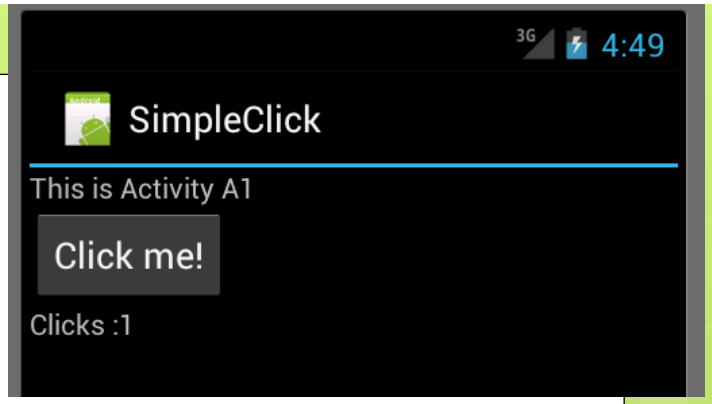

SimpleClick

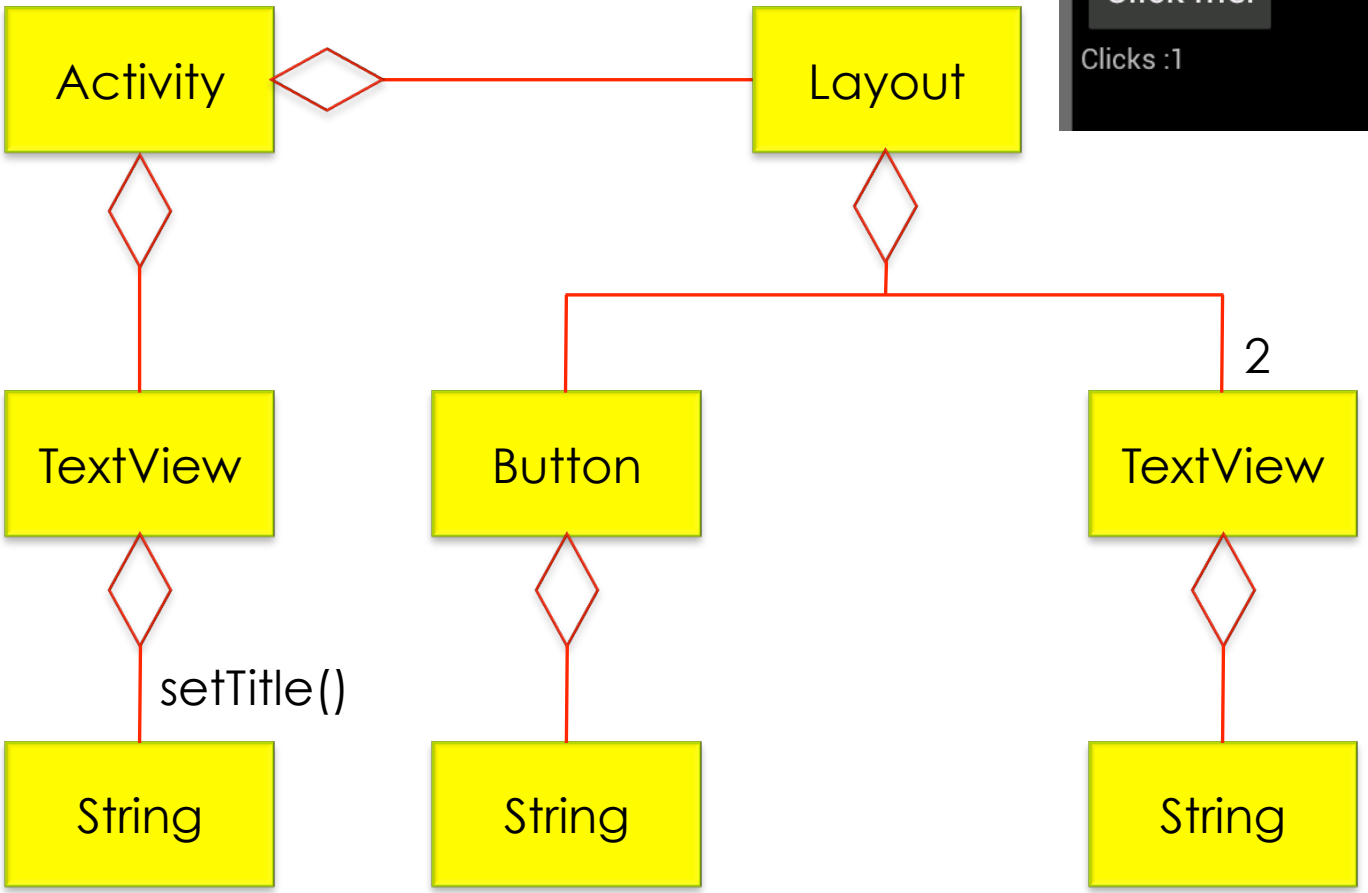


Let's recap how to build an app

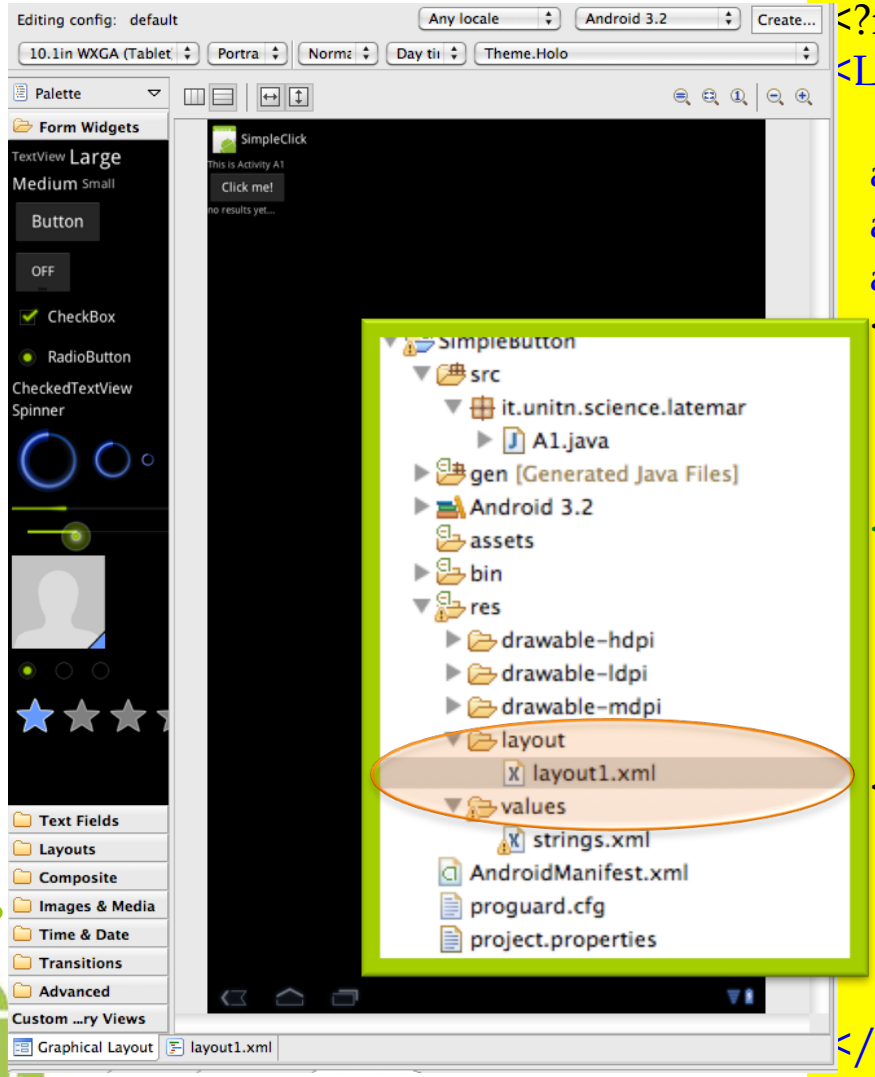
1) Define the Activity Resources
    1) Choose a Layout
    2) Add the components via XML
    3) Define the strings
2) Code the activity
3) Add info to the Manifest (if needed)

UML Diagram


4:49
SimpleClick
This is Activity A1
Click me!
Clicks :1

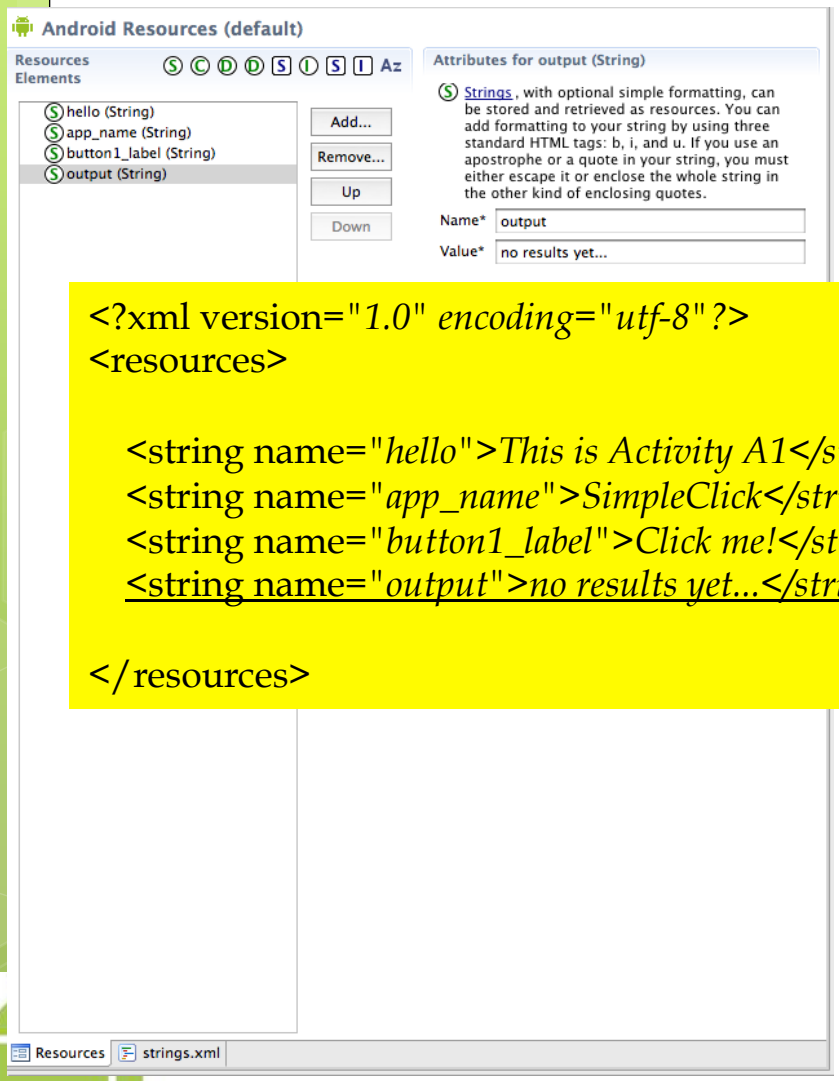

Activity
Layout
TextView
Button
TextView
2
setTitle()
String
String
String

Let's define the aspect of layout1

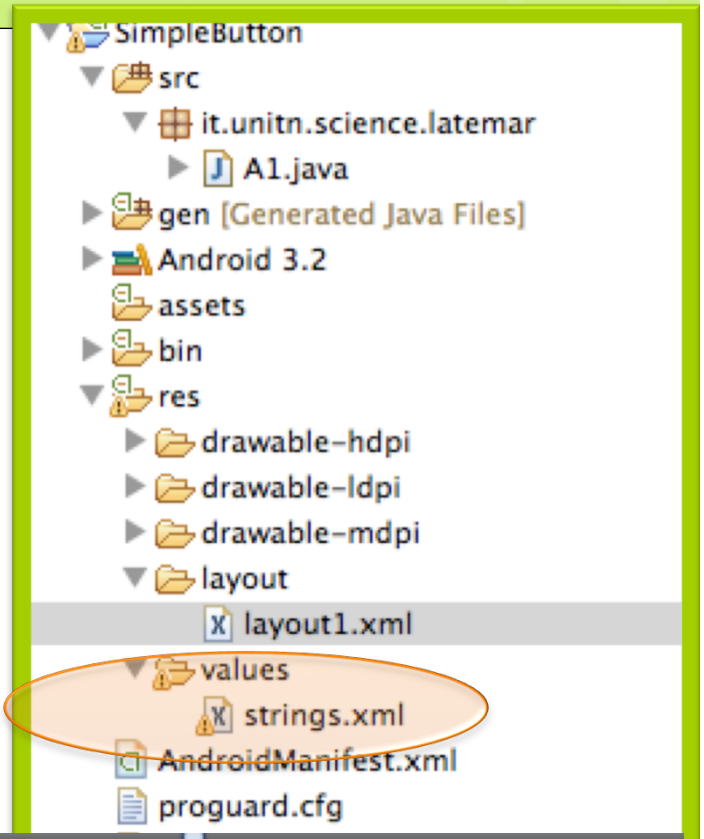


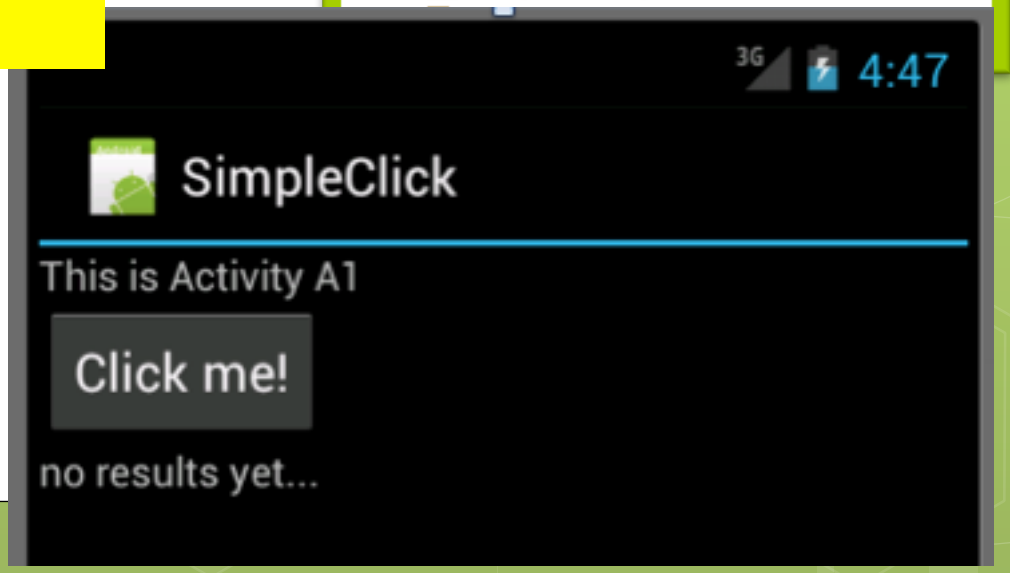

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
   "http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
  <TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello" />
  <Button
     android:id="@+id/button1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="@string/button1_label" />
  <TextView
     android:id="@+id/tf1"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/output" />
</LinearLayout>
```


Let's define the strings




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

    <string name="hello">This is Activity A1</string>
    <string name="app_name">SimpleClick</string>
    <string name="button1_label">Click me!</string>
    <string name="output">no results yet...</string>

</resources>
```





SimpleClick – A1

```
package it.unitn.science.latemar;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class A1 extends Activity {
   int nClicks=0;
   protected void onCreate(Bundle b) {
      super.onCreate(b);

      setContentView(R.layout.layout1);

      final Button button = (Button) findViewById(R.id.button1);

      final TextView tf = (TextView) findViewById(R.id.tf1);

      button.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                        tf.setText("Clicks :"+(++nClicks));
                  }
      });
   }
}
```