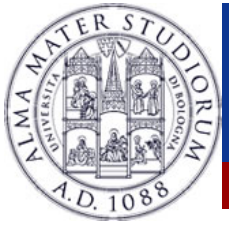
Programming with Android:
Layouts

Luca Bedogni

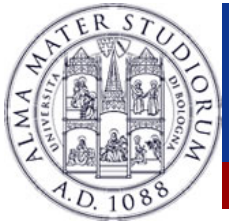
Dipartimento di Informatica: Scienza e Ingegneria

Università di Bologna



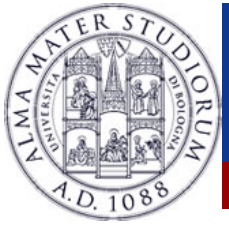
Views: outline

- Main difference between a Drawable and a View is reaction to events
- Could be declared in an XML file
- Could also be declared inside an Activity
- Every view has a unique ID
- Use findViewById(int id) to get it
- Views can be customized



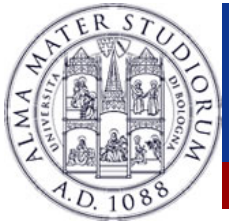
Some useful methods

- getLeft()
- getTop()
- getMeasuredWidth()
- getMeasuredHeight()
- getWidth()
- getHeight()
- requestLayout()
- invalidate()



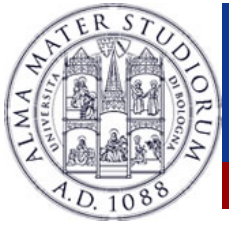
ViewGroup and layout

- ViewGroup is a view container
- It is responsible for placing other views on the display
- Every layout must extend a ViewGroup
- Every view needs to specify:
  - android:layout_height
  - android:layout_width
  - A dimension or one of match_parent or wrap_content



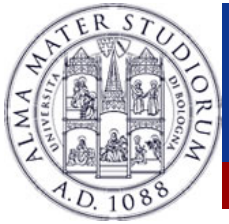
Layouts

- Some layouts are pre-defined by Android
- Some of these are
  - LinearLayout
  - RelativeLayout
  - TableLayout
  - FrameLayout
  - AbsoluteLayout
- A layout could be declared inside another layout



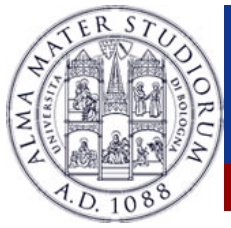
LinearLayout

- Dispose views on a single row or column, depending on android:layout_orientation
- The orientation could also be declared via setOrientation(int orientation)
  - orientation is one of HORIZONTAL or VERTICAL
- Has two other attributes:
  - gravity
  - weight

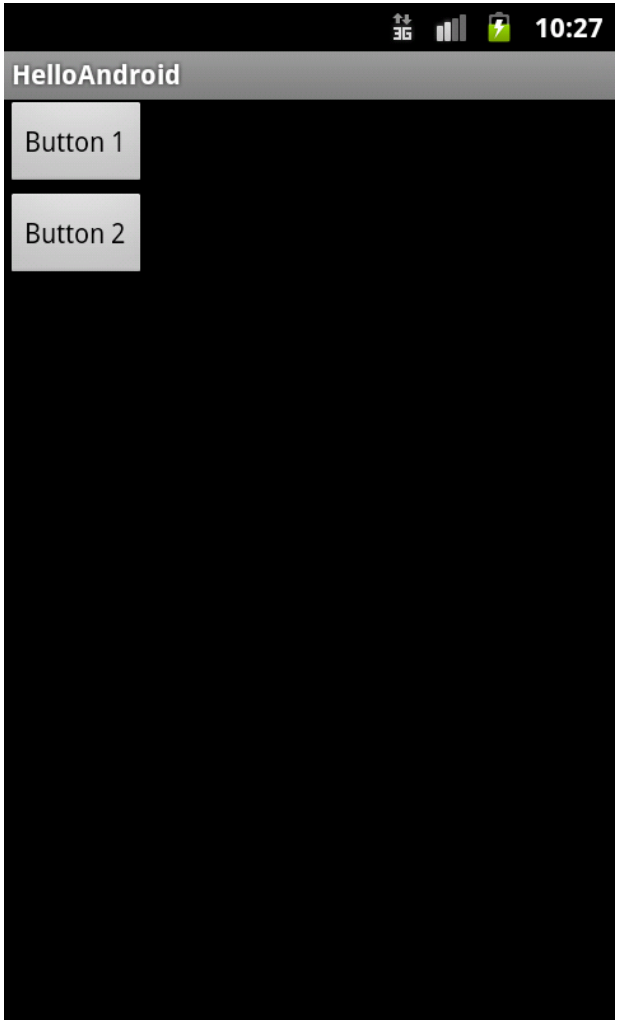


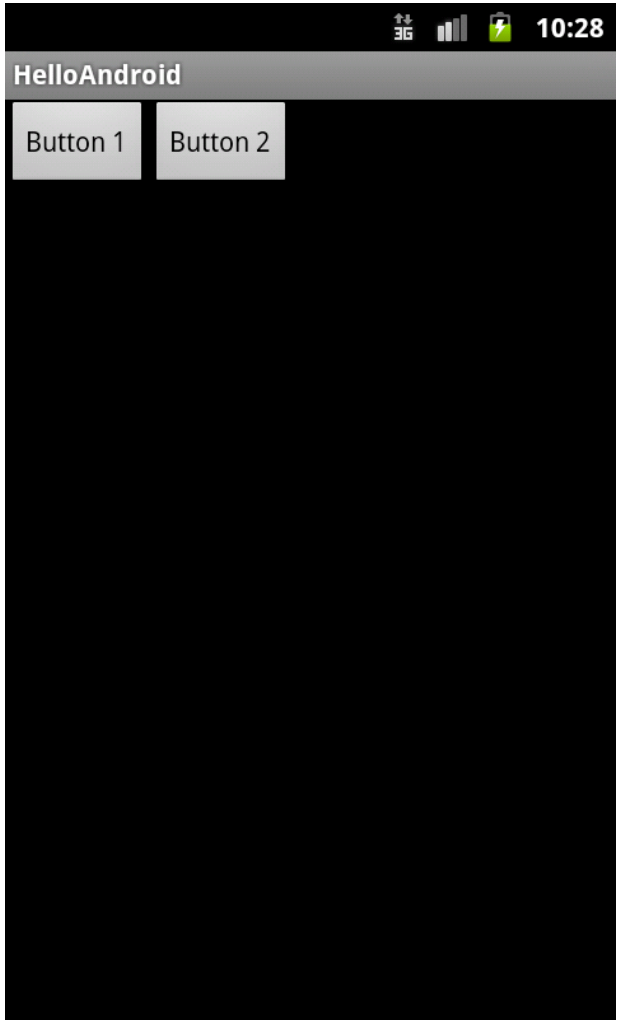
LinearLayout

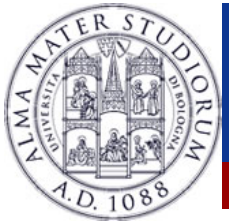
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >                <!-- Also horizontal -->

    <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>
```



LinearLayout

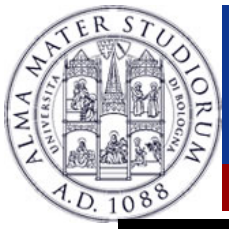




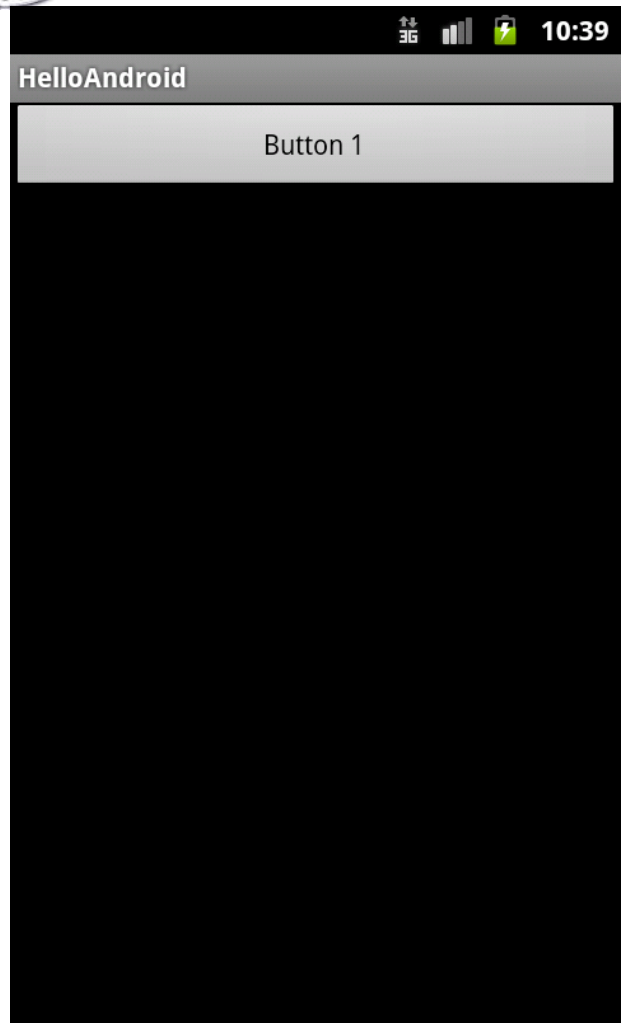


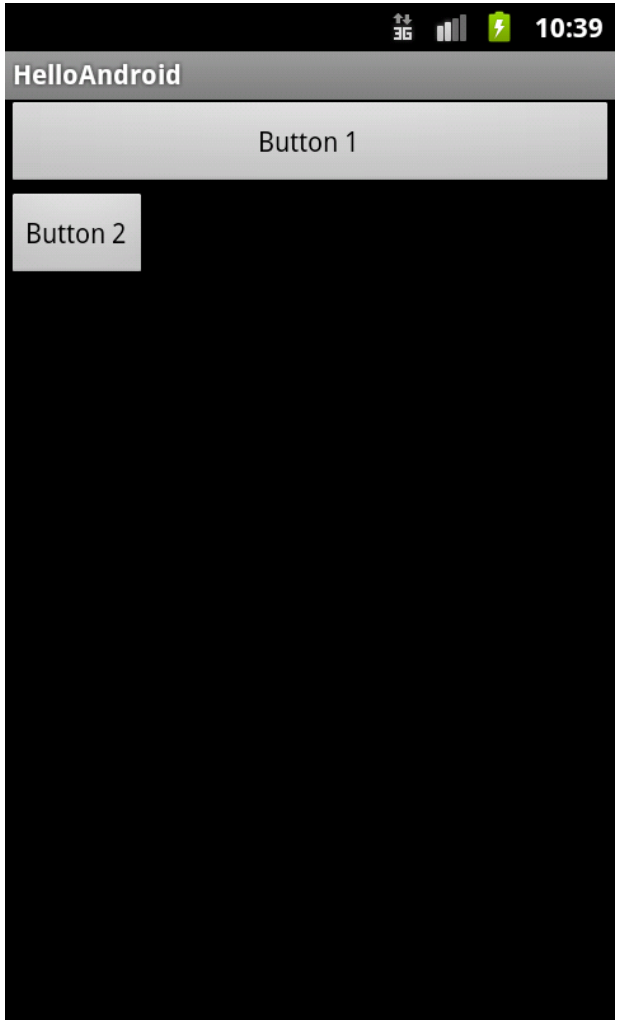
LinearLayout

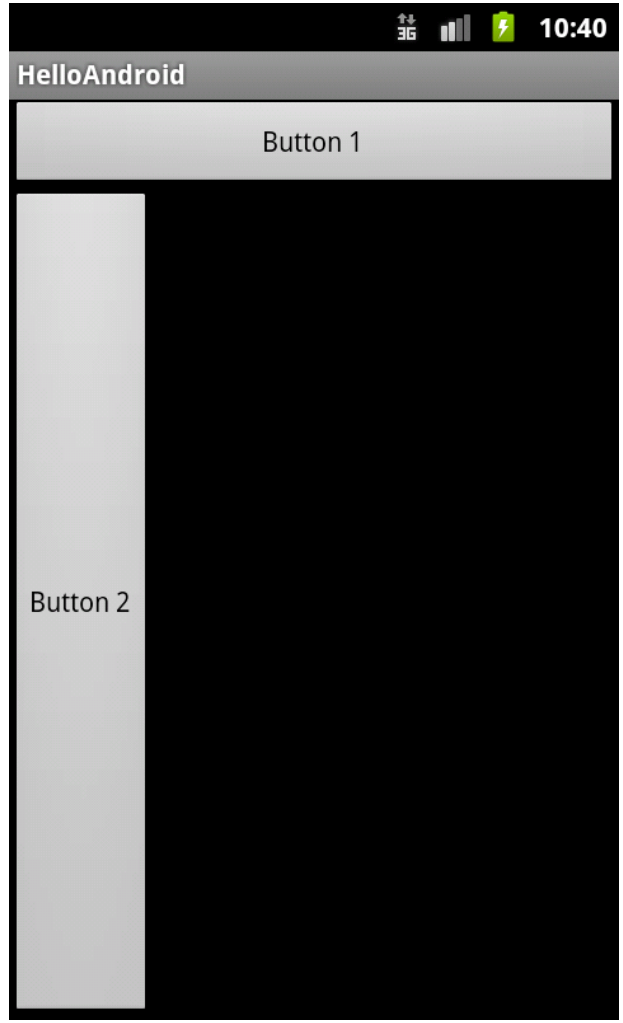
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <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>
```

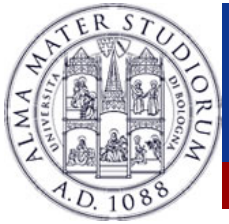


LinearLayout



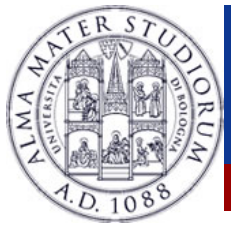




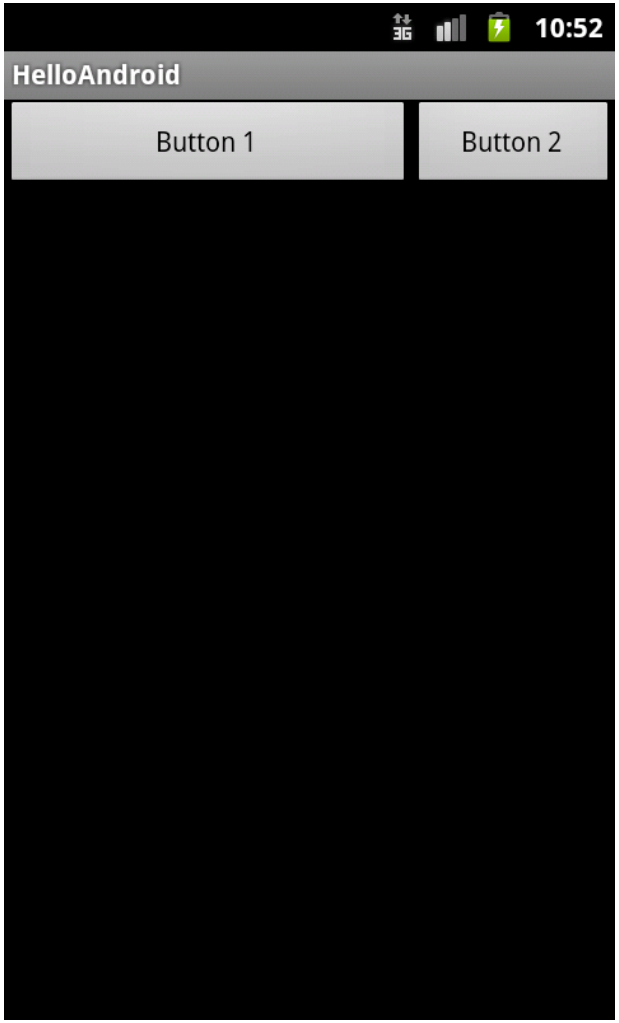


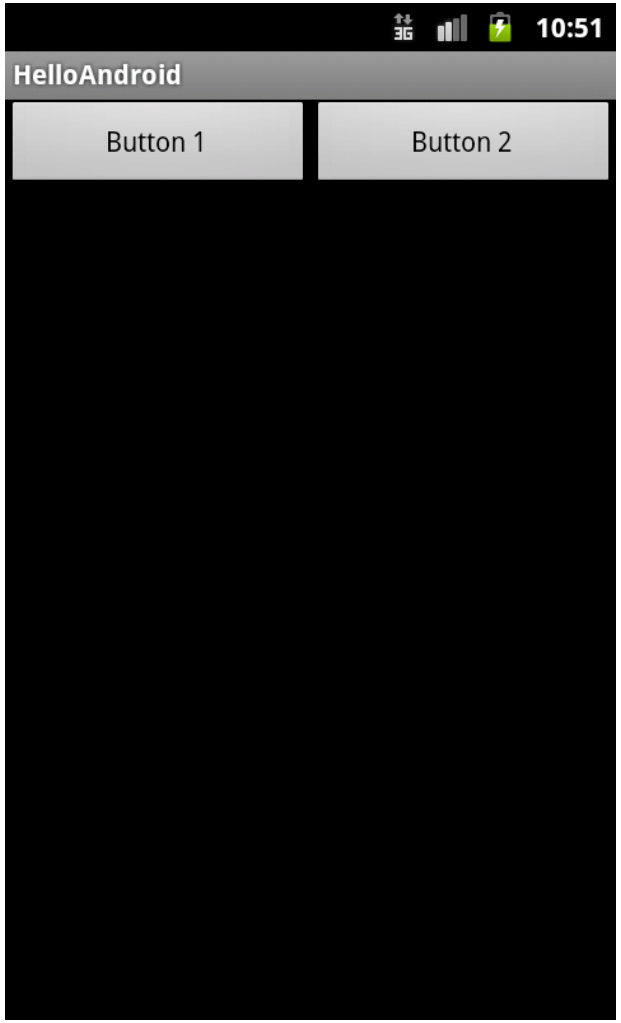
LinearLayout weight

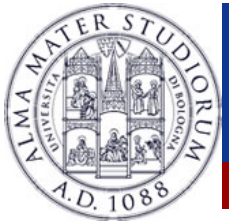
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"    android:layout_height="fill_parent"    android:orientation="horizontal" >

   <Button
      android:id="@+id/button1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="@string/buttonString1"
      android:layout_weight="1" />

   <Button
      android:id="@+id/button2"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="@string/buttonString2"
      android:layout_weight="2"       />
</LinearLayout>
```



LinearLayout weight

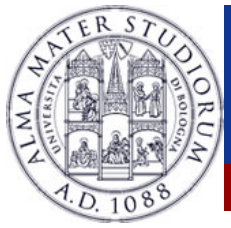




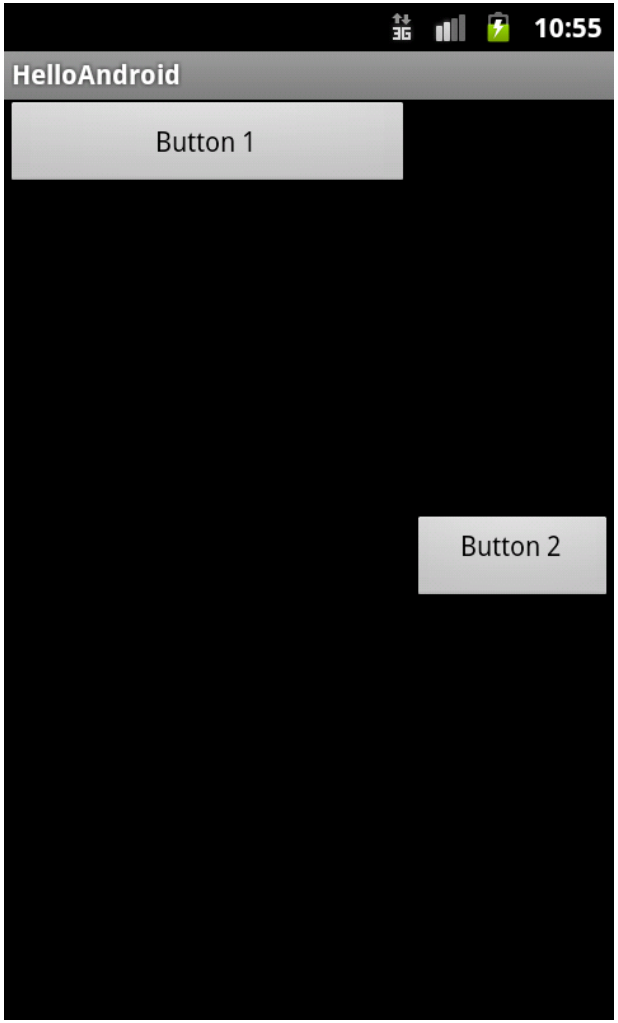


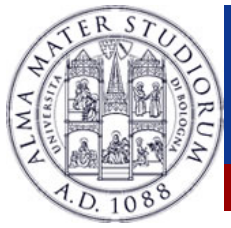
LinearLayout gravity

```
<?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="match_parent"       android:layout_height="wrap_content"
      android:text="@string/buttonString1"
      android:layout_weight="1" />
   <Button
      android:id="@+id/button2"
      android:layout_width="match_parent"       android:layout_height="wrap_content"
      android:text="@string/buttonString2"
      android:layout_weight="2"
      android:layout_gravity="center_vertical"
      android:gravity="top|center" />

</LinearLayout>
```

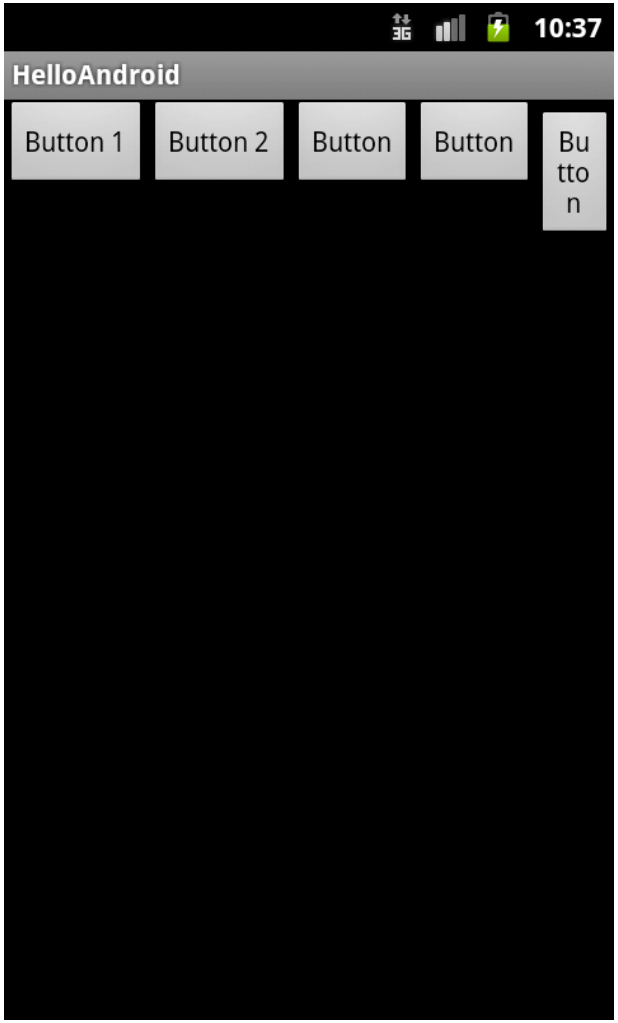


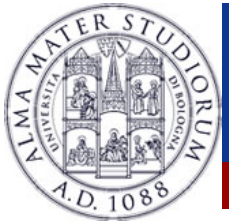
LinearLayout gravity





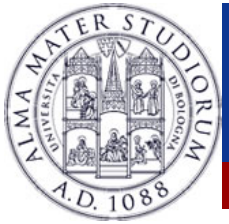
LinearLayout problem





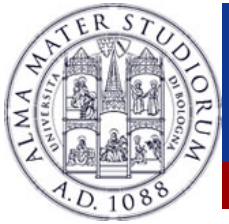
RelativeLayout

- Disposes views according to the container or according to other views
- The gravity attribute indicates what views are more important to define the layout
- Useful to align views



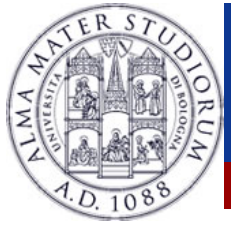
RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"    android:layout_height="match_parent" >

   <EditText
      android:id="@+id/username"        android:text="username"
      android:inputType="text"
      android:layout_width="wrap_content"        android:layout_height="wrap_content"
      android:layout_alignParentRight="true"
      android:layout_toRightOf="@+id/usernameLabel" >
   </EditText>

   <TextView
      android:id="@+id/usernameLabel"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/username"
      android:text="Username" />
```

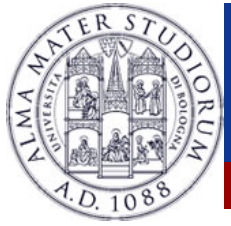


RelativeLayout

```
<EditText
    android:id="@+id/password"        android:text="password"
    android:inputType="textPassword"
    android:layout_below="@+id/username"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/username"
    android:layout_alignParentRight="true"
    android:layout_toRightOf="@+id/usernameLabel" >
  </EditText>

  <TextView
    android:id="@+id/passwordLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/password"
    android:text="Password" />
</RelativeLayout>
```

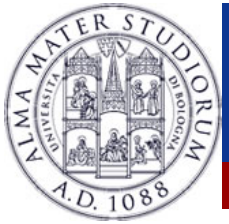


RelativeLayout



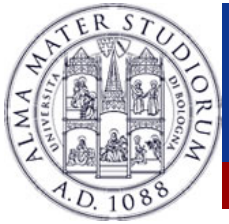
TableLayout

- As the name say, similar to a Table
- Has some attributes to customize the layout:
  - android:layout_column
  - android:layout_span
  - android:stretchColumns
  - android:shrinkColumns
  - android:collapseColumns
- Each row is inside a <TableRow> element



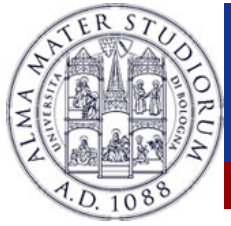
TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout android:layout_width="fill_parent"
    android:layout_height="fill_parent" xmlns:android="http://schemas.android.com/apk/res/android"    android:id="@+id/tableLayout">

  <TableRow android:layout_width="wrap_content" android:layout_height="wrap_content“ android:id="@+id/firstRow">
    <Button     android:id="@+id/button1“
                android:layout_width="wrap_content“
                android:layout_height="wrap_content“
          android:text="Button" />
    <Button android:id="@+id/button2“
          android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Button" />
    <Button android:id="@+id/button3"
          android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:text="Button" />
  </TableRow>
```

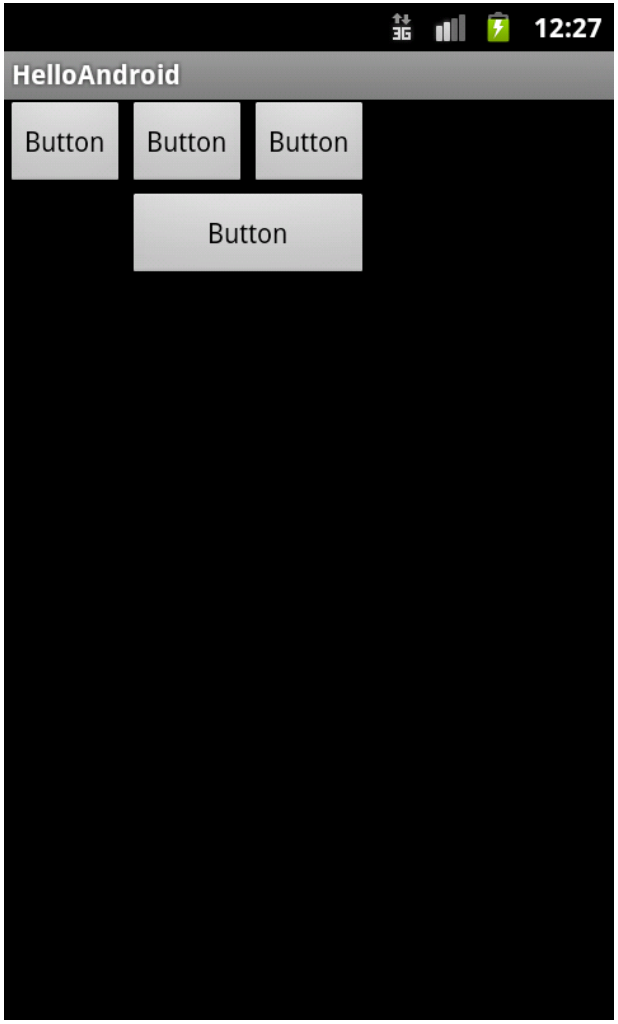


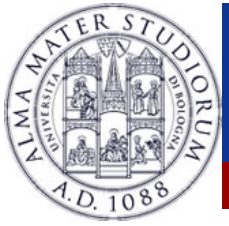
TableLayout

```
<TableRow
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/secondRow">

  <Button  android:layout_column="1"
                android:layout_span="2"
                android:id="@+id/button4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Button">
  </Button>
</TableRow>

</TableLayout>
```



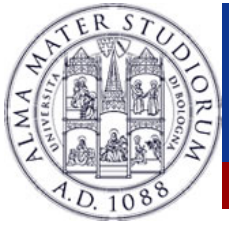
TableLayout





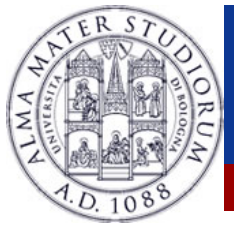
FrameLayout and AbsoluteLayout

- FrameLayout
  - Adds an attribute, android:visibility
  - Makes the user able to define layouts managing the visibility of views
- AbsoluteLayout
  - Deprecated
  - Specify position with x and y
  - Pay attention to different resolutions

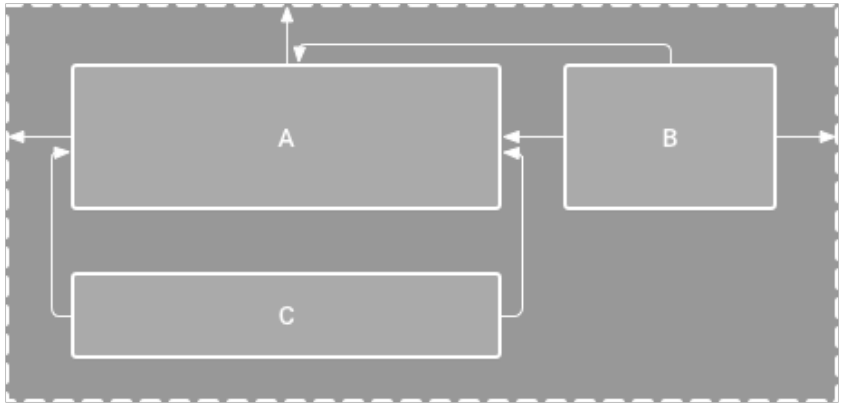


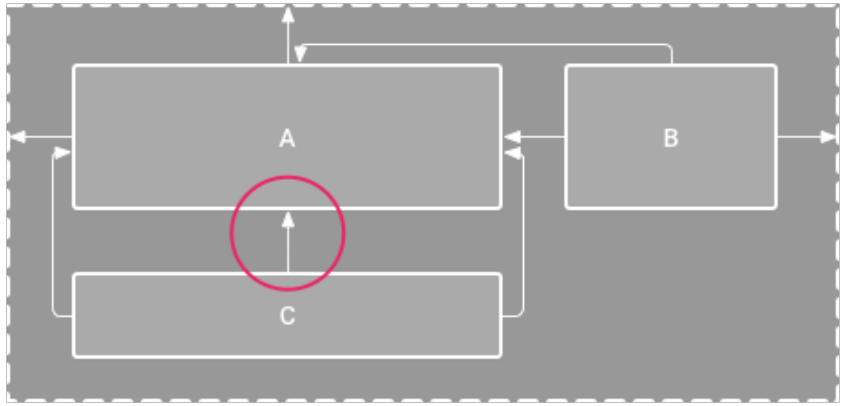
ConstraintLayout

- Flat view hierarchy
- Similar to RelativeLayout
- > Android 2.3
- Overarching idea: define constraints (top/bottom/left/right) for each view
- Each constraint has to be defined to another view, another layout or an invisible guideline

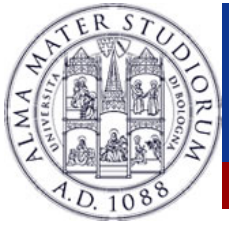


ConstraintLayout: example





- Both layouts are fine
  - The left one has no top constraint on C, which will then be placed at the top



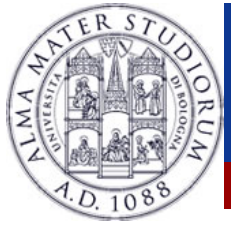
ConstraintLayout: how to use

- Add directions to build.gradle

```
repositories {
    maven {
        url 'https://maven.google.com'
    }
}
```

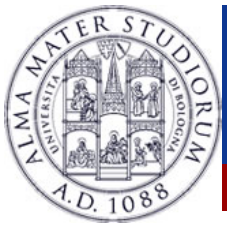
```
dependencies {
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

- Sync the project

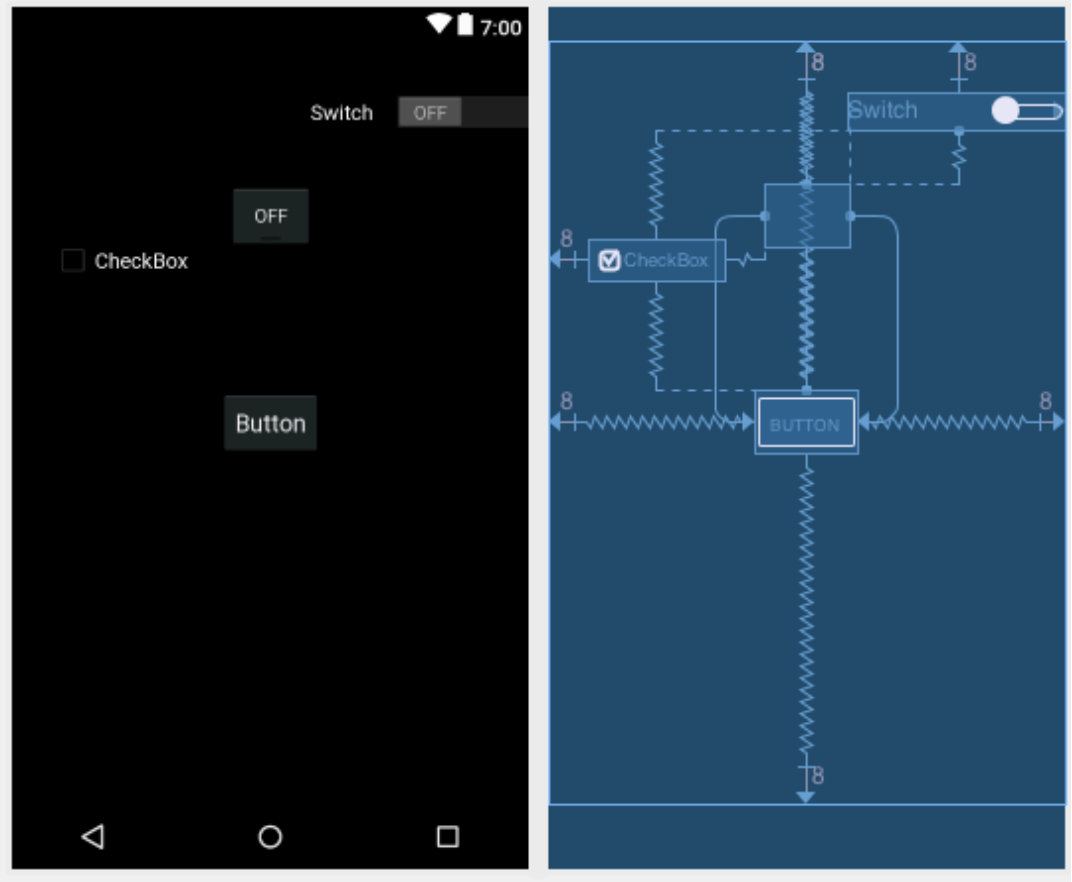


ConstraintLayout: how to create one

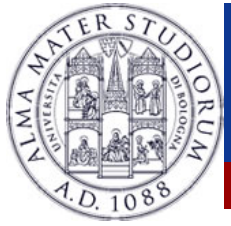
- Converting a Layout
  - Just right click on the layout and select the conversion option
- Creating a ConstraintLayout
  - Create a new layout
  - As root-tag, put ***android.support.constraint.ConstraintLayout***



ConstraintLayout: how to create one



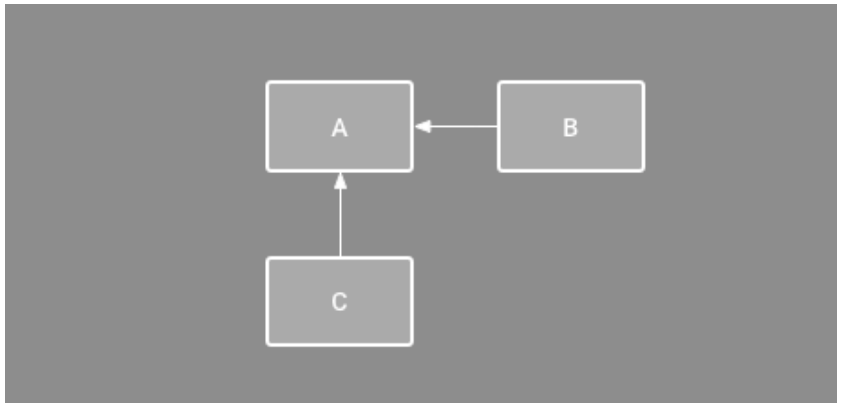
- In the layout editor you'll see on the right the constraints, and on the left a preview
- *Layouts are drawn according to the available space*

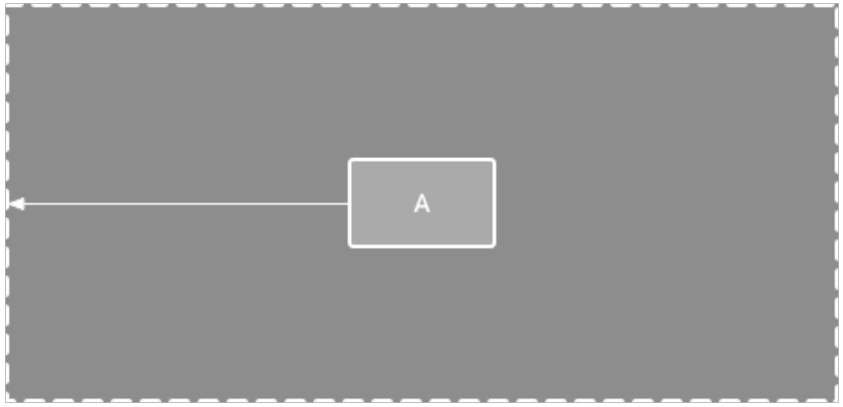


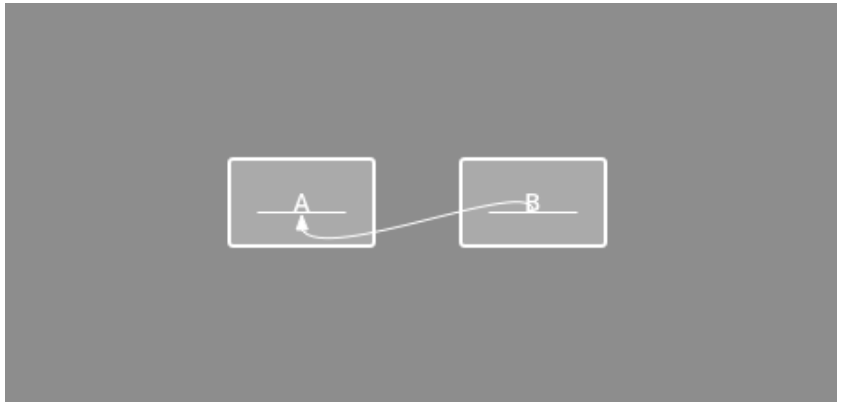
ConstraintLayout: constraints

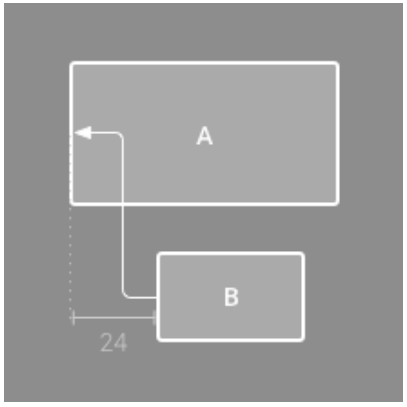
- Each view needs at least one constraint per plane
- Constraints can be defined only between anchor points sharing the same plane
- Each handle can define one constraints
- Multiple handles can define a constraint to a single anchor point
- Adding 2 opposite constraints places the view in the middles

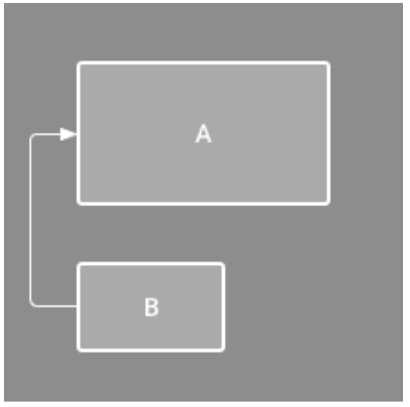
ConstraintLayout: constraints examples

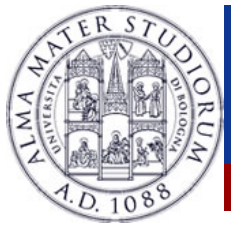




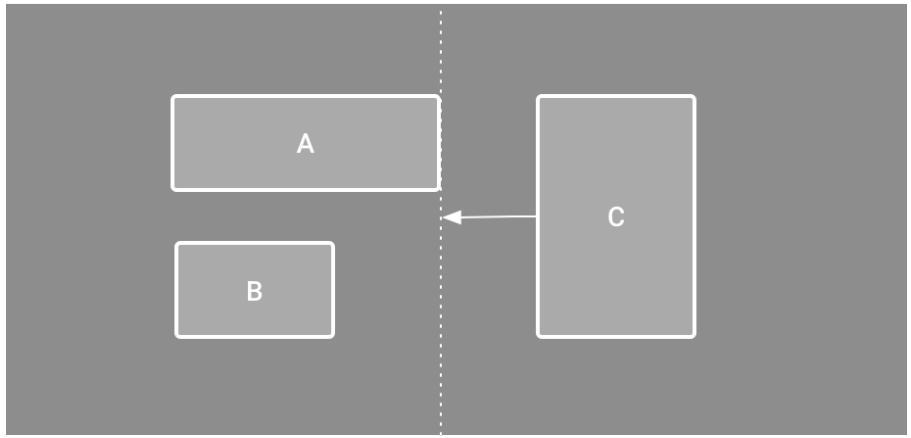


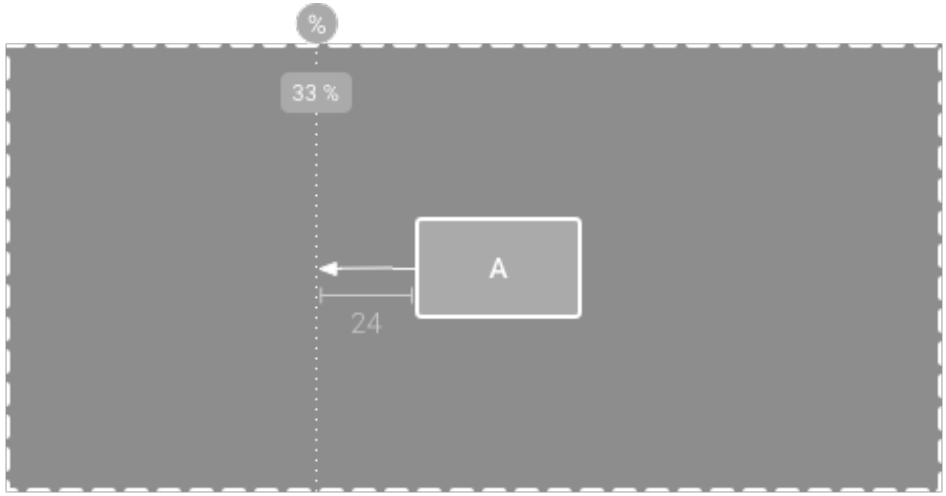


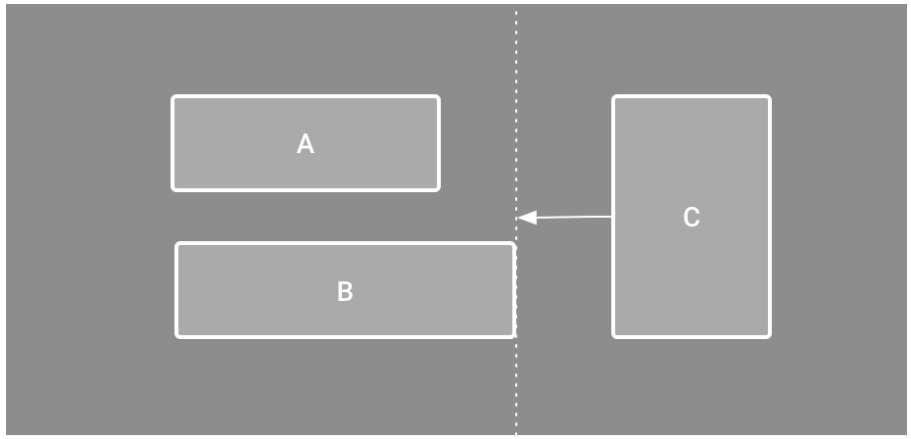


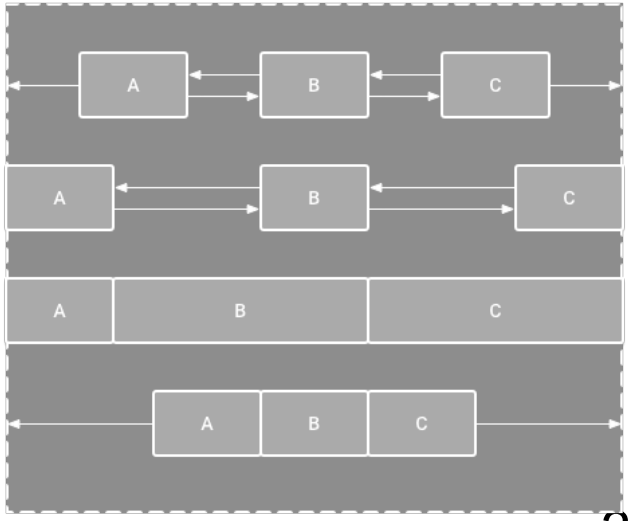


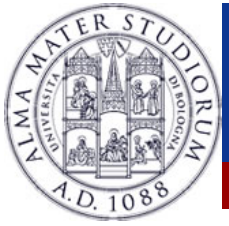
ConstraintLayout: constraints examples









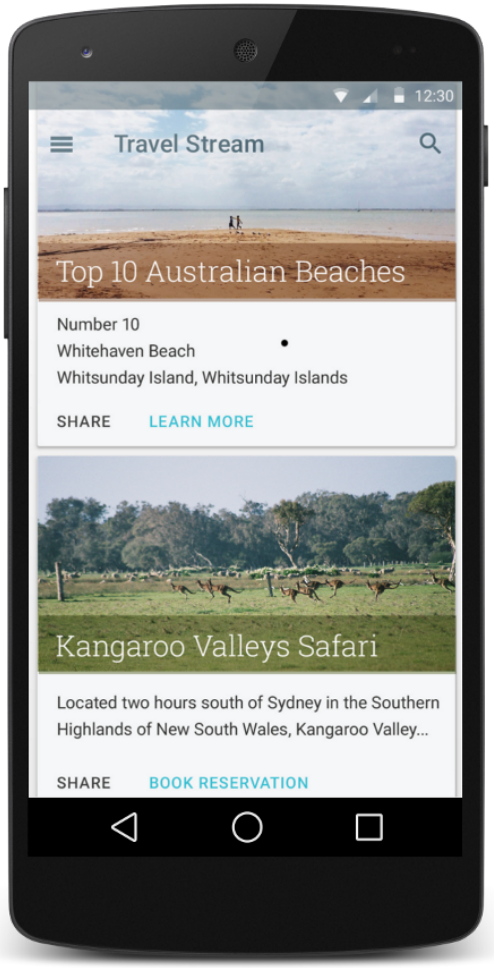


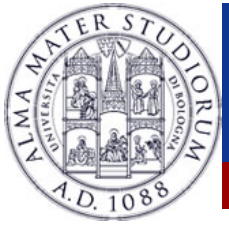
CardView

- A CardView is a ViewGroup
- It contains views
- Need to add

```
dependencies {
    implementation 'com.android.support:cardview-v7:27.1.0'
}
```

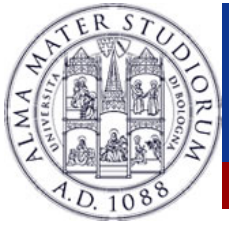
- Useful to group content related to the same entity





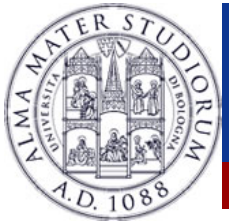
Adapters

- Used to visualize dynamic data
- Make a ViewGroup to interact with data
- Some methods:
  - isEmpty()
  - getItem(int position)
  - getCount()
  - getView()



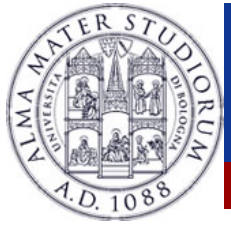
AdapterView

- A ViewGroup subclass
- Its subchilds are determined by an Adapter
- Some subclasses:
  - ListView
  - GridView
  - Spinner
  - Gallery

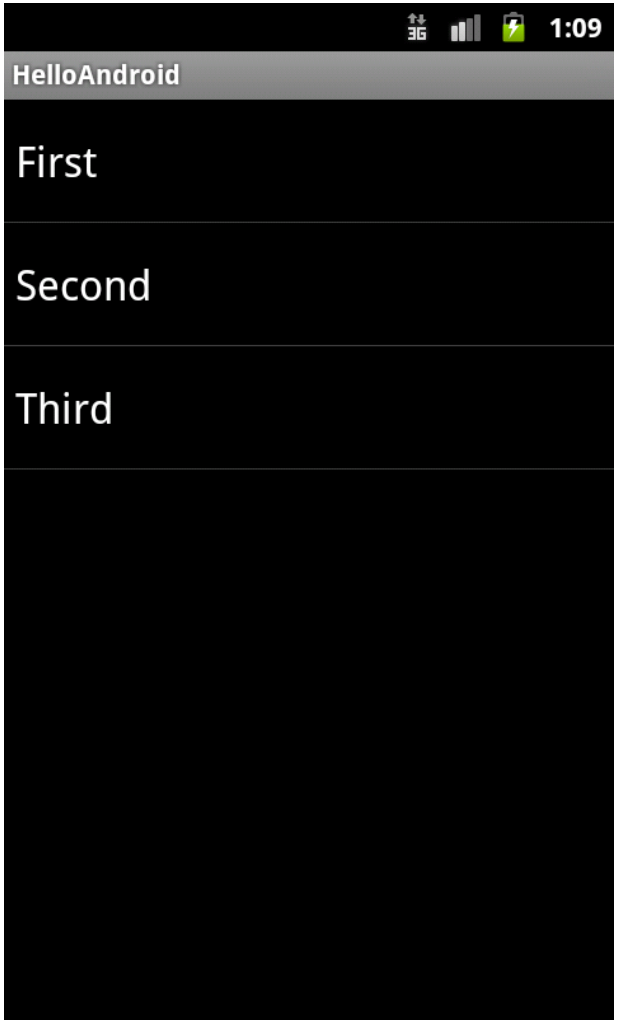


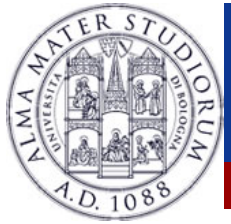
ListView example

```
public class HelloAndroidActivity extends Activity {


    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.list);


      String[] data = {"First", "Second", "Third"};
      ListView lv = (ListView)findViewById(R.id.list);
       lv.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, data));
    }
}


<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"    android:layout_height="match_parent"
    android:orientation="vertical"
    android:id="@+id/list" />
```



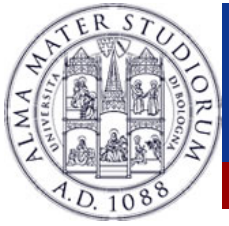
ListView





Other views/adapters

- Spinner, selection of multiple items
- Gallery, images
- ExpandableListView, list with hidden values
- TabWidget, tabbed layouts

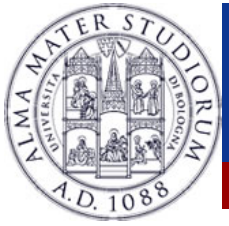


RecyclerView

- ListView available since API version 1
- Since Lollipop, RecyclerView has been introduced
  - Better handling of events
  - Separates data and layout
- Start with

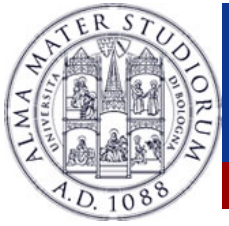
```
dependencies {
    implementation 'com.android.support:recyclerview-v7:27.0.0'
}
```

- Then add a **RecyclerView** to the Layout



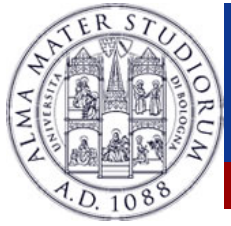
Step 1: LayoutManager

- For each RecyclerView, we have to define a LayoutManager
  - “A LayoutManager measues and positions item views on the RecyclerView. It also handles view view focus and visibility”
- In simple words, it is responsible for placing items in the layout
- Examples: LinearLayoutManager, GridLayoutManager, StaggeredGridLayoutManager, WearableLinearLayoutManager



Step 2: Adapter

- Create a class which extends RecyclerView.Adapter
- Override some methods:
  - getItemCount()
  - onCreateViewHolder()
    - Creates a new ViewHolder item
  - onBindViewHolder()
    - Bind the appropriate data to the ViewHolder



Differences: ListView and RecyclerView

- More efficient
- LayoutManager flexbility: think about LinearLayout and GridLayout
- Possible to add custom Decorations
- Animations made easy
- More than just notifyDataSetChanged()
  - notifyItemInserted(), notifyItemRemoved(), notifyItemChanged() and more