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

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

In JavaFX it was

```
Button btn=...
btn.addEventHandler(new EventHandler() {
                public void handle(Event t) {
                        ...;
                }
});
```

In Swing it was

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


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

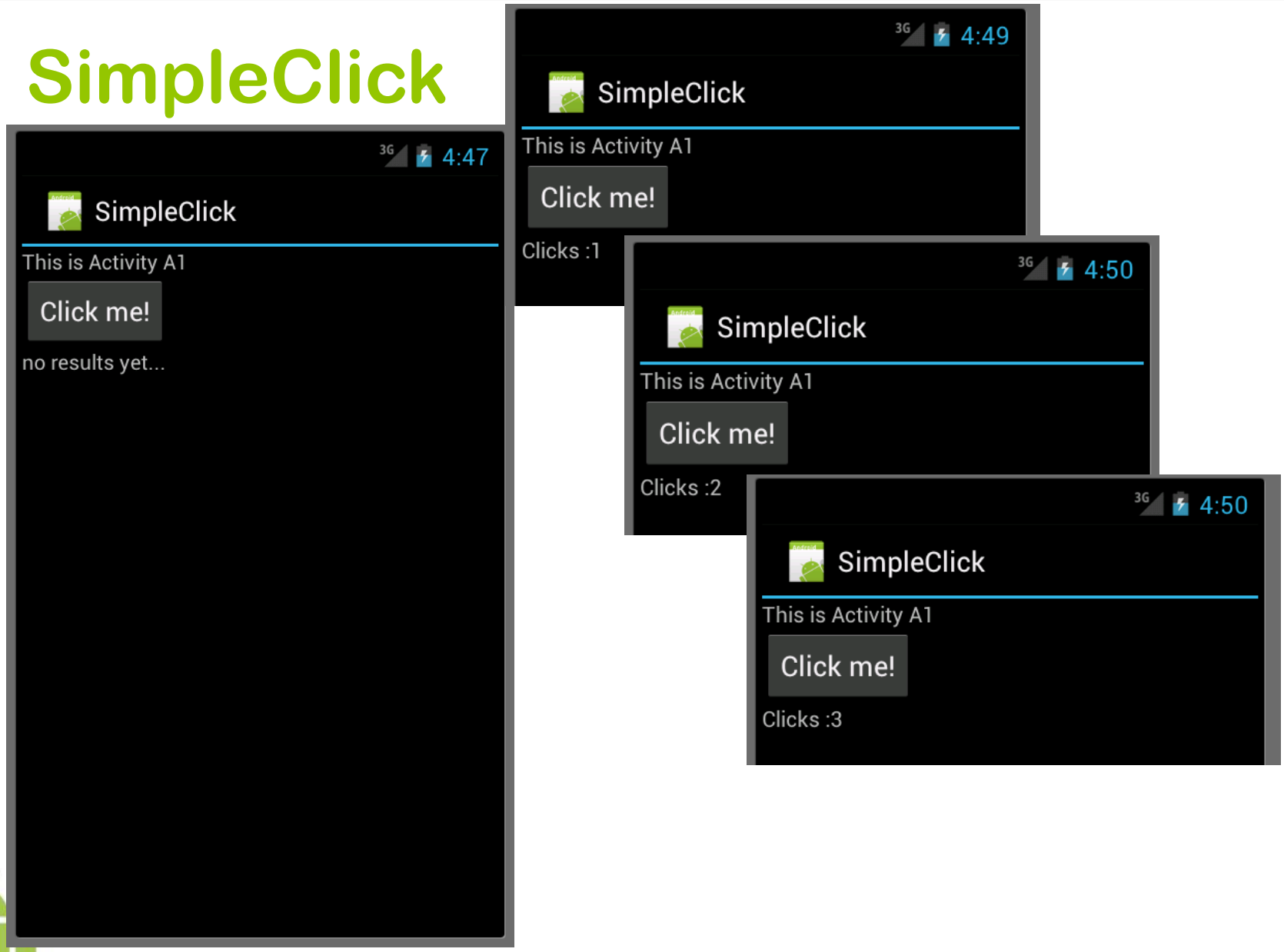
MAIN DIFFERENCE

In Swing (and in JavaFX):

The event
is passed

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

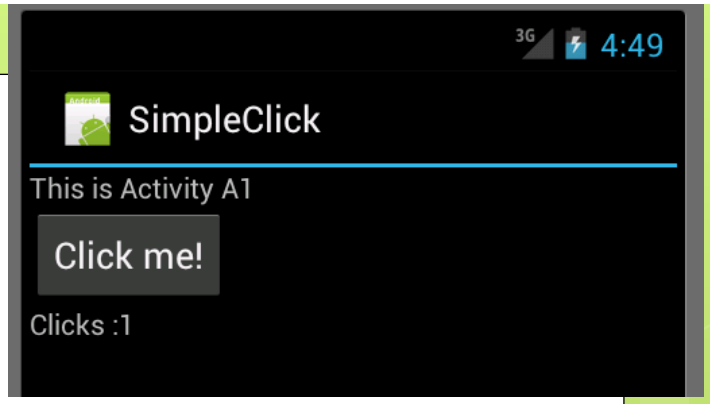

SimpleClick

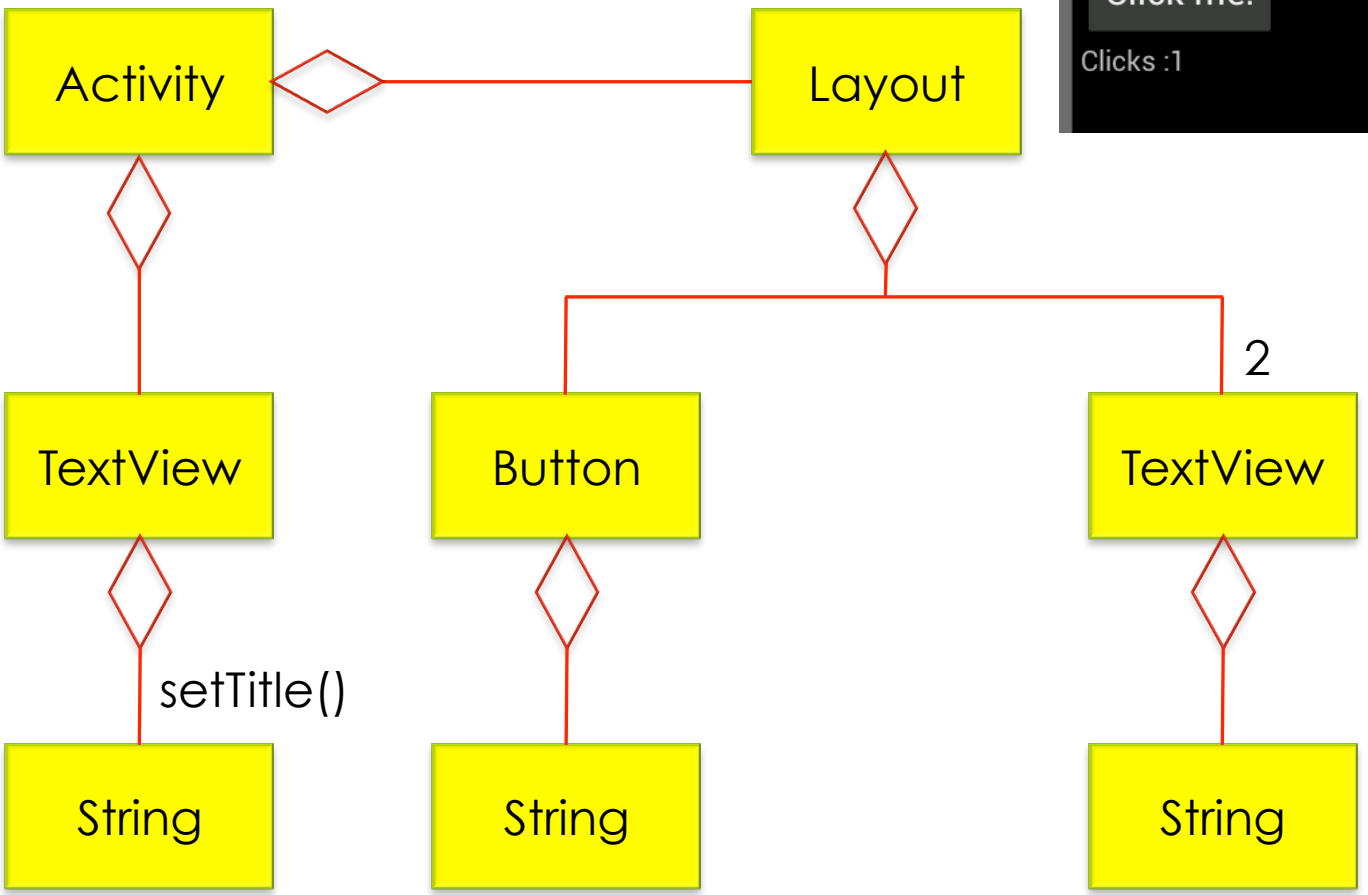


Let's recap how to build an app

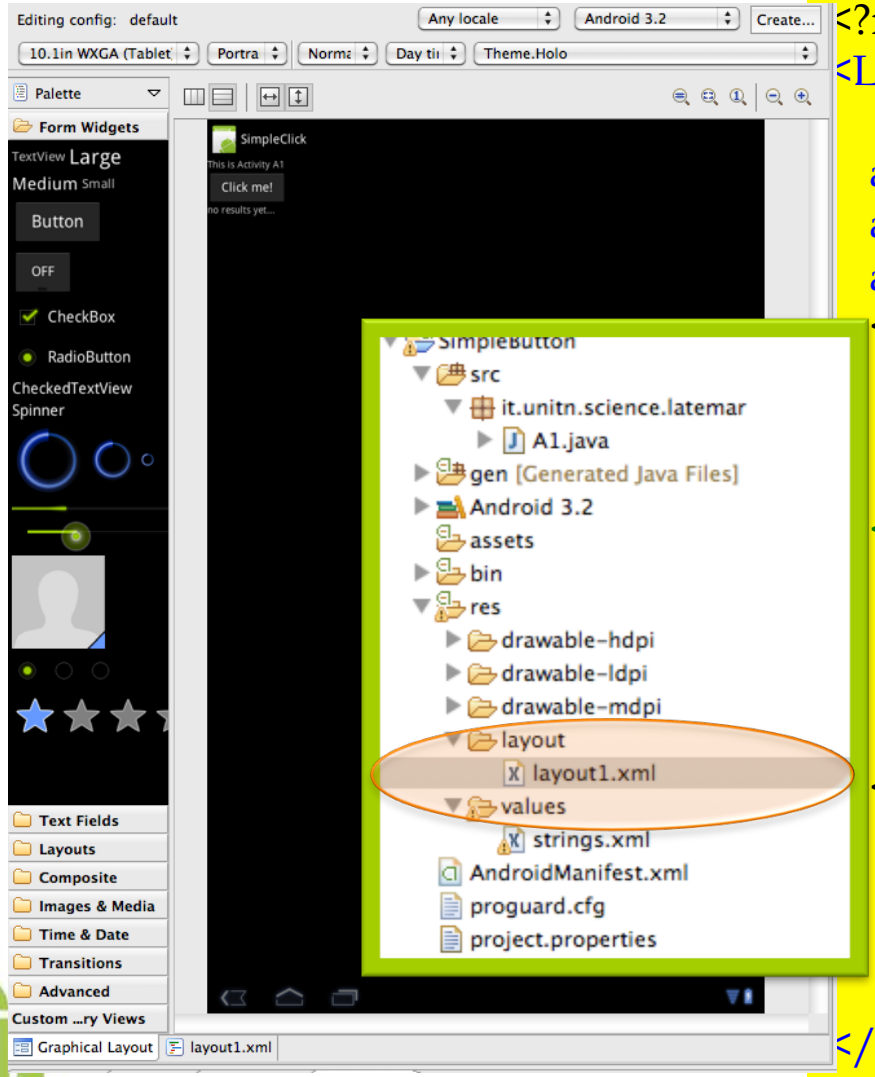
1) Define the Activity Resources
    1) Choose a Layout
    2) Add the components via XML
    3) Define the strings
2) Code the activity
3) Add info to the Manifest (if needed)

UML Diagram


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

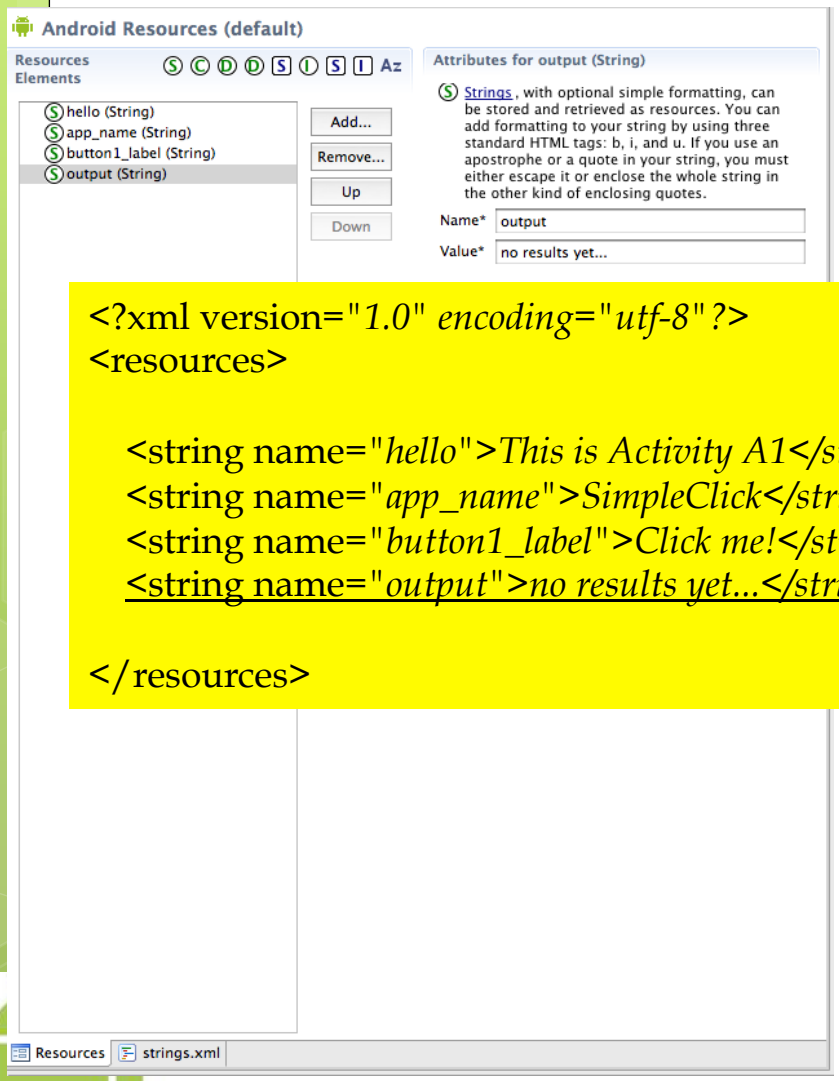

Activity
Layout
TextView
Button
TextView
2
setTitle()
String
String
String

Let's define the aspect of layout1

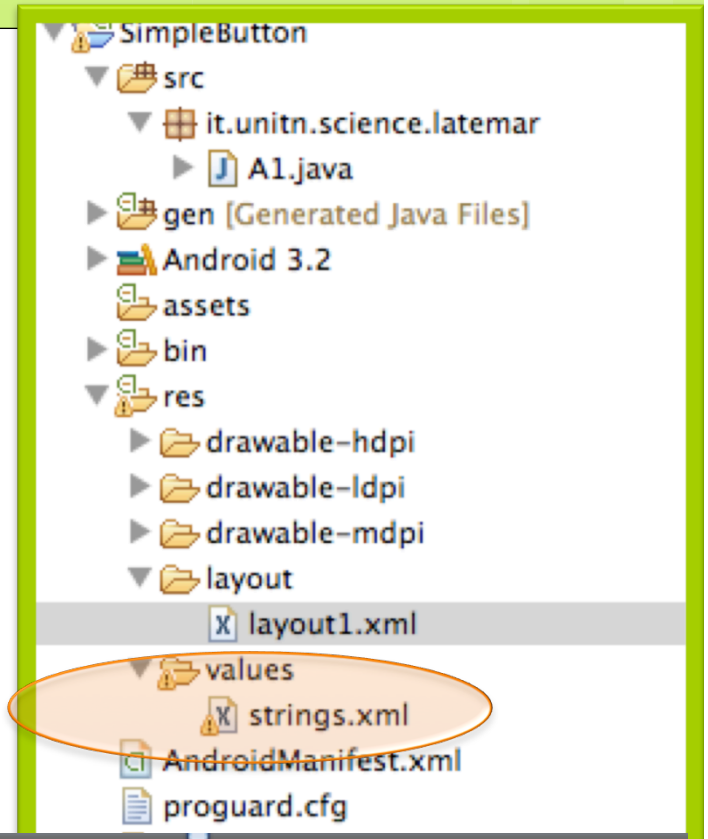


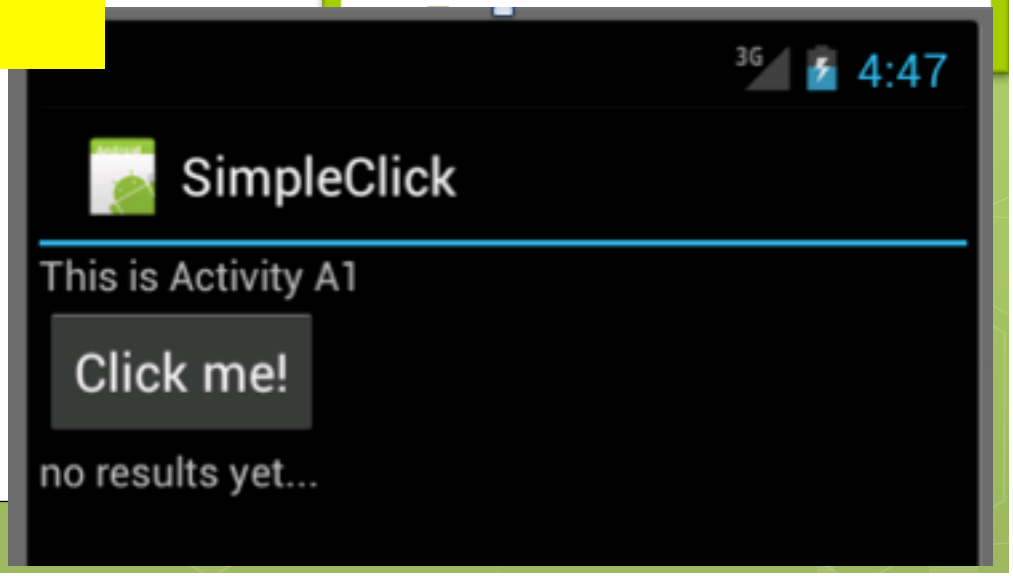

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
   "http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
  <TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     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));
                  }
      });
   }
}
```


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;
}
```


Calling Activities: Explicit Intents

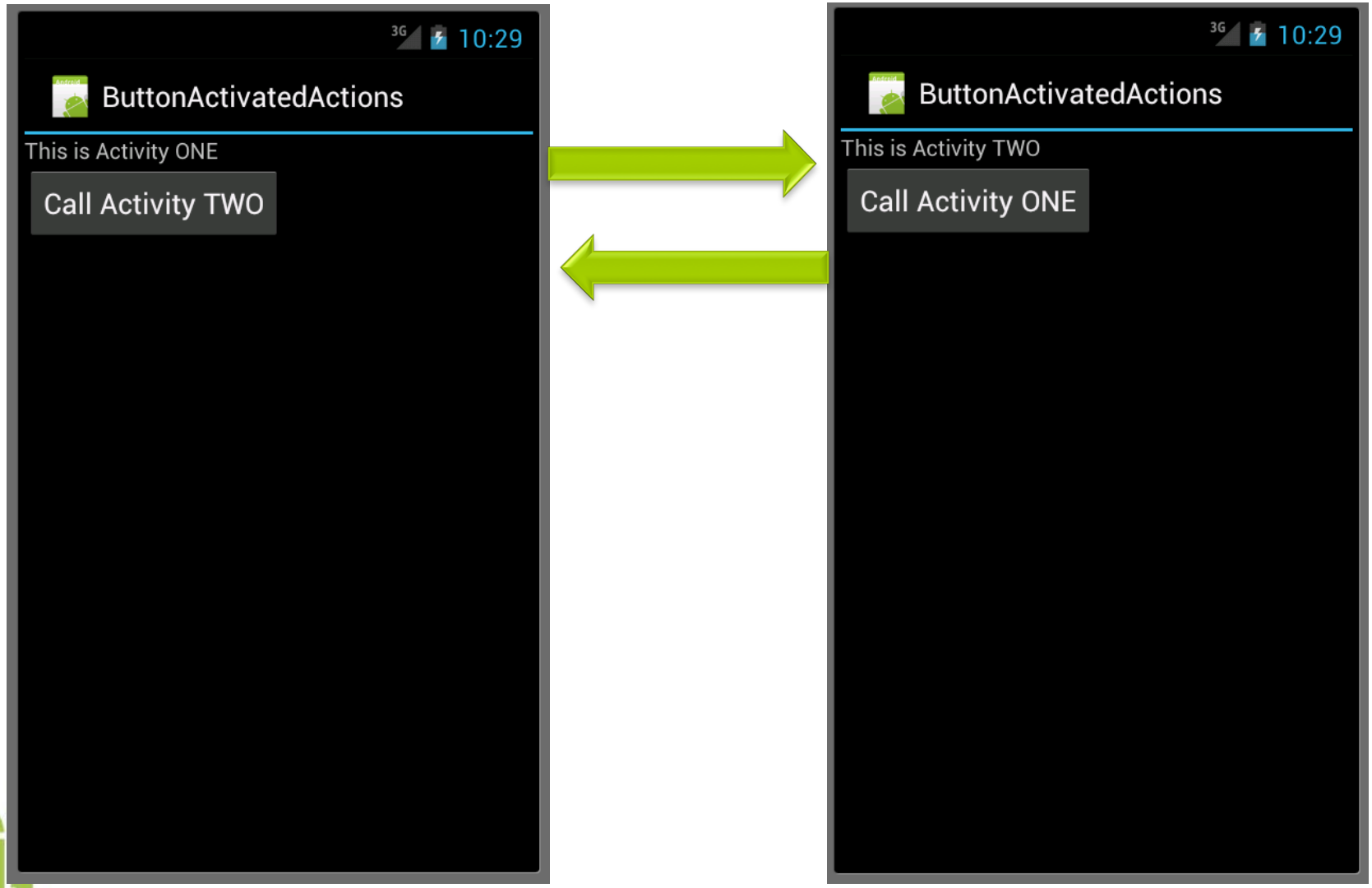
Marco Ronchetti
Università degli Studi di Trento

startActivity(Intent x)

startActivity(Intent x) (method of class Activity)

- starts a new activity, which will be placed at the top of the activity stack.
- takes a single argument which describes the activity to be executed.
- An intent is an abstract description of an operation to be performed.

A simple example: A1 calls A2



Explicit Intents

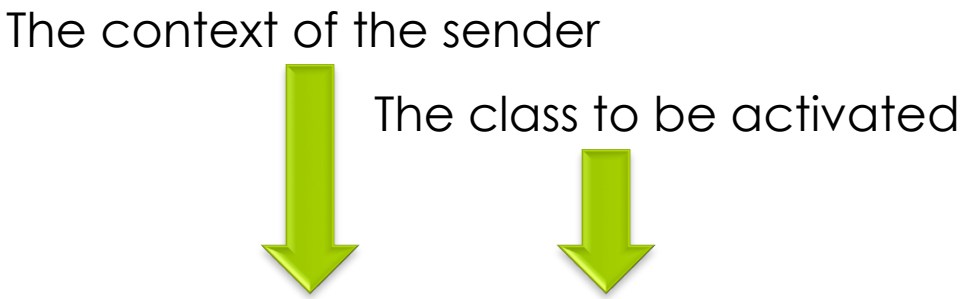
We will use the basic mode:

"Explicit starting an activity"

Explicit Intents specify the exact class to be run.

Often these will not include any other information, simply being a way for an application to launch various internal activities it has as the user interacts with the application.

Intent



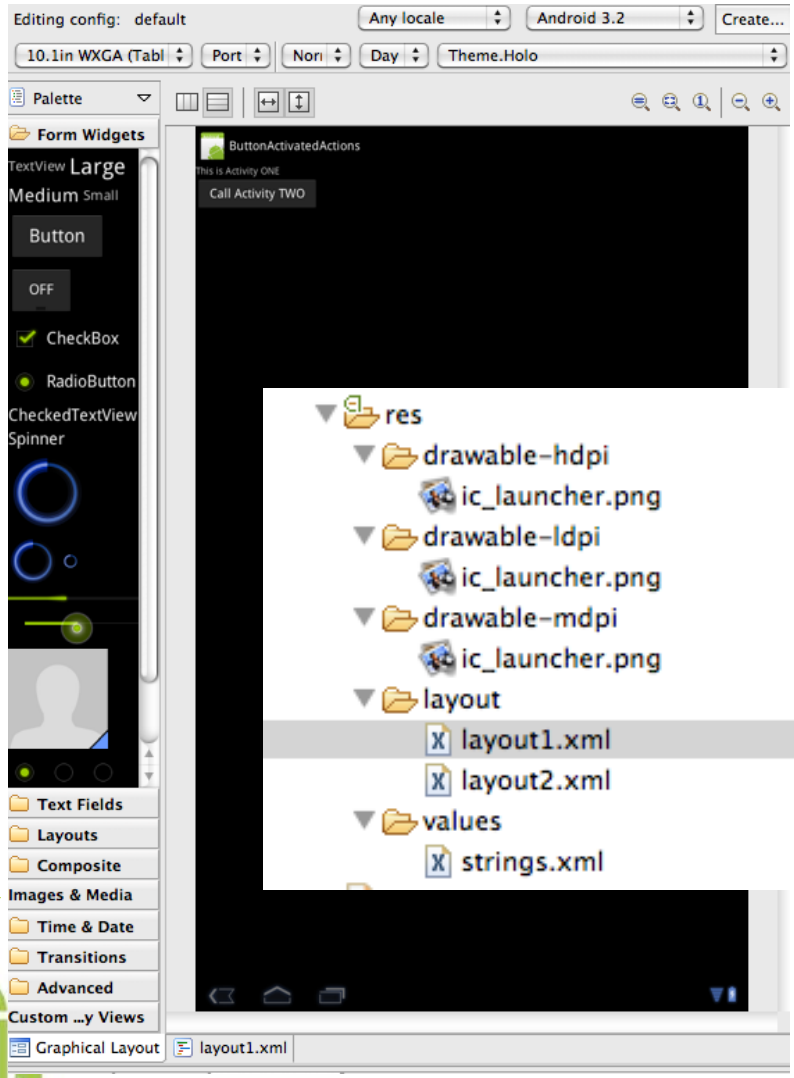
new Intent(Context c, Class c);

Remember that a Context is a wrapper for global information about an application environment, and that Activity subclasses Context

Equivalent form:

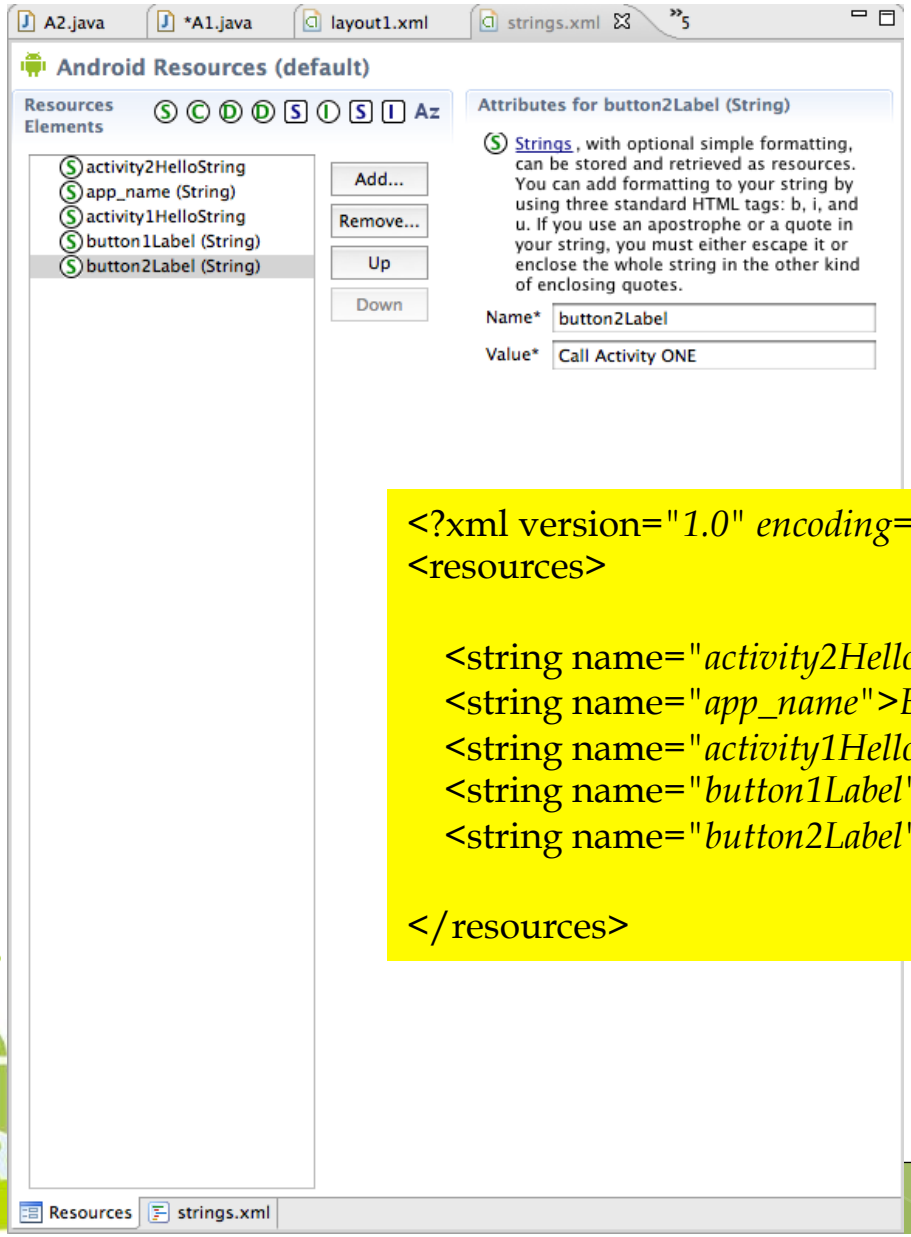
```
Intent i=new Intent();
i.setClass(Context c1, Class c2);
```

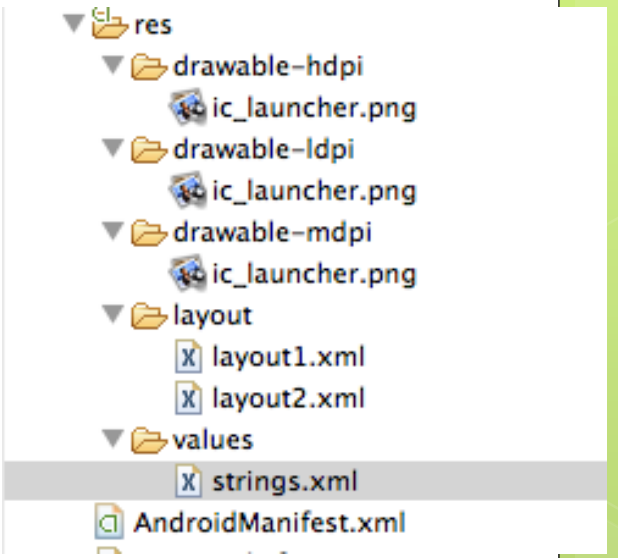

Let's define the aspect of layout1




```
<?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/activity1HelloString" />

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/button1Label" />

</LinearLayout>
```


Let's define the strings






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

    <string name="activity2HelloString">This is Activity TWO</string>
    <string name="app_name">ButtonActivatedActions</string>
    <string name="activity1HelloString">This is Activity ONE</string>
    <string name="button1Label">Call Activity TWO</string>
    <string name="button2Label">Call Activity ONE</string>

</resources>
```


A1 and A2

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

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

public class A1 extends Activity {
   protected void onCreate(Bundle icicle) {
     super.onCreate(icicle);

     setContentView(R.layout.layout1);
     final Button button = (Button) findViewById(
              R.id.button1);
     button.setOnClickListener(
         new View.OnClickListener() {
              public void onClick(View v) {
                   Intent intent = new Intent(A1.this, A2.class);
                   startActivity(intent);
              }
         });
   }
}
```

Anonymous Inner Class

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

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

public class A2 extends Activity {
   protected void onCreate(Bundle icicle) {
     super.onCreate(icicle);

     setContentView(R.layout.layout2);
     final Button button = (Button) findViewById(
              R.id.button2);
     button.setOnClickListener(
         new View.OnClickListener() {
              public void onClick(View v) {
                   Intent intent = new Intent(A2.this, A1.class);
                   startActivity(intent);
              }
         });
   }
}
```

A1.this ? What's that?

```
final Intent intent = new Intent(this, A2.class);
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            startActivity(intent);
        }
    });
```

```
final Activity me=this;
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent(me, A2.class);
            startActivity(intent);
        }
    });
```

```
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent(A1.this, A2.class);
            startActivity(intent);
        }
    });
```

3 different ways to do the same thing

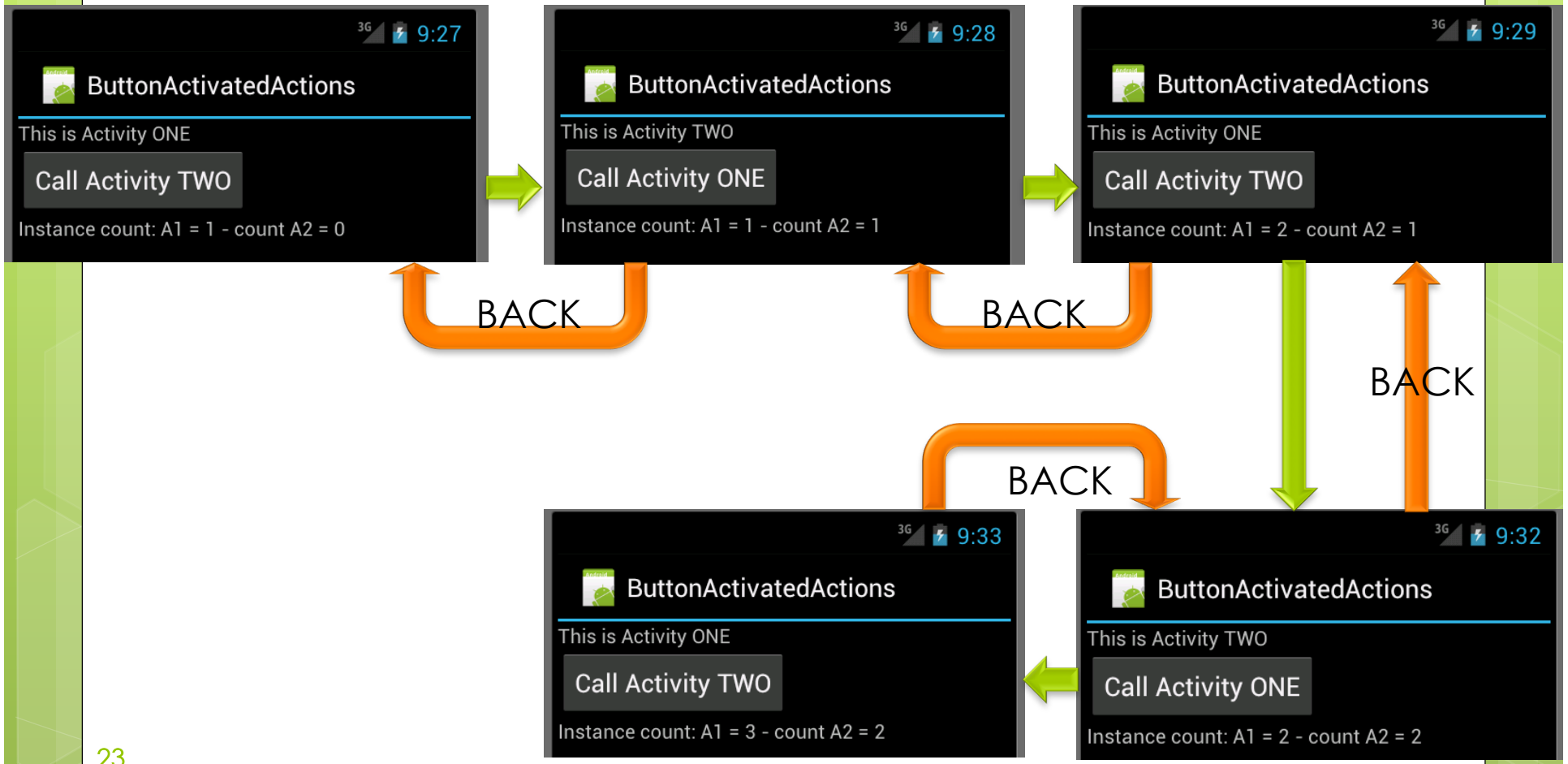
Let's declare A2 in the manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="it.unitn.science.latemar"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="13" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name="A1"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="A2"></activity>
    </application>

</manifest>
```


How many instances?

Marco Ronchetti
Università degli Studi di Trento

How many instances?

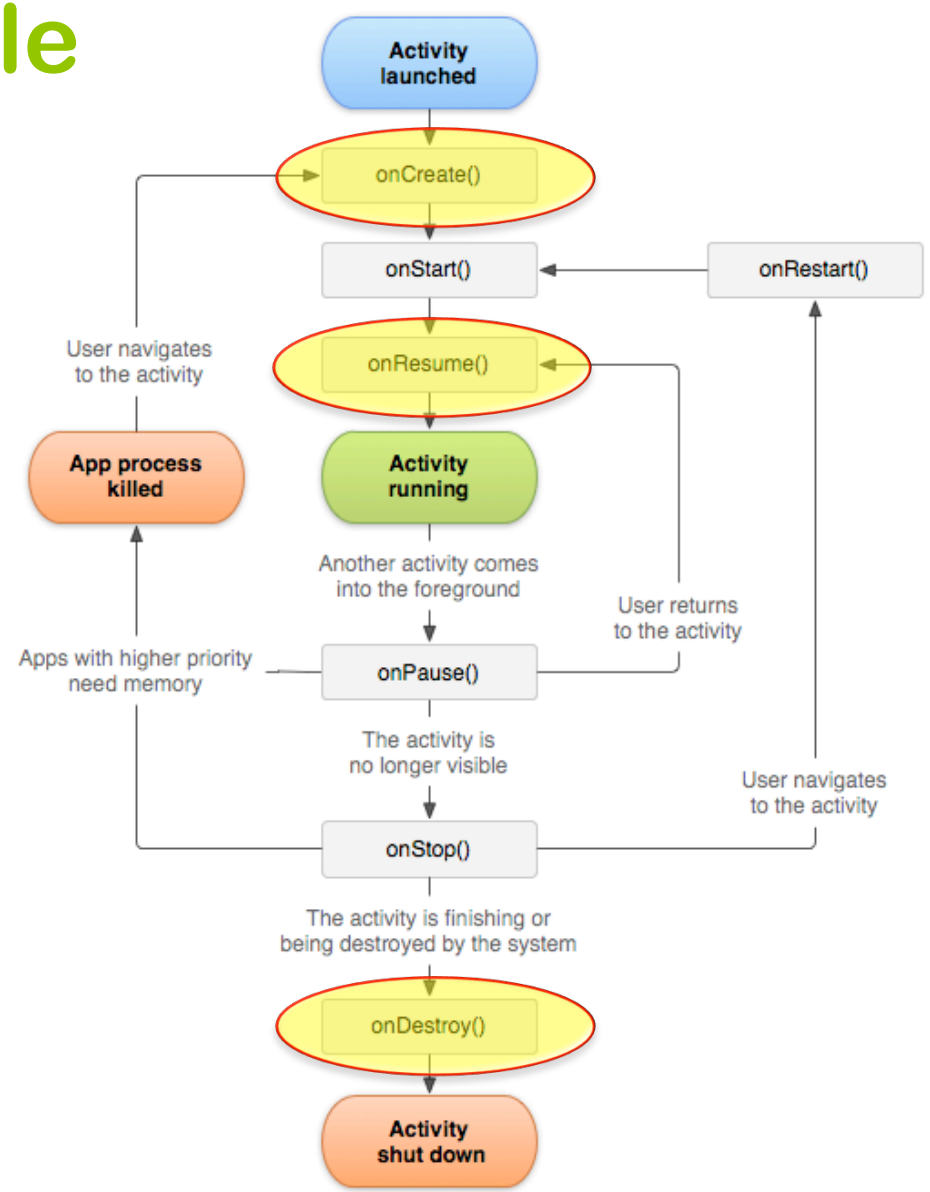


The code

```
public class A1 extends Activity {
   static int instances = 0;
   TextView tf = null;
   protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        instances++;
        setContentView(R.layout.layout1);
        tf = (TextView) findViewById(R.id.instanceCount);
        final Button button = (Button) findViewById(R.id.button1);
        final Intent intent = new Intent(this, A2.class);
        button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
              startActivity(intent);
              }});
        }
   protected void onDestroy() {
      super.onDestroy();
      instances--;
   }
   protected void onResume() {
       super.onResume();
       if (tf != null)
         tf.setText("Instance count: A1 = " +
              A1.instances+" - count A2 = " +
              A2.instances);
   }
}
```


Activity lifecycle

States (colored),
and
Callbacks (gray)



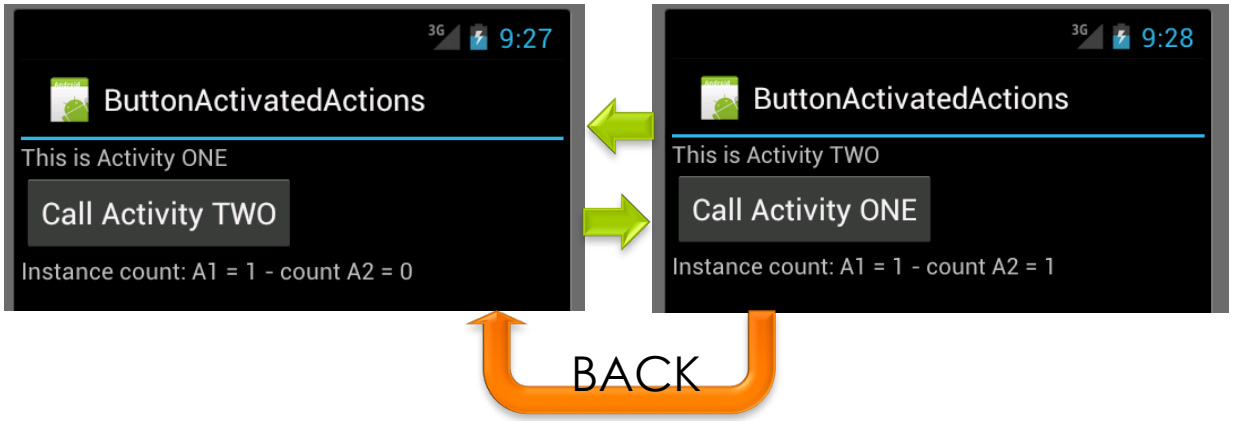
The xml

```
<?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" >

   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/activity1HelloString" />

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/button1Label" />

   <TextView
      android:id="@+id/instanceCount"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/instanceCount" />

</LinearLayout>
```

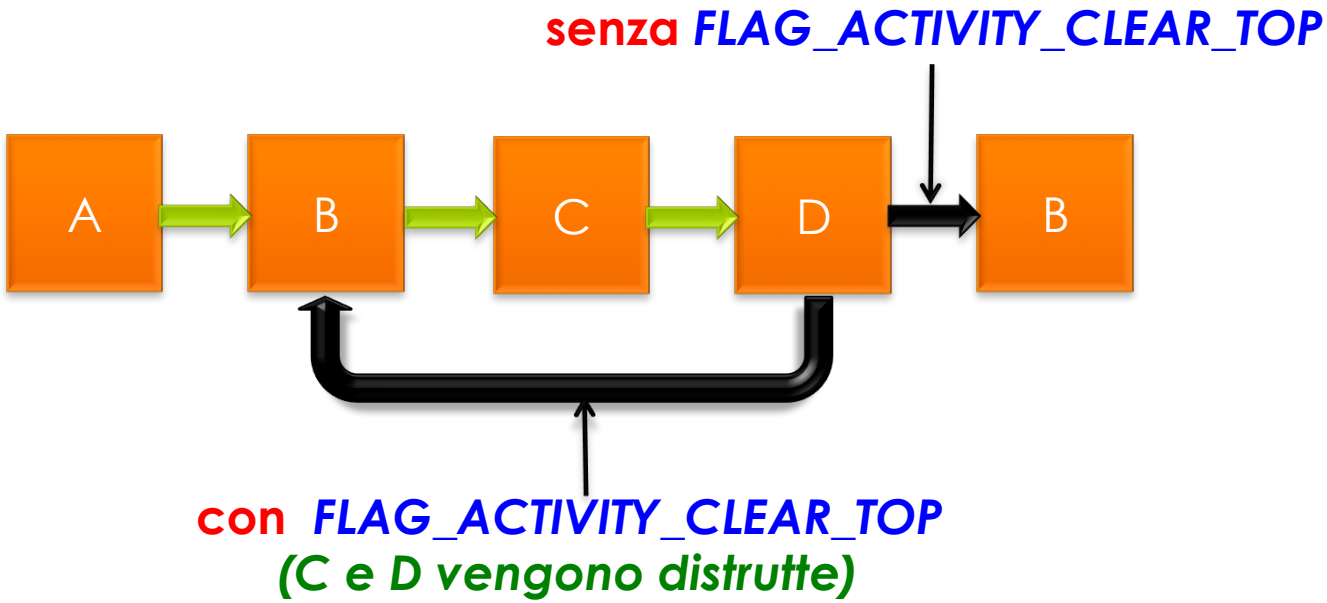
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="activity2HelloString">
            This is Activity TWO</string>
   <string name="app_name">
            ButtonActivatedActions</string>
   <string name="activity1HelloString">
            This is Activity ONE</string>
   <string name="button1Label">
            Call Activity TWO</string>
   <string name="button2Label">
            Call Activity ONE</string>
   <string name="instanceCount">
            Instance count: field not initialized</string>
   <string name="instanceCount2">
            Instance count: field not initialized</string>
</resources>
```

Minimizing instances

```
protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        instances++;
        setContentView(R.layout.layout2);
        tf2 = (TextView) findViewById(R.id.instanceCount2);
        final Button button = (Button) findViewById(R.id.button2);
        final Intent intent = new Intent(this, A1.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                startActivity(intent);
            }
        });
}
```



FLAG_ACTIVITY_CLEAR_TOP



For details see http://developer.android.com/reference/android/content/Intent.html#FLAG_ACTIVITY_BROUGHT_TO_FRONT

FLAGS

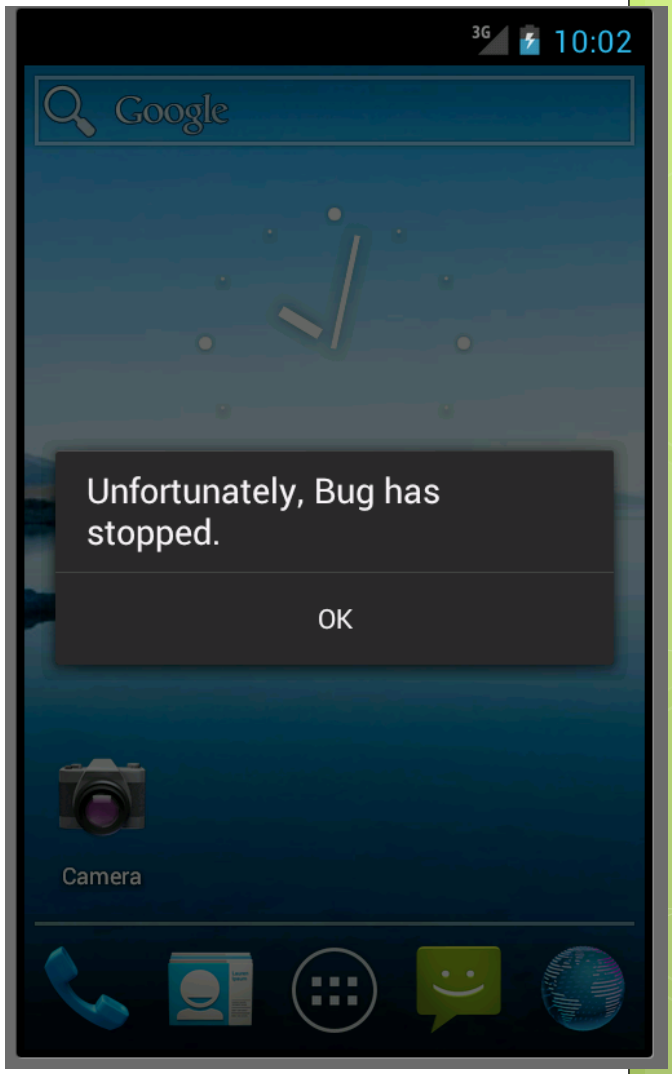
int	FLAG_ACTIVITY_BROUGHT_TO_FRONT	This flag is not normally set by application code, but set for you by the system as described in the `launc`
int	FLAG_ACTIVITY_CLEAR_TASK	If set in an Intent passed to `Context.startActivity()`, this flag will cause any existing task that wo
int	FLAG_ACTIVITY_CLEAR_TOP	If set, and the activity being launched is already running in the current task, then instead of launching a r delivered to the (now on top) old activity as a new Intent.
int	FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET	If set, this marks a point in the task's activity stack that should be cleared when the task is reset.
int	FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS	If set, the new activity is not kept in the list of recently launched activities.
int	FLAG_ACTIVITY_FORWARD_RESULT	If set and this intent is being used to launch a new activity from an existing one, then the reply target of t
int	FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY	This flag is not normally set by application code, but set for you by the system if this activity is being laur
int	FLAG_ACTIVITY_MULTIPLE_TASK	**Do not use this flag unless you are implementing your own top-level application launcher.**
int	FLAG_ACTIVITY_NEW_TASK	If set, this activity will become the start of a new task on this history stack.
int	FLAG_ACTIVITY_NO_ANIMATION	If set in an Intent passed to `Context.startActivity()`, this flag will prevent the system from applyir
int	FLAG_ACTIVITY_NO_HISTORY	If set, the new activity is not kept in the history stack.
int	FLAG_ACTIVITY_NO_USER_ACTION	If set, this flag will prevent the normal `onUserLeaveHint()` callback from occurring on the current fron
int	FLAG_ACTIVITY_PREVIOUS_IS_TOP	If set and this intent is being used to launch a new activity from an existing one, the current activity will n of starting a new one.
int	FLAG_ACTIVITY_REORDER_TO_FRONT	If set in an Intent passed to `Context.startActivity()`, this flag will cause the launched activity to b
int	FLAG_ACTIVITY_RESET_TASK_IF_NEEDED	If set, and this activity is either being started in a new task or bringing to the top an existing task, then it v
int	FLAG_ACTIVITY_SINGLE_TOP	If set, the activity will not be launched if it is already running at the top of the history stack.
int	FLAG_ACTIVITY_TASK_ON_HOME	If set in an Intent passed to `Context.startActivity()`, this flag will cause a newly launching task to
int	FLAG_DEBUG_LOG_RESOLUTION	A flag you can enable for debugging: when set, log messages will be printed during the resolution of this
int	FLAG_EXCLUDE_STOPPED_PACKAGES	If set, this intent will not match any components in packages that are currently stopped.
int	FLAG_FROM_BACKGROUND	Can be set by the caller to indicate that this Intent is coming from a background operation, not from direc
int	FLAG_GRANT_READ_URI_PERMISSION	If set, the recipient of this Intent will be granted permission to perform read operations on the Uri in the In
int	FLAG_GRANT_WRITE_URI_PERMISSION	If set, the recipient of this Intent will be granted permission to perform write operations on the Uri in the I
int	FLAG_INCLUDE_STOPPED_PACKAGES	If set, this intent will always match any components in packages that are currently stopped.
int	FLAG_RECEIVER_REGISTERED_ONLY	If set, when sending a broadcast only registered receivers will be called -- no BroadcastReceiver compo
int	FLAG_RECEIVER_REPLACE_PENDING	If set, when sending a broadcast the new broadcast will replace any existing pending broadcast that mat

Basic tips: having troubles…

Marco Ronchetti
Università degli Studi di Trento

A bugged program

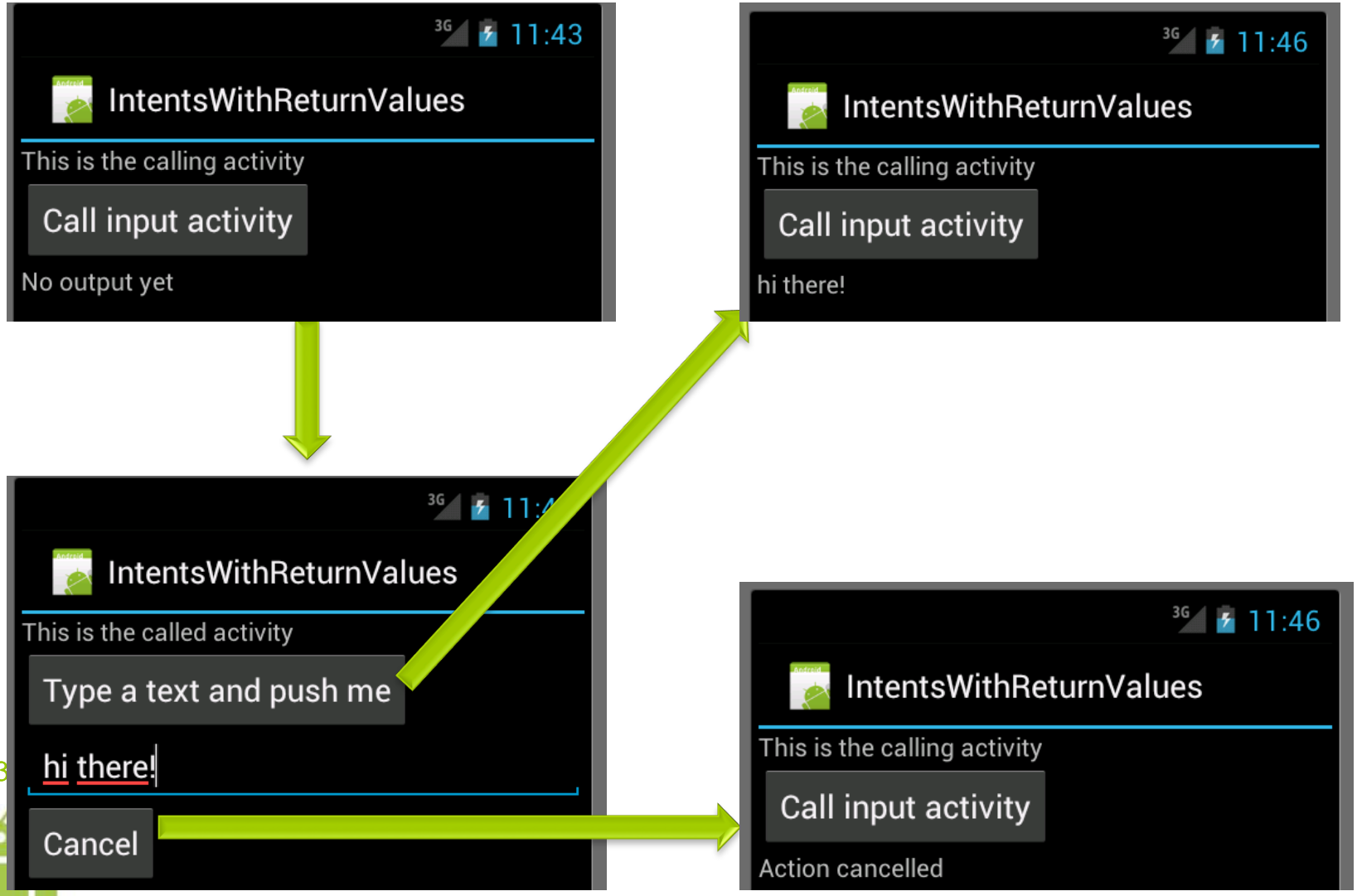
```
package com.example.helloandroid;

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);
    }
}
```



Calling Activities with return values: startActivityForResults

Marco Ronchetti
Università degli Studi di Trento

Returning data



How the dialog occours

Caller:

```
startActivityForResult(Intent in1, int requestCode)
```

Responder:

```
setResult(int resultCode, Intent data1);
data1.putExtra("payloadIdentifier", content);
finish();
```

Caller:

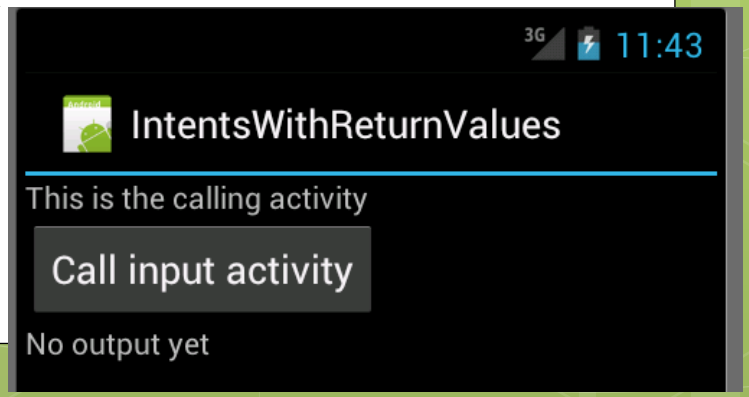
```
onActivityResult(int requestCode, int resultCode,
Intent data2) {
        data2.getStringExtra("payloadIdentifier")
```


Calling activity

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

import ...
```

```
public class CallingActivity extends Activity {
   int requestCode=100;
   public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.layout1);
     final Intent i = new Intent(this,CalledActivity.class);
     final Button button = (Button) findViewById(R.id.invokebutton);
     button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) { startActivityForResult(i, requestCode); }
                });
  }

  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
     super.onActivityResult(requestCode, resultCode, data);
     final TextView tf = (TextView) findViewById(R.id.output);
     if (resultCode==1){
             tf.setText(data.getStringExtra("payload")); }
     else{
              tf.setText("Action cancelled");
     }
  }
}
```

The name should be qualified with the package

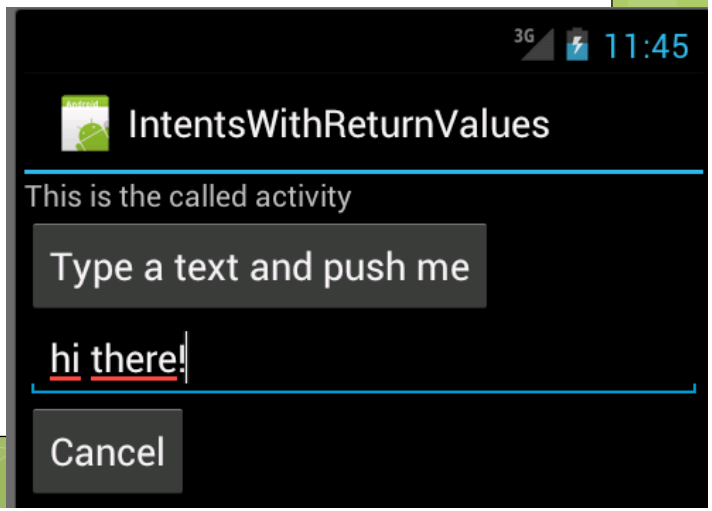


Called activity

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

import ...
```

```
public class CalledActivity extends Activity {
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.layout2);
      final Intent in = new Intent();
      final Button button = (Button) findViewById(R.id.OKbutton);
      final EditText tf = (EditText) findViewById(R.id.editText1);
      button.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                     setResult(1,in);
                     in.putExtra("payload", tf.getText().toString());
                     finish();
                  }
               });
      final Button button_c = (Button) findViewById(R.id.cancelButton);
      button_c.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                     setResult(0,in);
                     finish();
                  }
               });
   }
}
```

The name should be qualified with the package



Remember to register the Activity!

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="it.unitn.science.latemar"
   android:versionCode="1"
   android:versionName="1.0" >
   <uses-sdk android:minSdkVersion="13" />
   <application
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name" >
      <activity
         android:name=".CallingActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity android:name="CalledActivity"></activity>
   </application>
</manifest>
```


Extras

putExtra(String name, #TYPE# payload)
where #TYPE# = one of

- Primitive data types
- String or various types of serializable objects

hasExtra(String name)

get…Extra(String name) where … is the name of the data types available in getters

removeExtra(String name)

replaceExtra completely replace the Extras

putExtra

Intent putExtra(String name, boolean value)
Intent putExtra(String name, boolean[] value)
Intent putExtra(String name, double value)
Intent putExtra(String name, double[] value)
Intent putExtra(String name, byte value)
Intent putExtra(String name, byte[] value)
Intent putExtra(String name, float value)
Intent putExtra(String name, float[] value)
Intent putExtra(String name, int value)
Intent putExtra(String name, int[] value)
Intent putExtra(String name, short value)
Intent putExtra(String name, short[] value)
Intent putExtra(String name, long value)
Intent putExtra(String name, long[] value)
Intent putExtra(String name, char value)
Intent putExtra(String name, char[] value)

Intent putExtra(String name, CharSequence value)
Intent putExtra(String name, CharSequence[] value)
Intent putExtra(String name, String[] value)
Intent putExtra(String name, String value)

Intent putExtra(String name, Bundle value)
Intent putExtra(String name, Serializable value)
Intent putExtra(String name, Parcelable value)
Intent putExtra(String name, Parcelable[] value)

putExtras

Intent putExtras(Intent src)

Copy all extras in 'src' in to this intent.

Intent putExtras(Bundle extras)

Add a set of extended data to the intent.

getExtra:

boolean getbooleanExtra(String name)

boolean[] getbooleanArrayExtra (String name)

… same for double, byte, float, int, short, long, char, CharSequence, String, Parcelable

Bundle getBundleExtra(String name)

Serializable getSerializableExtra(String name)