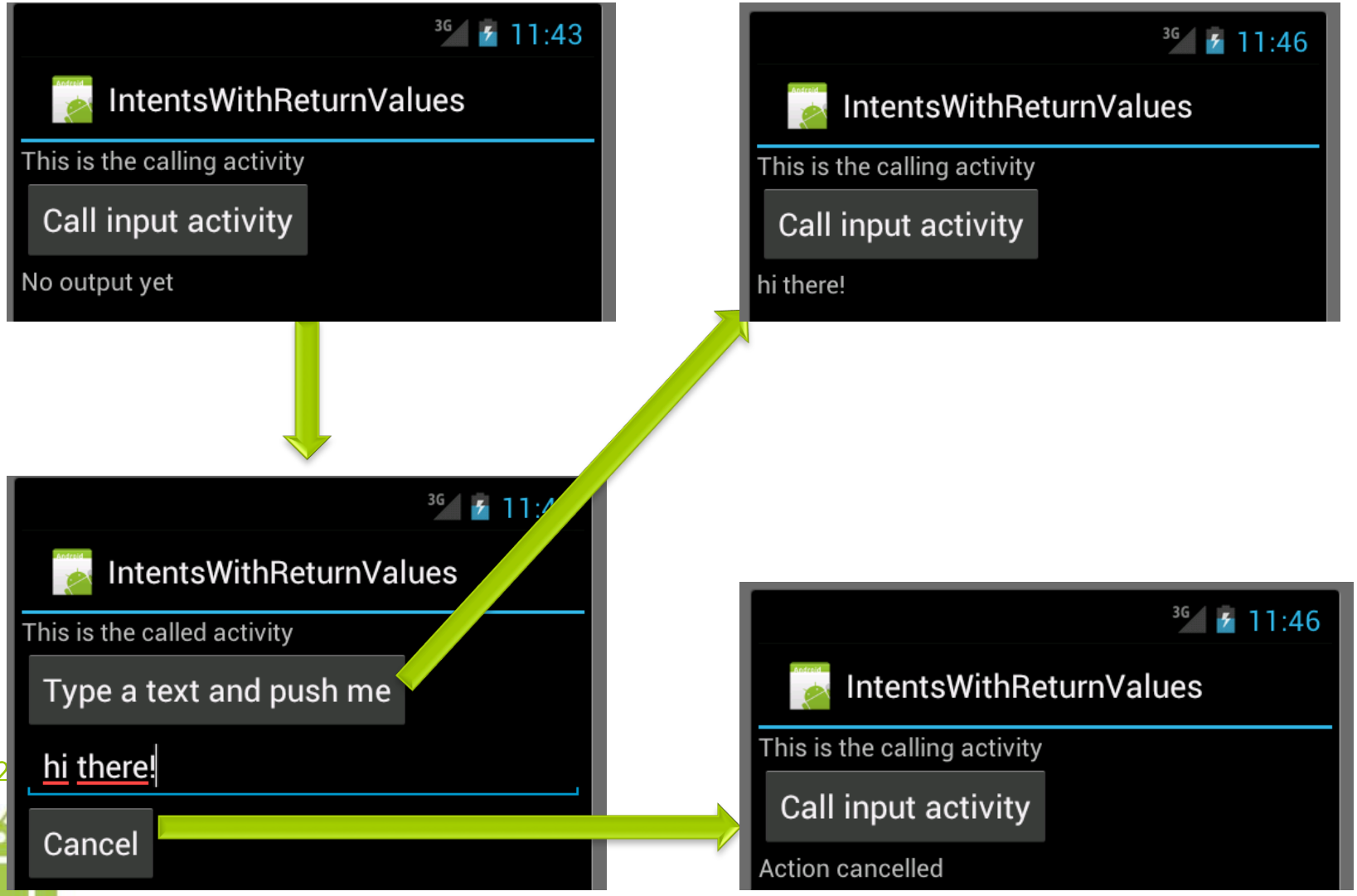
Calling Activities with return values: startActivityForResults

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

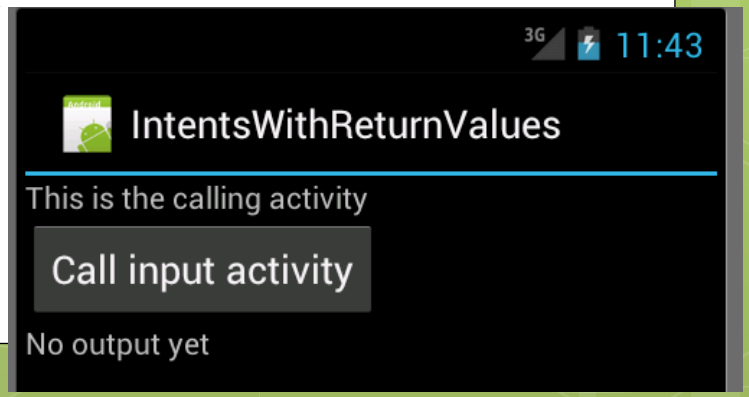
```
package it.unitn.science.latemar;

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



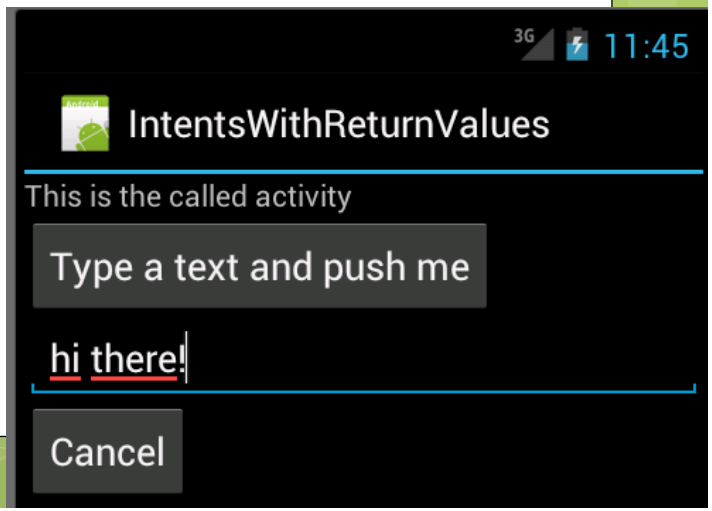
Called activity

```
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.

Basic UI elements: Views and Layouts, a deeper insight

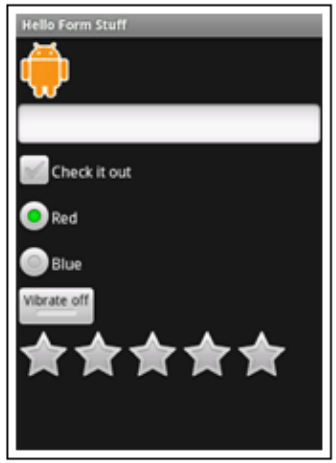
Marco Ronchetti
Università degli Studi di Trento

Widgets examples

Date Picker

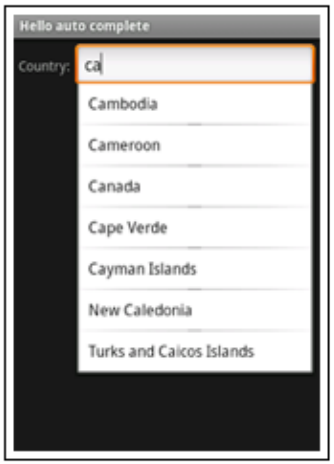
Time Picker

Form Stuff

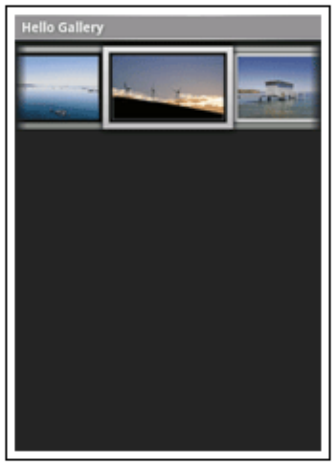


Spinner

Auto Complete



Gallery



Google Map View

Web View

See http://developer.android.com/guide/topics/ui/controls.html

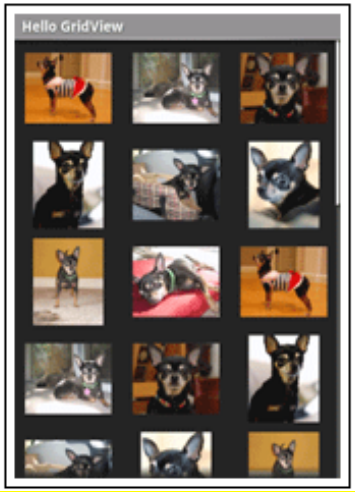
Layout examples

Linear Layout

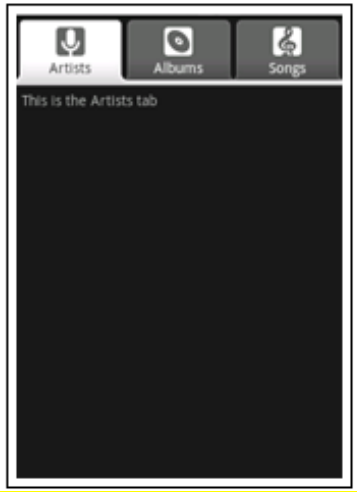
Relative Layout

Table Layout

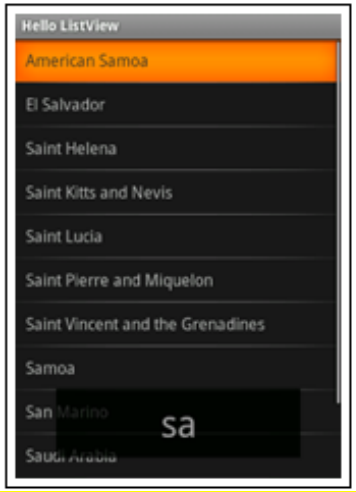
Grid View



Tab Layout



List View



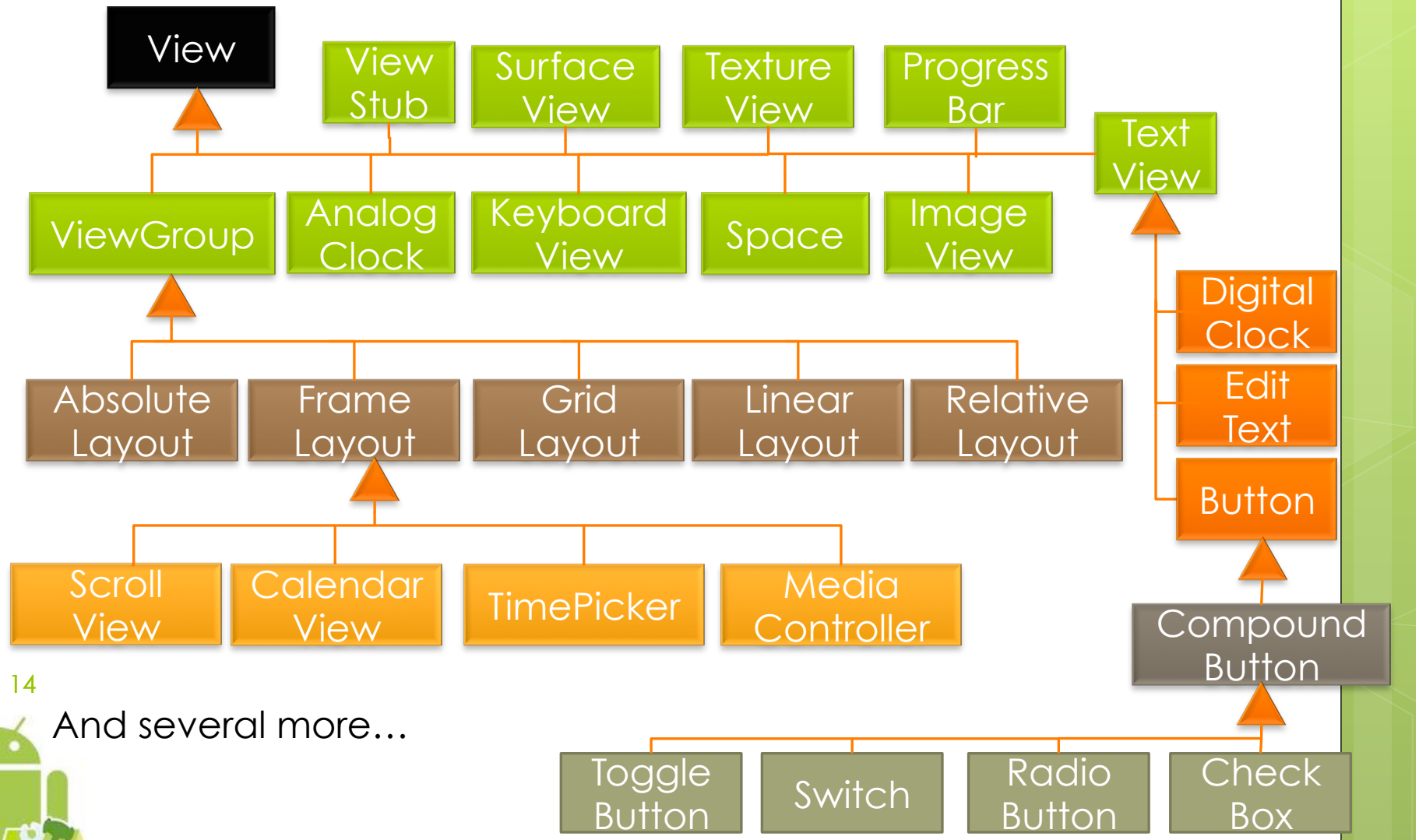
See http://developer.android.com/guide/topics/ui/declaring-layout.html

Defining layouts in XML

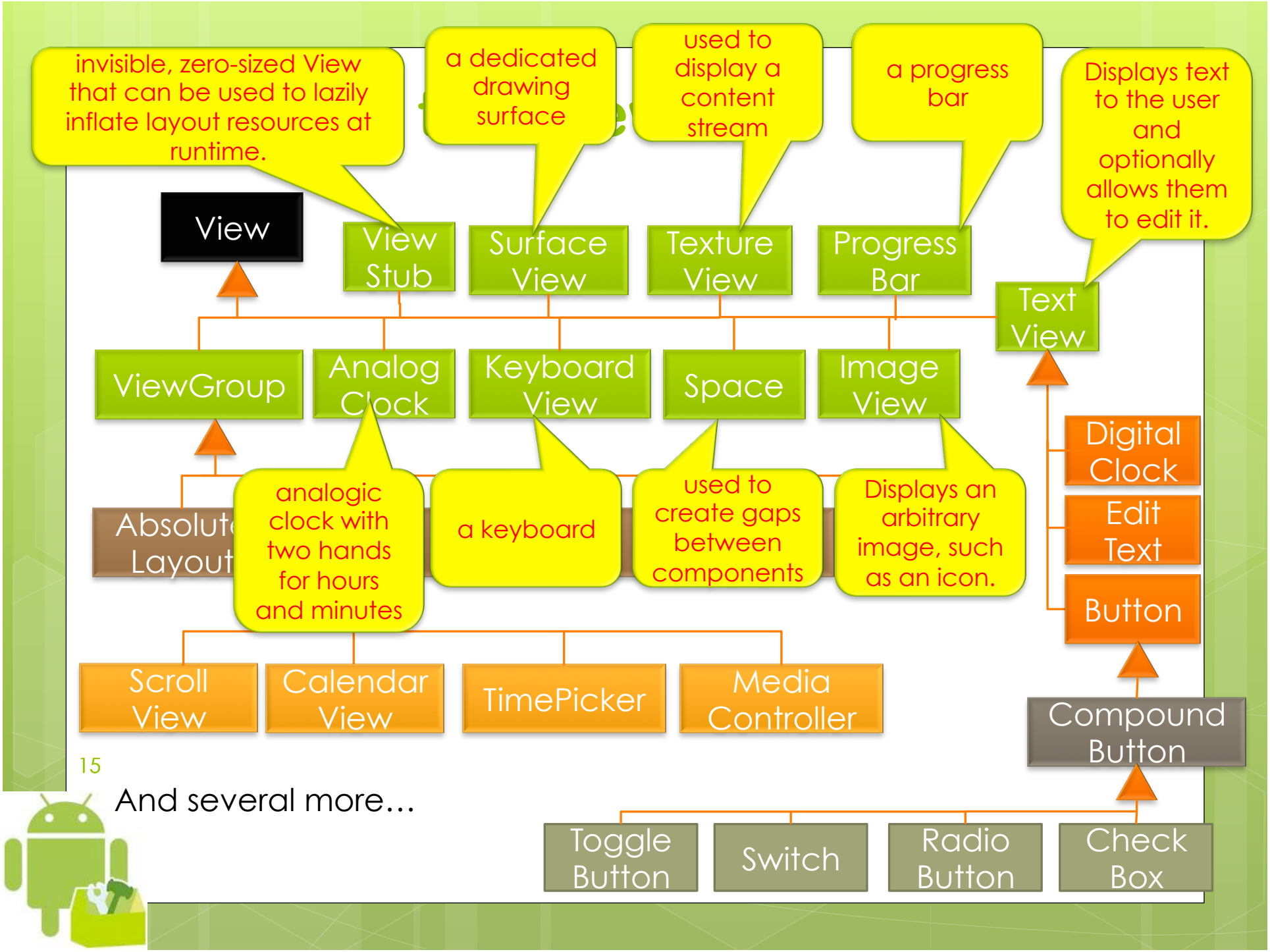
Each layout file must contain exactly one root element, which must be a View or ViewGroup object.

Once you've defined the root element, you can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout.

A view of the Views



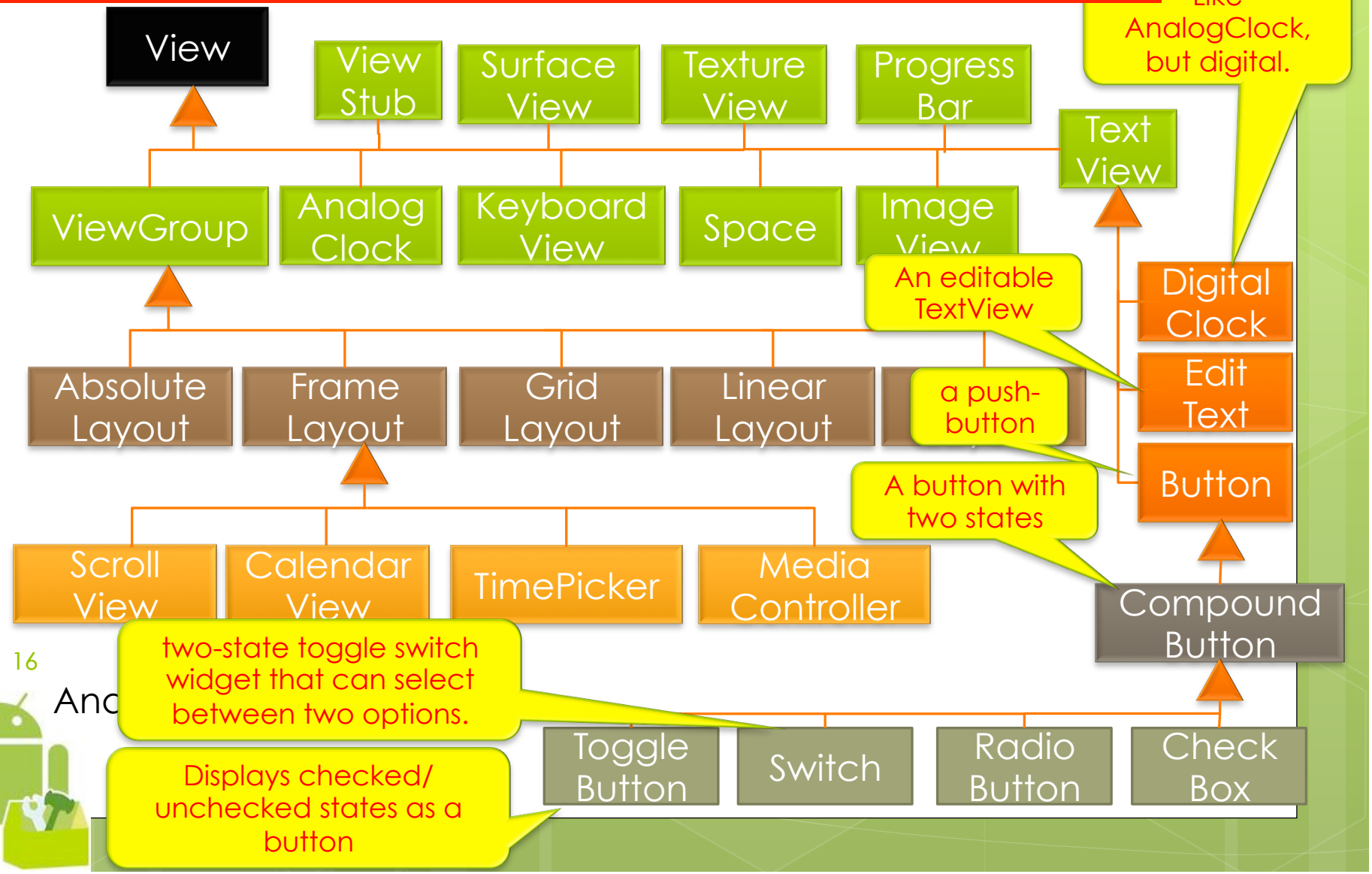
And several more…



And several more…

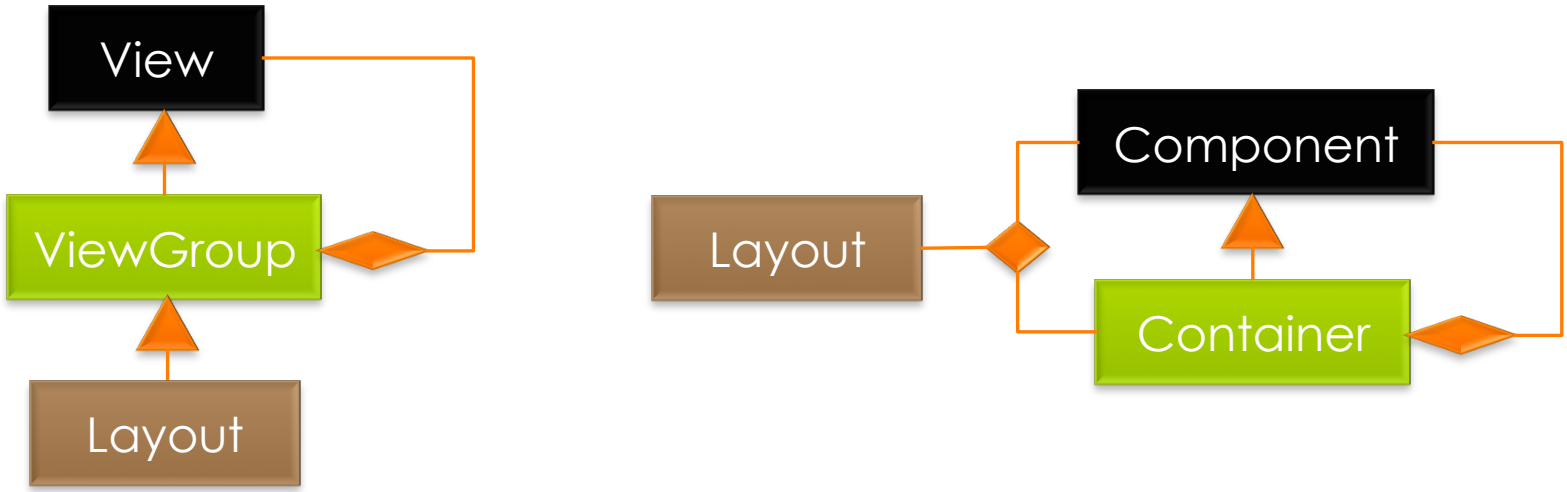
A view of the Views

PLEASE READ THIS:
http://developer.android.com/reference/android/widget/package-summary.html

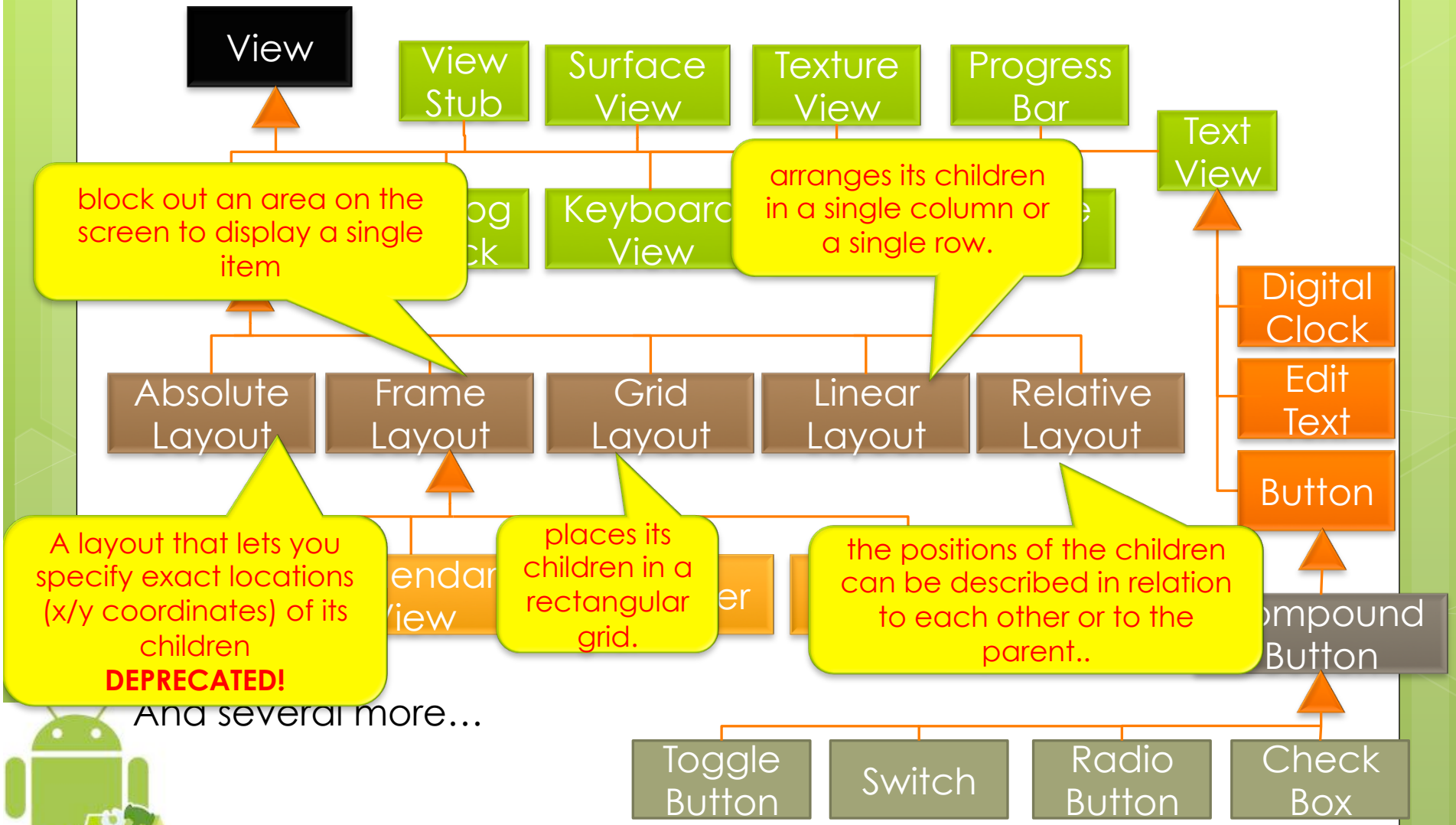


Anc

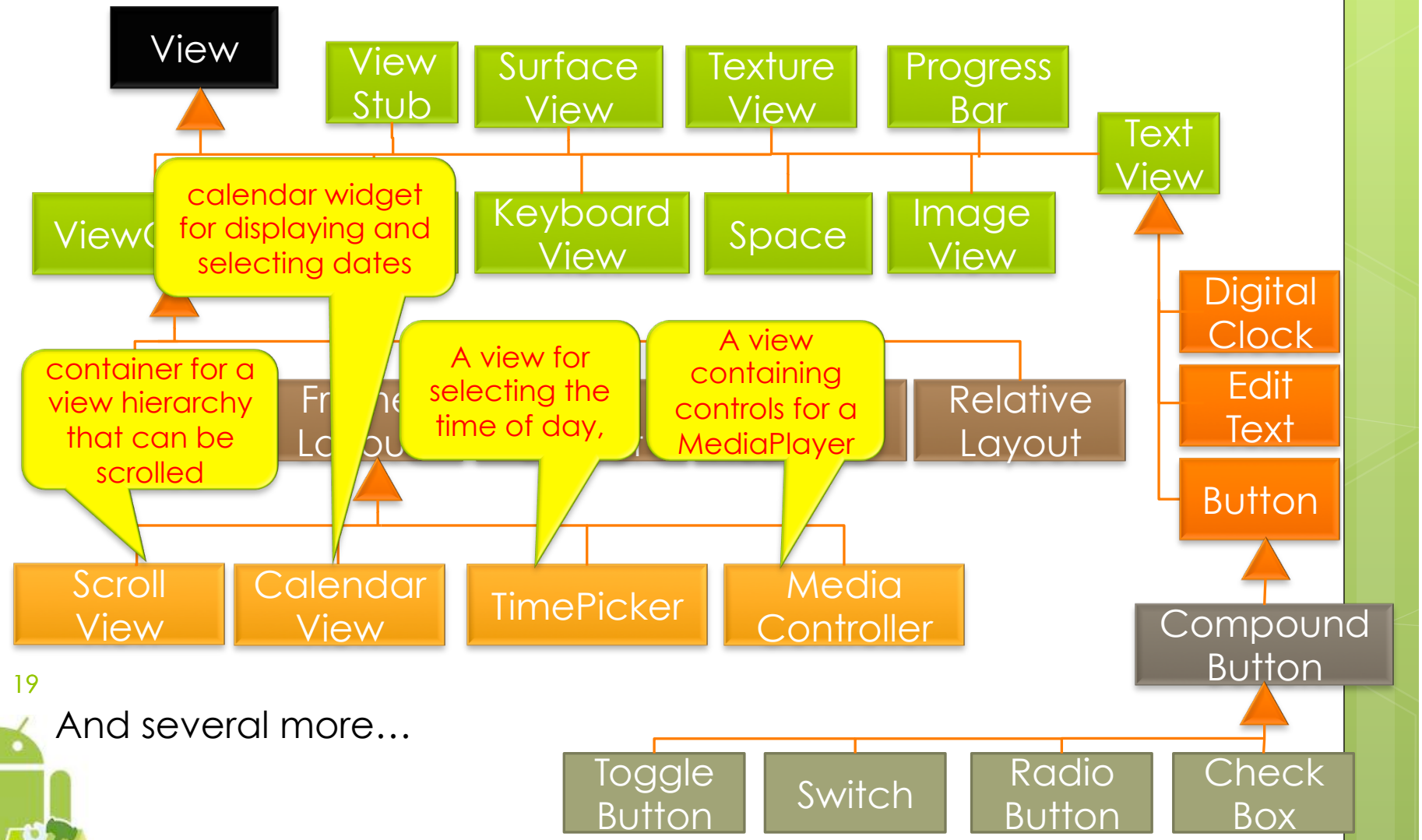
Android vs. Swing architectures



A view of the Views



A view of the Views



And several more…

Layout examples

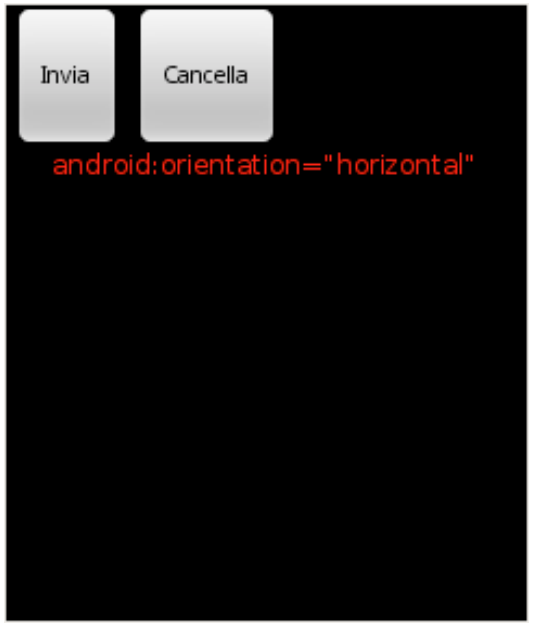
See several examples in

http://luca-petrosino.blogspot.com/2011/03/android-view-e-layout.html

We quickly discuss some of them here.

Layout properties - horizontal

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
    <Button
    android:text="Invia"
    android:id="@+id/Button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
    <Button
    android:text="Cancella"
    android:id="@+id/Button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```

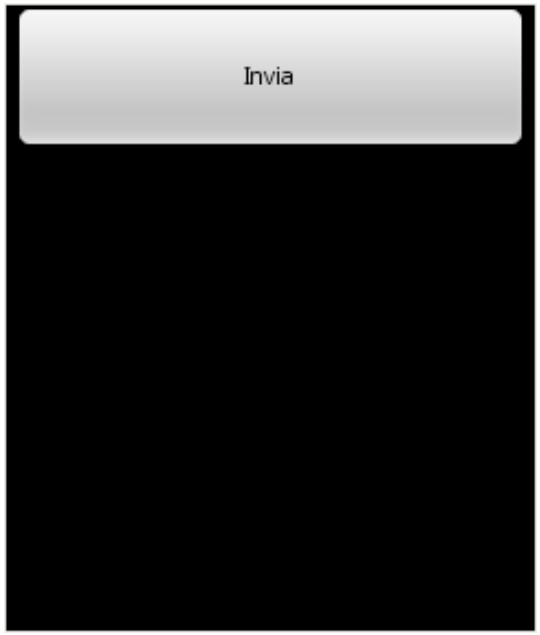


Layout properties - margin

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
    <Button
    android:text="Invia"
    android:id="@+id/Button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="25px"
    android:layout_marginRight="25px"/>
    <Button
    android:text="Cancella"
    android:id="@+id/Button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```

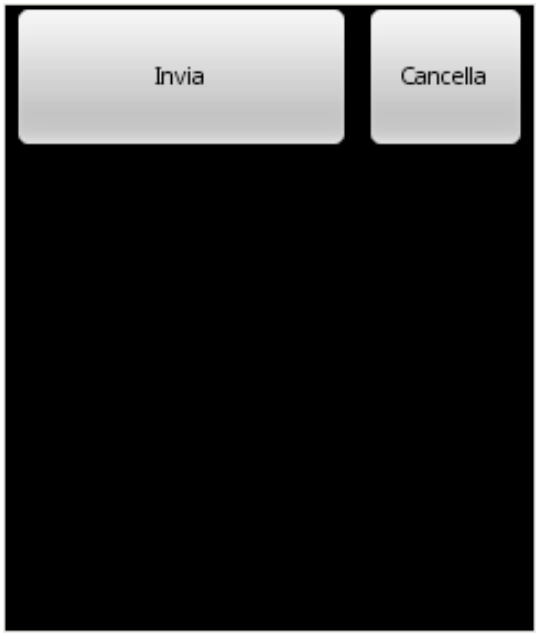
Layout properties – fill_parent

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation=”horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
    <Button
    android:text="Invia"
    android:id="@+id/Button1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"/>
    <Button
    android:text="Cancella"
    android:id="@+id/Button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```



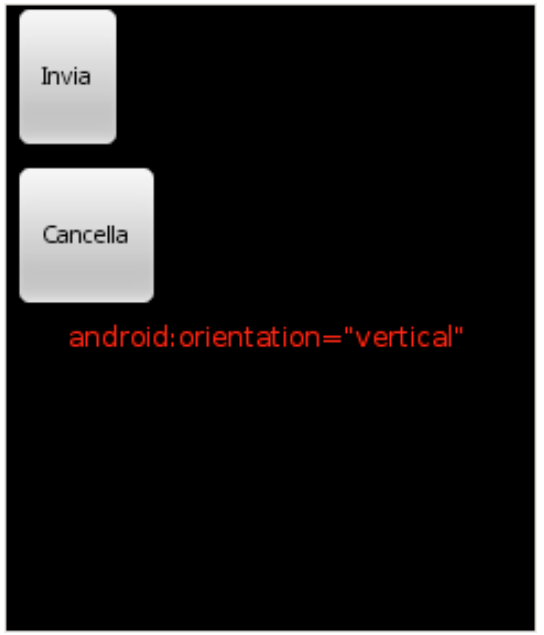
Layout properties – weight

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation=”horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
    <Button
    android:text="Invia"
    android:id="@+id/Button1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content”
    android:layout_weight="1"/>
    <Button
    android:text="Cancella"
    android:id="@+id/Button2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content”
    android:layout_weight=“2"/>
</LinearLayout>
```



Layout properties - vertical

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
    <Button
    android:text="Invia"
    android:id="@+id/Button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
    <Button
    android:text="Cancella"
    android:id="@+id/Button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```



Basic operations on View

- **Set properties:** e.g. setting the text of a TextView (**setText()**). Available properties, getters and setters vary among the different subclasses of views. Properties can also be set in the XML layout files.

- **Set focus:** Focus is moved in response to user input. To force focus to a specific view, call **requestFocus()**.

- **Set up listeners:** e.g. **setOnFocusChangeListener (android.view.View.OnFocusChangeListener)**. View subclasses offer specialized listeners.

- **Set visibility:** You can hide or show views using **setVisibility(int)**. (One of **VISIBLE**, **INVISIBLE**, or **GONE**.)

Invisible, but it still takes up space for layout purposes

Invisible, and it takes no space for layout purposes

Size and location of a View

Although there are setters for position and size, their values are usually controlled (and overwritten) by the Layout.

- **getLeft()**, **getTop()** : get the coordinates of the upper left vertex.
- **getMeasuredHeight()** e **getMeasuredWidth()** return the preferred dimensions
- **getWidth()** e **getHeight()** return the actual dimensions.

Custom views

To implement a custom view, you will usually begin overriding for some of the standard methods that the framework calls on all views (at least onDraw()).

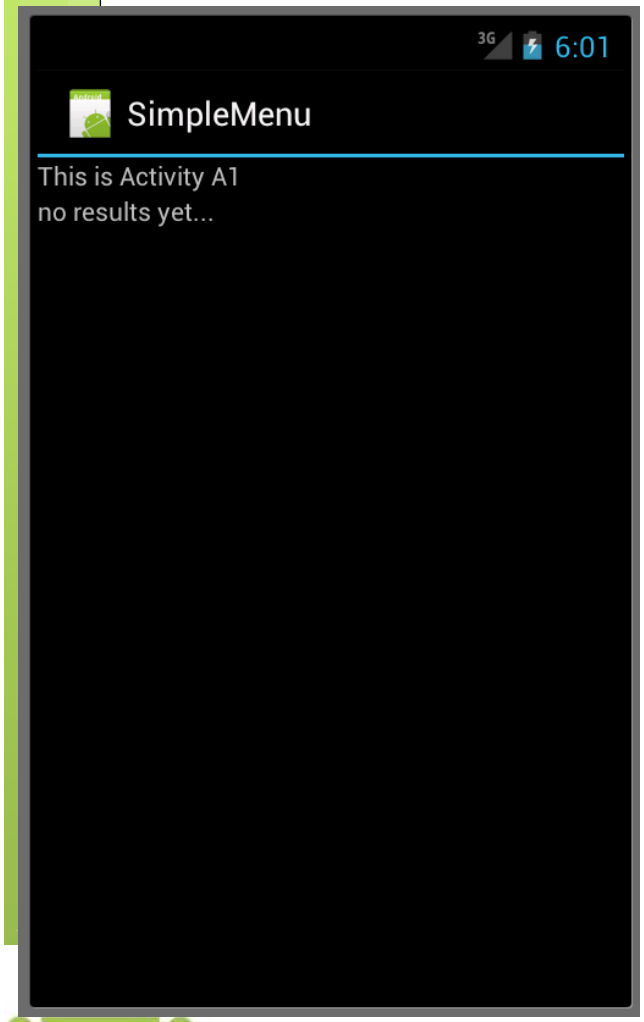
For more details, see

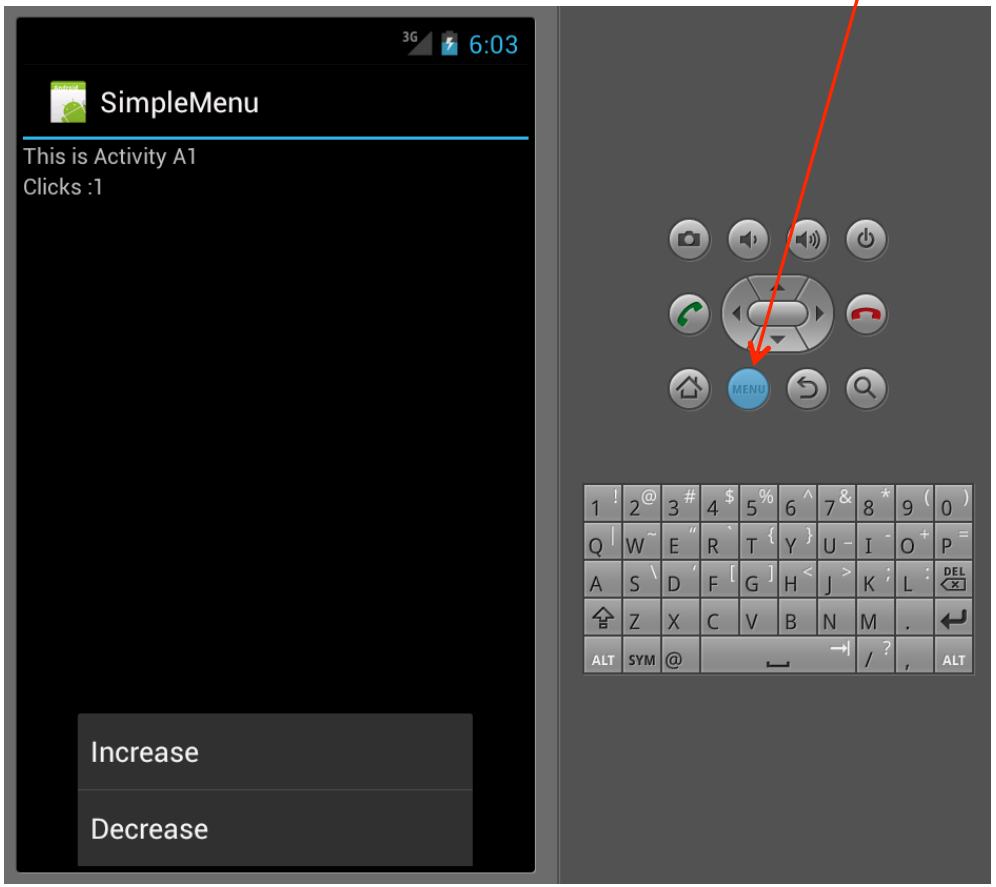
http://developer.android.com/reference/android/view/View.html

Basic UI elements: Android Menus (basics)

Marco Ronchetti
Università degli Studi di Trento

SimpleMenu


6:01
SimpleMenu
This is Activity A1
no results yet...


6:03
SimpleMenu
This is Activity A1
Clicks :1
Increase
Decrease
MENU

Layout & Strings

```
<?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" />
<TextView
      android:id="@+id/tf1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/output" />
</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="hello">This is Activity A1</string>
   <string name="app_name">SimpleMenu</string>
   <string name="output">no results yet...</string>
</resources>
```


SimpleMenu – A1

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

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
public class A1 extends Activity {
    int nClicks=0;
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.layout1);
    }
    public boolean onCreateOptionsMenu(Menu menu){
        super.onCreateOptionsMenu(menu);
        int base=Menu.FIRST;
        MenuItem item1=menu.add(base,1,1,"Increase");
        MenuItem item2=menu.add(base,2,2,"Decrease");
        return true;
    }
    public boolean onOptionsItemSelected(MenuItem item) {
        TextView tf = (TextView) findViewById(R.id.tf1);
        if (item.getItemId()==1) increase();
        else if (item.getItemId()==2) decrease();
        else return super.onOptionsItemSelected(item);
        tf.setText("Clicks :"+nClicks);
        return true;
    }
```

Menu is created

Respond to a Menu event

```
private void increase() {
                nClicks++;
        }
        private void decrease() {
                nClicks--;
        }
}
```

SimpleMenu – A1

```
public class A1 extends Activity {
    int nClicks=0;
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.layout1);
    }
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        int base=Menu.FIRST;
        MenuItem item1=menu.add(base,1,1,"Increase");
        MenuItem item2=menu.add(base,2,2,"Decrease");
        return true;
    }
    public boolean onOptionsItemSelected(MenuItem item) {
        TextView tf = (TextView) findViewById(R.id.tf1);
        if (item.getItemId()==1) increase();
        else if (item.getItemId()==2) decrease();
        else return super.onOptionsItemSelected(item);
        tf.setText("Clicks :"+nClicks);
        return true;
    }
```

ID

Group

Order

Menu is created

This could be a resource

Respond to a Menu event

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

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
    private void increase() {
        nClicks++;
    }
    private void decrease() {
        nClicks--;
    }
}
```

Calling Activities in other apps: Android Intents

Marco Ronchetti
Università degli Studi di Trento

Re-using activities

When you create an application, you can assemble it from

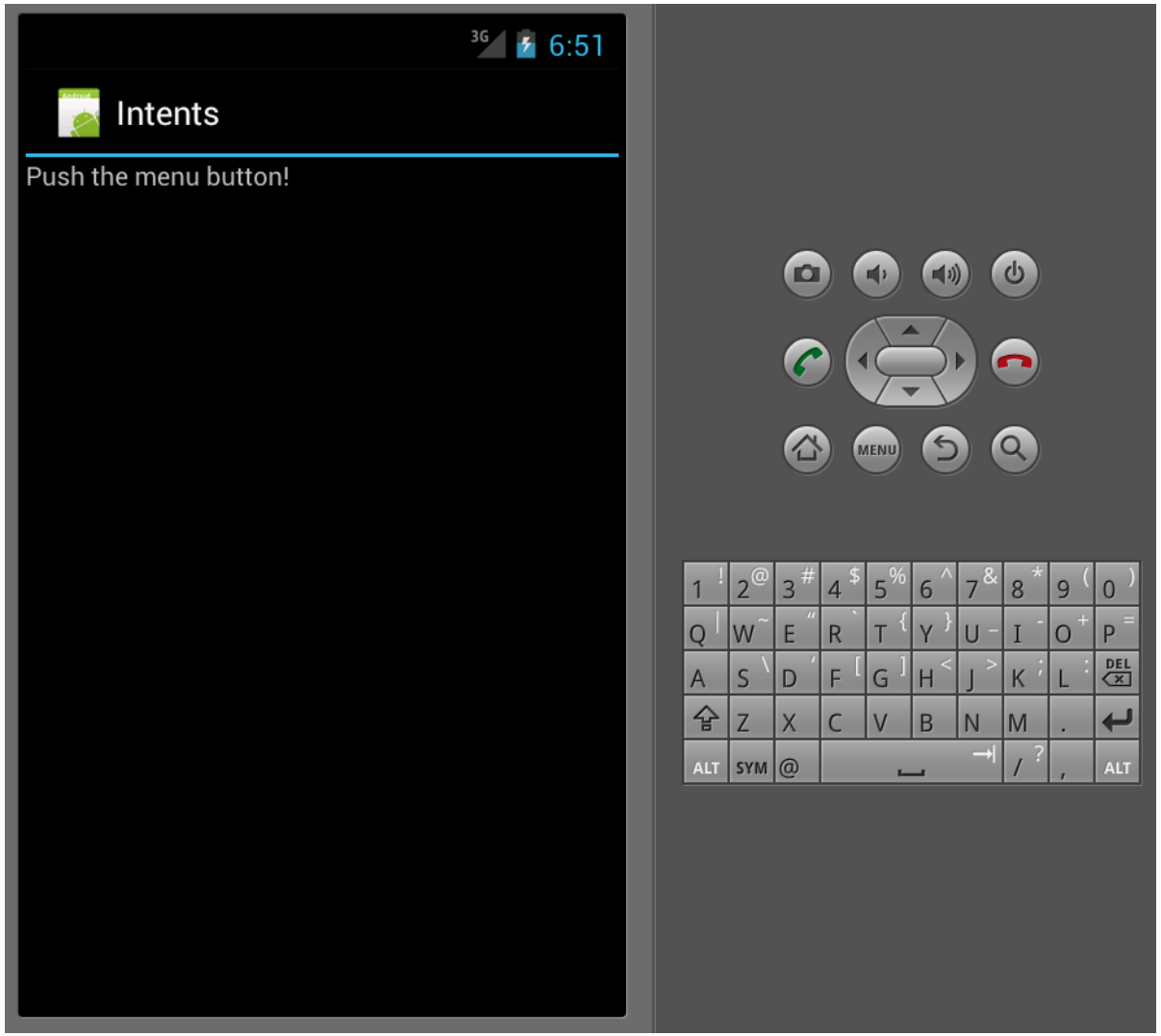
- activities that you create
- activities you re-use from other applications.

An app can incorporate activities from other apps.

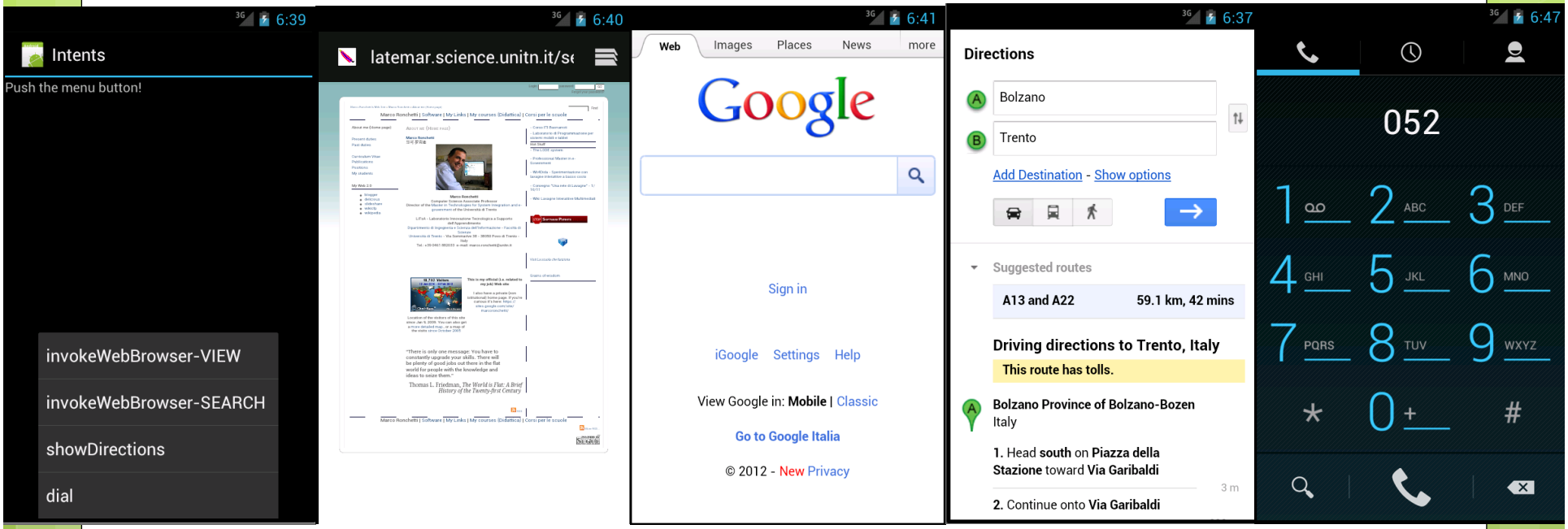
Yes, but how? By means of Intents

These activities are bound at runtime: newly installed applications can take advantage of already installed activities

Our App



Our activities



Our code – IntentUtils - 1

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

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;

public class IntentUtils {

  public static void invokeWebBrowser(Activity activity) {
        Intent intent=new Intent(Intent.ACTION_VIEW);
        intent.setData(Uri.parse("http://latemar.science.unitn.it"));
        activity.startActivity(intent);
  }

  public static void invokeWebSearch(Activity activity) {
        Intent intent=new Intent(Intent.ACTION_WEB_SEARCH,
            Uri.parse("http://www.google.com"));
        activity.startActivity(intent);
  }
```

my definitions

Our code – IntentUtils - 2

my definitions

```
public static void dial(Activity activity) {
        Intent intent=new Intent(Intent.ACTION_DIAL);
        activity.startActivity(intent);
}

public static void showDirections(Activity activity){
        Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
          Uri.parse("http://maps.google.com/maps? saddr=Bolzano&daddr=Trento")
        activity.startActivity(intent);
}
}
```


Our Code: IntentsActivity -1

```
package it.unitn.science.latemar;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

public class IntentsActivity extends Activity {
@Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      TextView tv=new TextView(this);
      tv.setText("Push the menu button!");
      setContentView(tv);
   }

  public boolean onCreateOptionsMenu(Menu menu){
                super.onCreateOptionsMenu(menu);
                int base=Menu.FIRST;
                MenuItem item1=menu.add(base,1,1,"invokeWebBrowser-VIEW");
                MenuItem item2=menu.add(base,2,2,"invokeWebBrowser-SEARCH");
                MenuItem item3=menu.add(base,3,3,"showDirections");
                MenuItem item5=menu.add(base,4,4,"dial");
                return true;
  }
```


Our Code: IntentsActivity -2

```
public boolean onOptionsItemSelected(MenuItem item) {
        System.err.println("item="+item.getItemId());
        if (item.getItemId()==1)
                IntentUtils.invokeWebBrowser(this);
        else if (item.getItemId()==2)
                IntentUtils.invokeWebSearch(this);
        else if (item.getItemId()==3)
                IntentUtils.showDirections(this);
        else if (item.getItemId()==4)
                IntentUtils.dial(this);
        else
                return super.onOptionsItemSelected(item);
        return true;
    }
}
```


Intent structure and resolution

Marco Ronchetti
Università degli Studi di Trento

Intent structure

Who will perform the action?

- **Component name (can be unnamed)**

Which action should be performed?

- **Action identifier (a string)**

Which data should the action act on ?

- **Data** The URI of the data to be acted on

How we classify the action to be performed?

- **Category** A (usually codified) string.

How do we directly pass data?

- **Extras** Key-value pairs for additional information that should be delivered to the component handling the intent

How do we specify behavior modification?

- **Flags** Flags of various sorts.

Examples of action/data pairs

ACTION_VIEW content://contacts/people/1

- Display information about the person whose identifier is "1".

ACTION_DIAL content://contacts/people/1

- Display the phone dialer with the person filled in.

ACTION_VIEW tel:123

- Display the phone dialer with the given number filled in. Note how the VIEW action does what what is considered the most reasonable thing for a particular URI.

ACTION_DIAL tel:123

- Display the phone dialer with the given number filled in.

ACTION_EDIT content://contacts/people/1

- Edit information about the person whose identifier is "1".

ACTION_VIEW content://contacts/people/

- Display a list of people, which the user can browse through.

Standard Actions Identifiers

ACTION_MAIN
ACTION_VIEW
ACTION_ATTACH_DATA
ACTION_EDIT
ACTION_PICK
ACTION_CHOOSER
ACTION_GET_CONTENT
ACTION_DIAL
ACTION_CALL
ACTION_SEND
ACTION_SENDTO
ACTION_ANSWER
ACTION_INSERT
ACTION_DELETE
ACTION_RUN
ACTION_SYNC
ACTION_PICK_ACTIVITY
ACTION_SEARCH
ACTION_WEB_SEARCH
ACTION_FACTORY_TEST

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

Standard Categories

String	CATEGORY_ALTERNATIVE	Set if the activity should be considered as an alternative action to the data the user is currently viewing.
String	CATEGORY_APP_BROWSER	Used with ACTION_MAIN to launch the browser application.
String	CATEGORY_APP_CALCULATOR	Used with ACTION_MAIN to launch the calculator application.
String	CATEGORY_APP_CALENDAR	Used with ACTION_MAIN to launch the calendar application.
String	CATEGORY_APP_CONTACTS	Used with ACTION_MAIN to launch the contacts application.
String	CATEGORY_APP_EMAIL	Used with ACTION_MAIN to launch the email application.
String	CATEGORY_APP_GALLERY	Used with ACTION_MAIN to launch the gallery application.
String	CATEGORY_APP_MAPS	Used with ACTION_MAIN to launch the maps application.
String	CATEGORY_APP_MARKET	This activity allows the user to browse and download new applications.
String	CATEGORY_APP_MESSAGING	Used with ACTION_MAIN to launch the messaging application.
String	CATEGORY_APP_MUSIC	Used with ACTION_MAIN to launch the music application.
String	CATEGORY_BROWSABLE	Activities that can be safely invoked from a browser must support this category.
String	CATEGORY_CAR_DOCK	An activity to run when device is inserted into a car dock.
String	CATEGORY_CAR_MODE	Used to indicate that the activity can be used in a car environment.
String	CATEGORY_DEFAULT	Set if the activity should be an option for the default action (center press) to perform on a piece of data.
String	CATEGORY_DESK_DOCK	An activity to run when device is inserted into a car dock.
String	CATEGORY_DEVELOPMENT_PREFERENCE	This activity is a development preference panel.
String	CATEGORY_EMBED	Capable of running inside a parent activity container.
String	CATEGORY_FRAMEWORK_INSTRUMENTATION_TEST	To be used as code under test for framework instrumentation tests.
String	CATEGORY_HE_DESK_DOCK	An activity to run when device is inserted into a digital (high end) dock.
String	CATEGORY_HOME	This is the home activity, that is the first activity that is displayed when the device boots.
String	CATEGORY_INFO	Provides information about the package it is in; typically used if a package does not contain a CATEGORY_LAUNCHER to provid
String	CATEGORY_LAUNCHER	Should be displayed in the top-level launcher.
String	CATEGORY_LE_DESK_DOCK	An activity to run when device is inserted into a analog (low end) dock.
String	CATEGORY_MONKEY	This activity may be exercised by the monkey or other automated test tools.
String	CATEGORY_OPENABLE	Used to indicate that a GET_CONTENT intent only wants URIs that can be opened with ContentResolver.openInputStream.
String	CATEGORY_PREFERENCE	This activity is a preference panel.
String	CATEGORY_SAMPLE_CODE	To be used as an sample code example (not part of the normal user experience).
String	CATEGORY_SELECTED_ALTERNATIVE	Set if the activity should be considered as an alternative selection action to the data the user has currently selected.
String	CATEGORY_TAB	Intended to be used as a tab inside of an containing TabActivity.
String	CATEGORY_TEST	To be used as a test (not part of the normal user experience).
String	CATEGORY_UNIT_TEST	To be used as a unit test (run through the Test Harness).

Implicit intents and intent resolution

Implicit intents do not name a target (the field for the component name is blank).

In the absence of a designated target, the Android system must find the best component (or components) to handle the intent.

It does so by comparing the contents of the Intent object to *intent filters*, structures associated with components that can potentially receive intents.

Filters advertise the capabilities of a component and delimit the intents it can handle. They open the component to the possibility of receiving implicit intents of the advertised type. If a component does not have any intent filters, it can receive only explicit intents. A component with filters can receive both explicit and implicit intents.

Intent Filters

Only three aspects of an Intent object are consulted when the object is tested against an intent filter:

action
data (both URI and data type)
category

The extras and flags play no part in resolving which component receives an intent.

AndroidManifest.xml

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

    <uses-sdk android:minSdkVersion="15" />

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

</manifest>
```


Intents

Intent messaging is a facility for late run-time binding between components in the same or different applications.

The intent itself, an Intent object, is a passive data structure holding an abstract description of an operation to be performed — or, often in the case of broadcasts, a description of something that has happened and is being announced.

There are separate mechanisms for delivering intents to each type of component.

Using intents with activities

An Intent object is passed to Context.startActivity() or Activity.startActivityForResult() to launch an activity or get an existing activity to do something new. (It can also be passed to Activity.setResult() to return information to the activity that called startActivityForResult().)

Other uses of Intents

An Intent object is passed to Context.startService() to initiate a service or deliver new instructions to an ongoing service. Similarly, an intent can be passed to Context.bindService() to establish a connection between the calling component and a target service. It can optionally initiate the service if it's not already running.

Intent objects passed to any of the broadcast methods (such as Context.sendBroadcast(), Context.sendOrderedBroadcast(), or Context.sendStickyBroadcast()) are delivered to all interested broadcast receivers. Many kinds of broadcasts originate in system code.

In each case, the Android system finds the appropriate activity, service, or set of broadcast receivers to respond to the intent, instantiating them if necessary. There is no overlap within these messaging systems: Broadcast intents are delivered only to broadcast receivers, never to activities or services. An intent passed to startActivity() is delivered only to an activity, never to a service or broadcast receiver, and so on.