MODULE 4

LAYOUT

TABLE OF CONTENT

What is a Layout?

- A layout defines the visual structure for a user interface
  - Subclass of the **ViewGroup** class
- A layout can be declared in two ways
  - UI elements are declared in XML
    - XML vocabulary that corresponds to the View classes and subclasses
  - Layout elements are instanciated at runtime
    - View and ViewGroup objects can be created (and their properties manipulated) programmatically
- To separate View and business

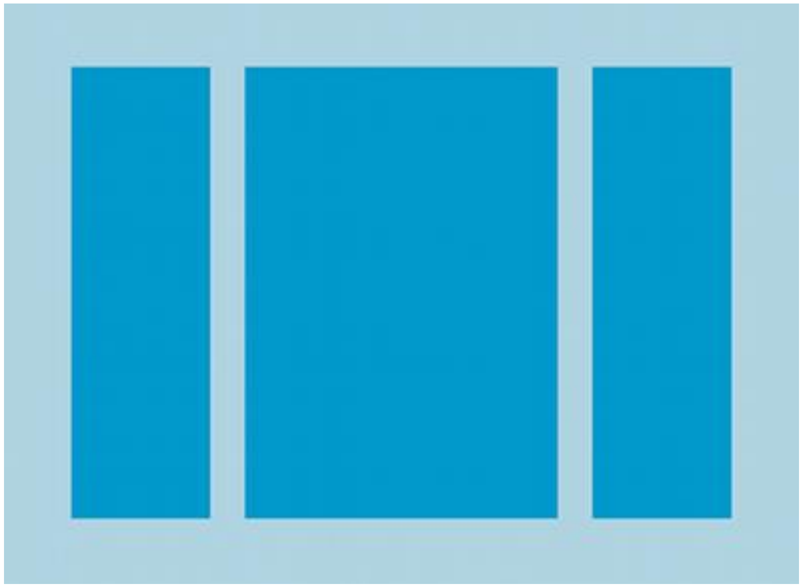
  ↳ declare views and layouts in xml

Layout Parameters

- **layout_width** and **layout_height**
  - Values
    - Absolute units such as pixels (not recommended !)
    - Density-independent pixel units (**dp**)
    - **wrap_content** : to size the view to the dimensions required by its content
    - **match_parent** : the view becomes as big as its parent view group will allow

Linear Layout

- Aligns all children in a single direction, vertically or horizontally
  - A single vertical or horizontal row
  - Scrollbar created if the length of the window exceeds the length of the screen



Linear Layout

- Specify the layout direction with the **android:orientation** attribute
- Linear Layout respects
  - *margins* between children
  - *gravity* (right, center, or left alignment) of each child

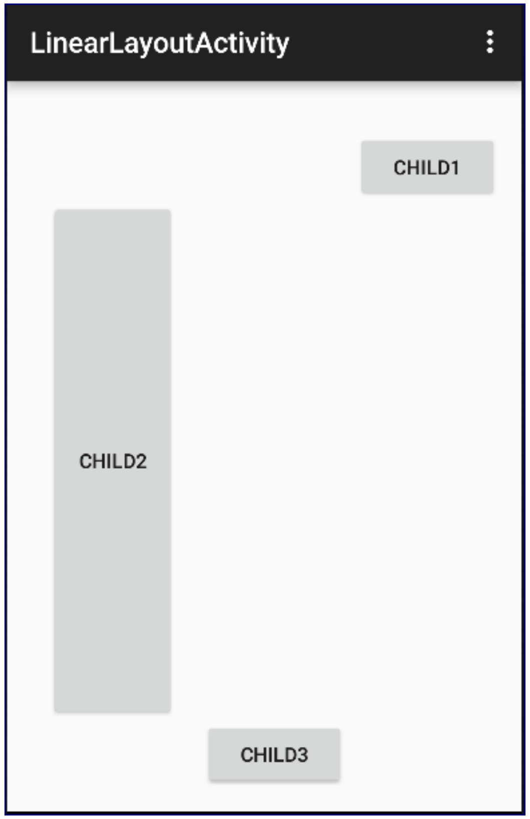
Linear Layout

- Assigning a *weight/importance* to individual children
  - How much space they should occupy on the screen
    - The remaining space in the parent view group is assigned to children in the proportion of their declared weight
    - Default weight is zero
  - With the **android:layout_weight** attribute

Linear Layout

E.g,

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="14dp" android:layout_marginTop="20dp">
    <Button
        android:id="@+id/relativeButton"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/Child1TextId" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="@string/Child2TextId" />
    <Button
        android:id="@+id/button3"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/Child3TextId" />
</LinearLayout>
```



Relative Layout

- Displays child views in relative positions
- The position of each view can be specified as
  - Relative to a sibling element (by specifying the ID of the sibling)
    - E.g, to the left-of or below another view
  - Relative to the parent RelativeLayout area
    - E.g, aligned to the bottom, left of center

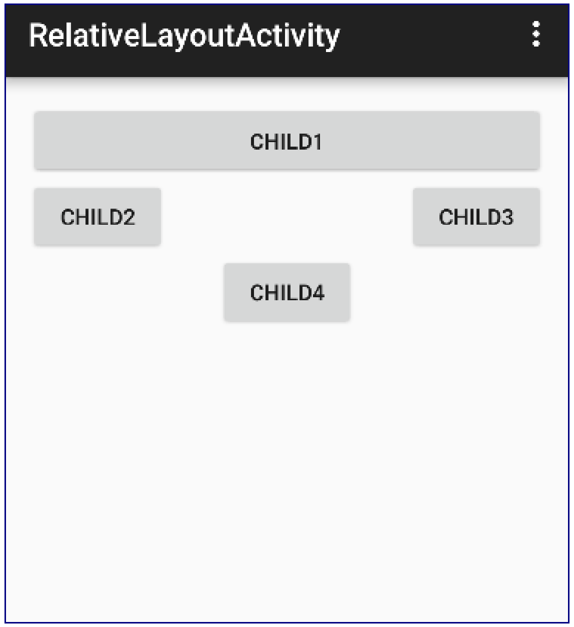
Relative Layout

- Properties to enable a layout position relative to the parent
  - Value is a boolean
  - E.g,
    - **android:layout_alignParentTop**
      - If "true", makes the top edge of this view match the top edge of the parent
    - **android:layout_centerVertical**
      - If "true", centers this child vertically within its parent
- Properties to enable a layout position relative to a sibling
  - Value is a view ID
  - E.g,
    - **android:layout_below**
      - Positions the top edge of this view below the view specified with a resource ID
    - **android:layout_toRightOf**
      - Positions the left edge of this view to the right of the view specified with a resource ID

Relative Layout

- By default, all child views are drawn at the top-left of the layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".RelativeLayoutActivity" >
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:text="@string/Child1TextId" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/button1"
        android:text="@string/Child2TextId" />
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/button1"
        android:layout_alignParentRight="true"
        android:text="@string/Child3TextId" />
    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button2"
        android:layout_centerHorizontal="true"
        android:text="@string/ChildText4ID" />
</RelativeLayout>
```

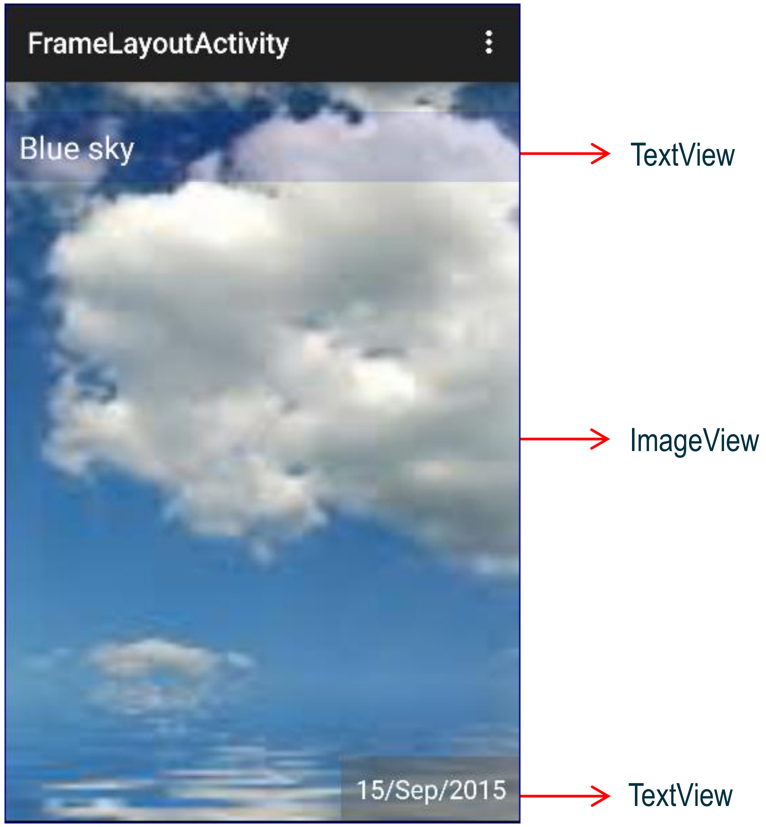


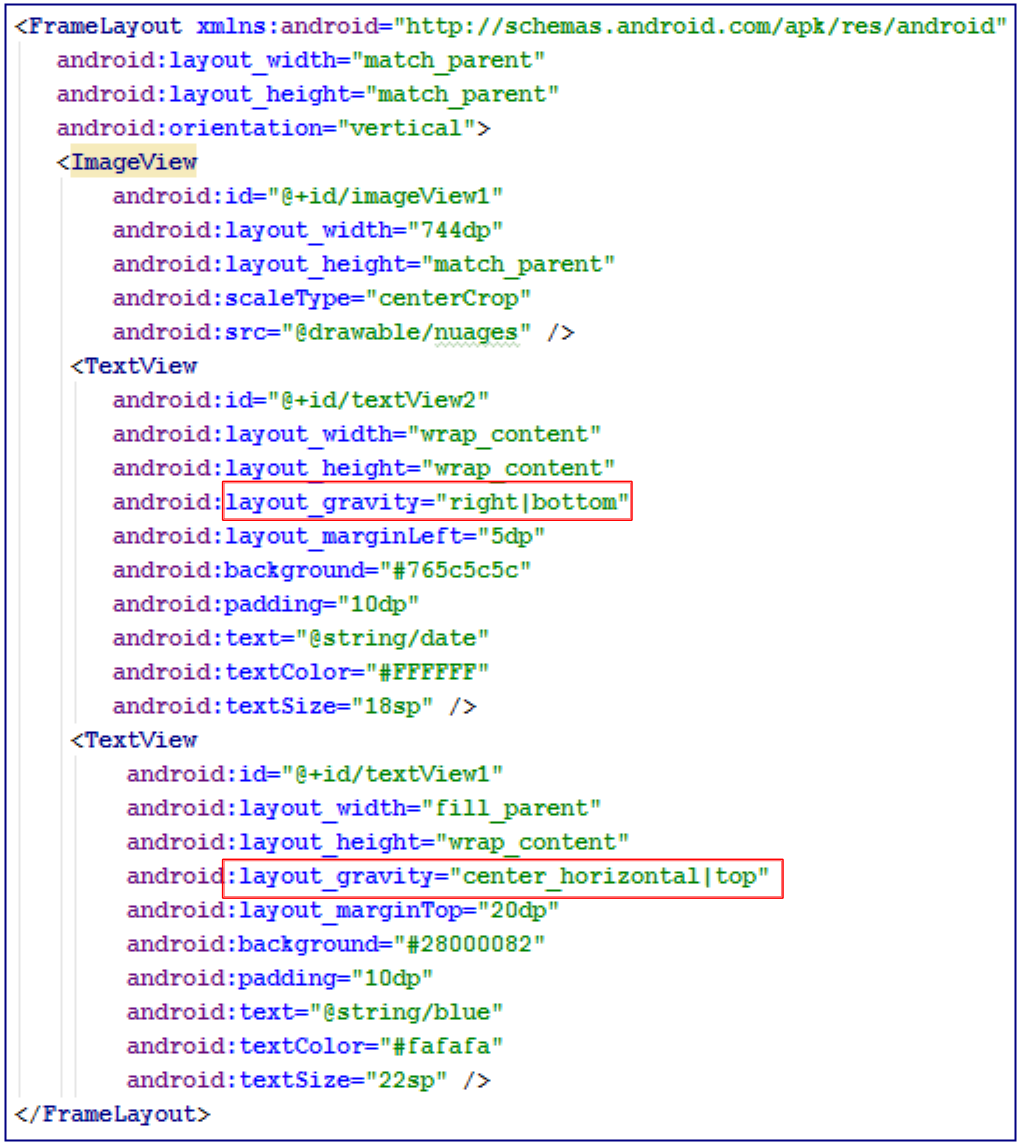
Frame Layout

- Usually used to display a single view item
- Or used to display multiple views which overlap
  - By assigning gravity to each child
  - Using the android:layout_gravity attribute
    - Values: top, bottom, left, right, center, center_vertical, center_horizontal

Frame Layout

- E.g,





```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <ImageView
       android:id="@+id/imageView1"
       android:layout_width="744dp"
       android:layout_height="match_parent"
       android:scaleType="centerCrop"
       android:src="@drawable/nuages" />
    <TextView
       android:id="@+id/textView2"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="right|bottom"
       android:layout_marginLeft="5dp"
       android:background="#765c5c5c"
       android:padding="10dp"
       android:text="@string/date"
       android:textColor="#FFFFFF"
       android:textSize="18sp" />
    <TextView
        android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal|top"
        android:layout_marginTop="20dp"
        android:background="#28000082"
        android:padding="10dp"
        android:text="@string/blue"
        android:textColor="#fafafa"
        android:textSize="22sp" />
</FrameLayout>
```

Table Layout

- Displays child views in rows and columns

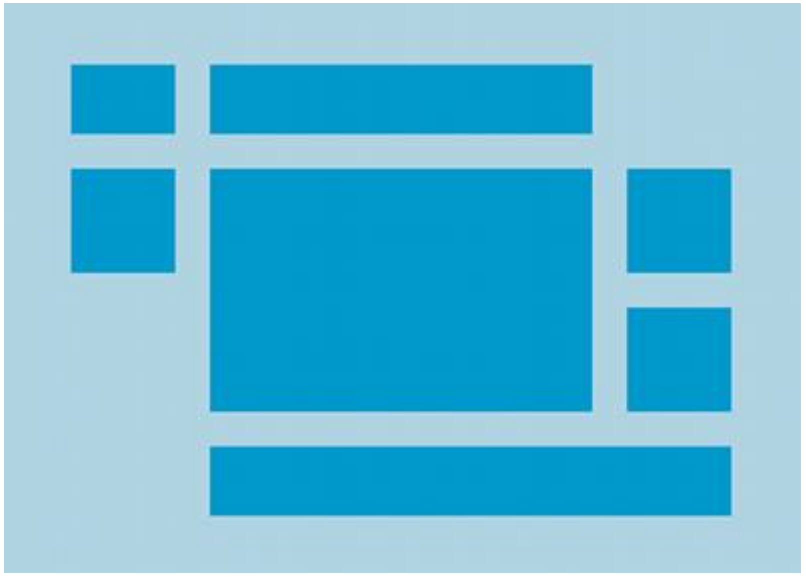
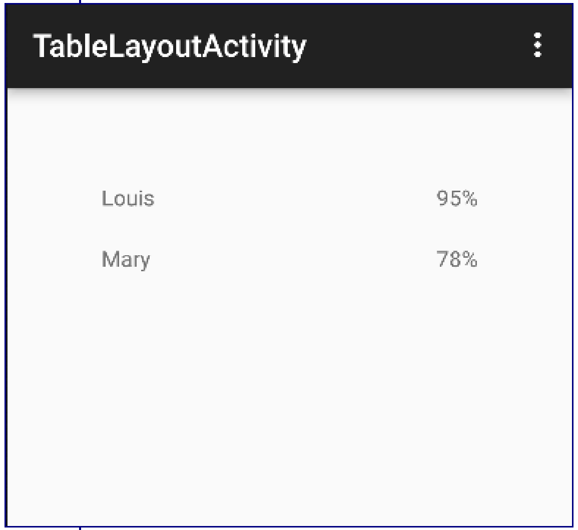


Table Layout

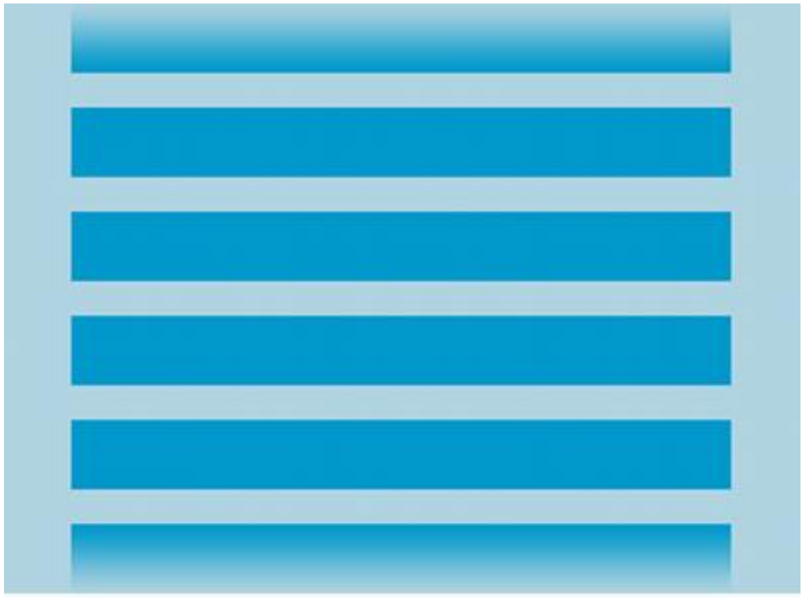
```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="50dip"
    android:paddingRight="50dip"
    android:paddingTop="50dip"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="@string/student1NameID"
            android:padding="10dp" />
        <TextView
            android:text="@string/student1ResultID"
            android:gravity="right"
            android:padding="10dp" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="@string/student2NameID"
            android:padding="10dp" />
        <TextView
            android:text="@string/student2ResultID"
            android:gravity="right"
            android:padding="10dp" />
    </TableRow>
</TableLayout>
```



```
Louis        95%
Mary         78%
```

ListView

- Container displaying a list of scrollable items



ListView

- For dynamic or not pre-determined content
  - To populate the layout with views at runtime
- Items are automatically inserted to the list using an Adapter
  - Items are populated from a source such as an array or database query
  - Item values are converted into a views placed into the list

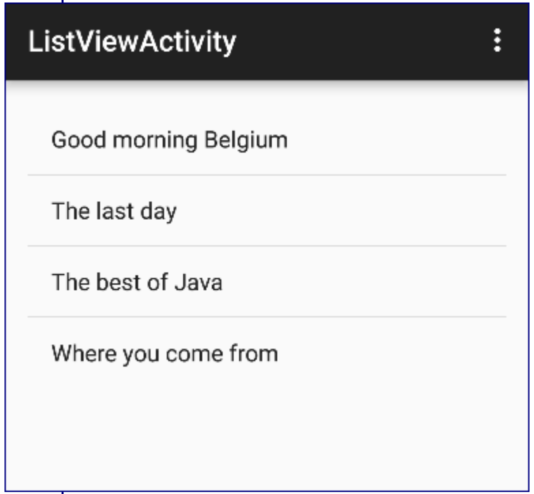
ListView

- Use ArrayAdapter to populate ListView object
  - The arguments of constructor
    - The context
    - The layout that contains a TextView for each value in the array
    - The string array

ListView

- E.g,

```
import android.app.Activity;
import android.os.Bundle;
import java.util.ArrayList;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends Activity {

  private ListView bookList;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      bookList = (ListView)this.findViewById(R.id.listView1);

      ArrayList<String> allBooks =  ...

      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
              android.R.layout.simple_list_item_1, allBooks);

      bookList.setAdapter(adapter);
```


ListViewActivity

Good morning Belgium

The last day

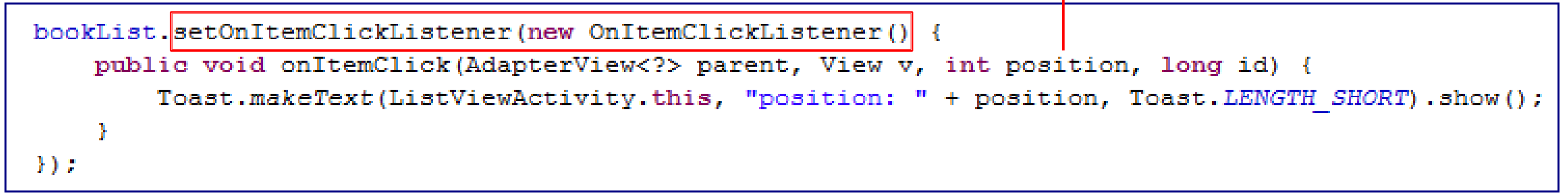
The best of Java

Where you come from

ListView

- Click event handling on items

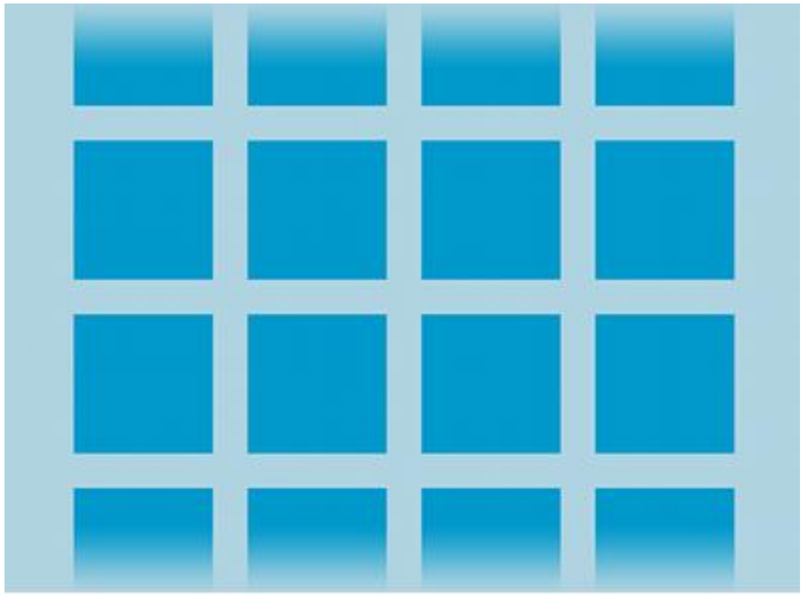
Position of the item in the list



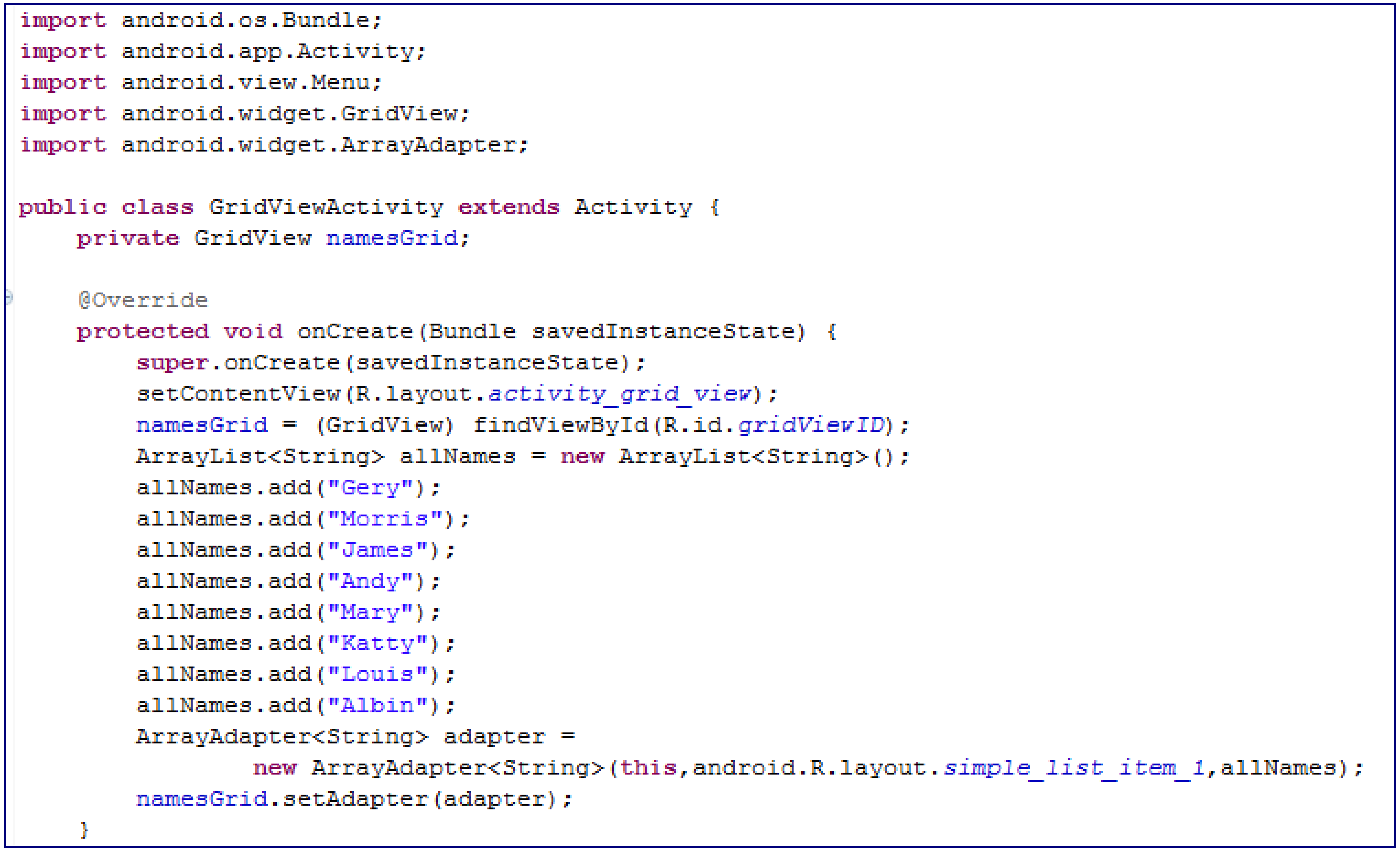
```
bookList.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
        Toast.makeText(ListViewActivity.this, "position: " + position, Toast.LENGTH_SHORT).show();
    }
});
```

GridView

- Container displaying items in a two-dimensional, scrollable grid
- Items automatically inserted to the layout using a ListAdapter



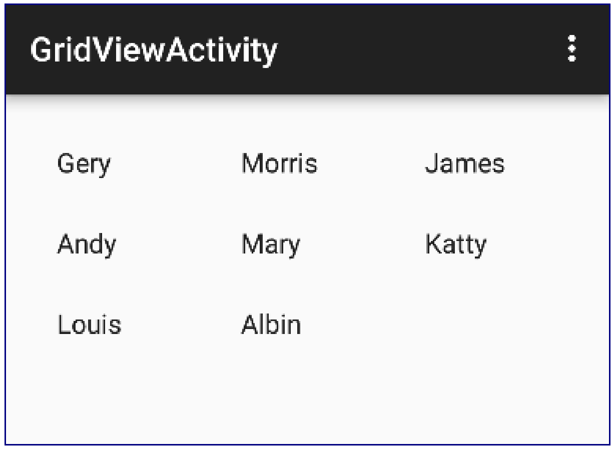
GridView



```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.GridView;
import android.widget.ArrayAdapter;

public class GridViewActivity extends Activity {
    private GridView namesGrid;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_grid_view);
        namesGrid = (GridView) findViewById(R.id.gridViewID);
        ArrayList<String> allNames = new ArrayList<String>();
        allNames.add("Gery");
        allNames.add("Morris");
        allNames.add("James");
        allNames.add("Andy");
        allNames.add("Mary");
        allNames.add("Katty");
        allNames.add("Louis");
        allNames.add("Albin");
        ArrayAdapter<String> adapter =
                new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,allNames);
        namesGrid.setAdapter(adapter);
    }
```

GridView

```xml
<GridView
    android:id="@+id/gridViewID"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:numColumns="3" >
</GridView>
```



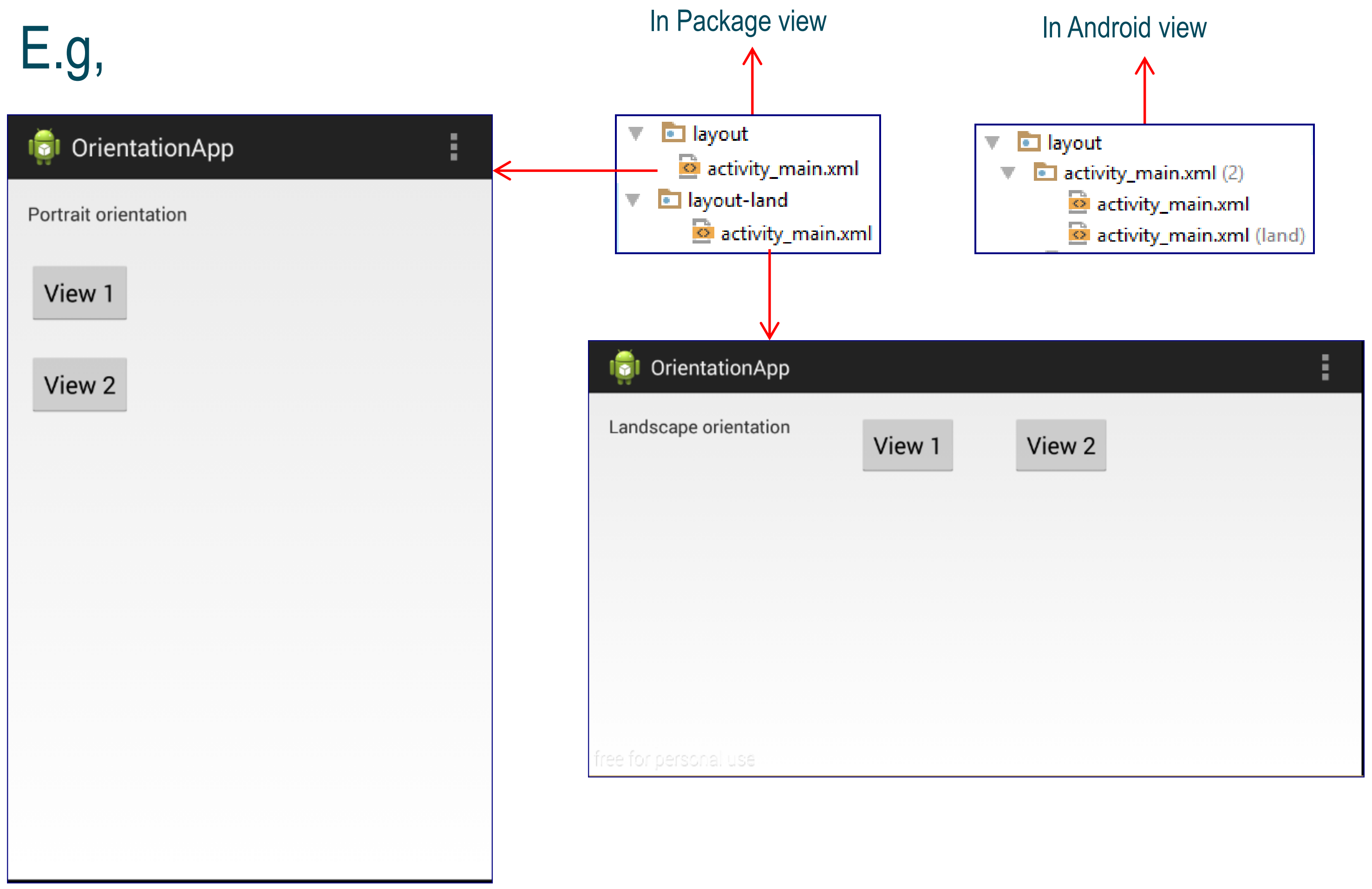
```
GridViewActivity

Gery        Morris      James

Andy        Mary        Katty

Louis       Albin
```

Portrait or Landscape Orientation

- Different layouts for portrait and landscape orientations
- Portrait orientation (by default)
  - Layout in **res/layout**
- Landscape orientation
  - Layout in **res/layout-land**
- Same filenames for xml files in
  - res/layout
  - res/layout-land

Portrait or Landscape Orientation

- E.g,



Supporting Multiple Screens

- Different types of devices
- Measured by density-independent pixel (dp)
  - 1 dp is equivalent to one physical pixel on a 160 dpi screen
    - The baseline density for a "medium" density screen
  - The conversion of dp units to screen pixels is
    - px = dp * (dpi / 160)
    - E.g, on a 240 dpi screen, 1 dp equals 1.5 physical pixels

Supporting Multiple Screens

- E.g., typical screen widths
  - 320dp: typical phone screen
    - 240x320 ldpi, 320x480 mdpi, 480x800 hdpi, …
  - 480dp: tweener tablet
    - 480x800 mdpi
  - 600dp: 7” tablet
    - 600x1024 mdpi
  - 720dp: 10” tablet
    - 720x1280 mdpi, 800x1280 mdpi,…

Supporting Multiple Screens

- Different layout folders according to the type of device
  - res/layout/main_activity.xml
    - For handsets (smaller than 600dp available width)
  - res/layout-sw600dp/main_activity.xml
    - For 7” tablets (600dp wide and bigger)
  - res/layout-sw720dp/main_activity.xml
    - For 10” tablets (720dp wide and bigger)

Webography

- http://developer.android.com/guide/topics/ui/declaring-layout.html
- http://developer.android.com/guide/topics/ui/layout/linear.html
- http://developer.android.com/guide/topics/ui/layout/relative.html
- http://developer.android.com/guide/topics/ui/layout/listview.html
- https://developer.android.com/guide/topics/ui/layout/gridview.html
- https://developer.android.com/guide/topics/ui/layout/grid.html
- http://developer.android.com/training/basics/supporting-devices/screens.html
- http://developer.android.com/guide/practices/screens_support.html