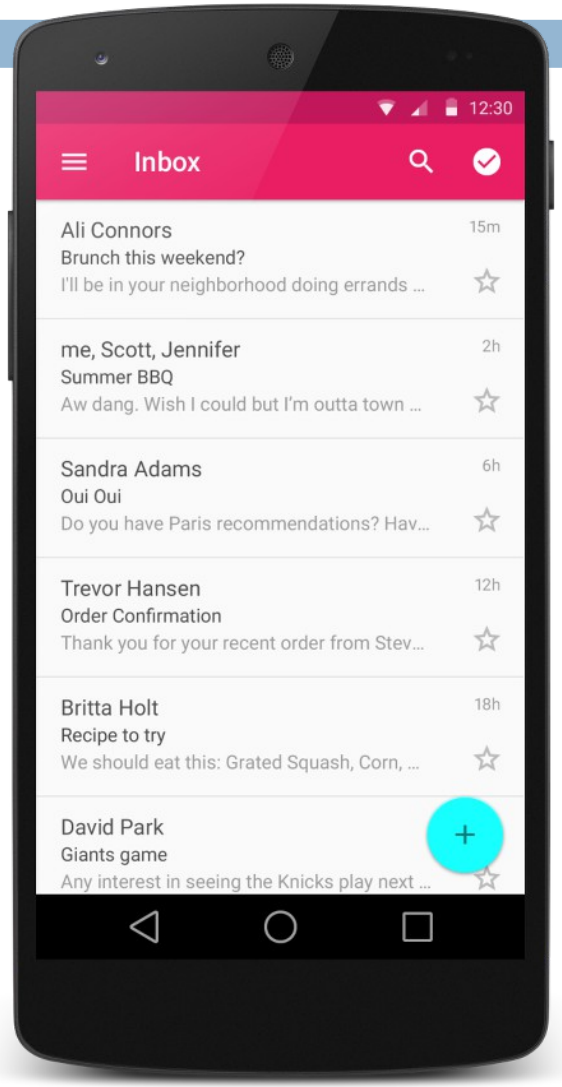
P82 Mobile Application Development - Android

Graphical Interface

User Interfaces (UI)

The Androids application interfaces are **organized in View and Layout (templates, layouts)**.

The views are the **elements** of the graphical interface that the user sees and on which he can **act**.



ViewGroups hierarchy

- *Graphic elements* **inherit** the View class
- We can group them in a **ViewGroup**: some of which are layouts predefined (LinearLayout, RelativeLayout, TableLayout ...)
- others are **Views**, specialty (ListView, ScrollView, SurfaceView ...)
- or simple **groupings of components** (Gallery, RadioGroup, Spinner ...)
- (they are Views containers)

Create a user interface

Define two elements:

- **the user interface:**

  1) XML file (res / layout)

  2) and / or dynamic in Java code

- user **logic** (interface behavior)

Interaction through EventListeners or by redefining callback methods

The advantages:

- separation between presentation and logic of your application,
- integrator will be able to modify the interface without interfering with the developer code.

Views and templates (main)

- **LinearLayout**: The elements are aligned from left to right or from top to bottom.
  - Modifying the meaning with the orientation property
- **RelativeLayout**: child elements are positioned relative to each other, the first child serving as a reference for others
- **TableLayout**: allows you to position your views in rows and columns like a table (matrix layout)
- **FrameLayout**: used to display objects by stacking them on top of each other (stacking elements)

Important attributes

- android:layout_width,

  android:layout_height (required):
  - "match_parent": the element fills all the parent element
  - "wrap_content": takes the necessary place to display
- android:gravity: defines the alignment of the elements relative to its parent (top, left, bottom, center ...)

Units of measurement

- **dp (Density-independent Pixels):**
  - independent of pixel density
  - 160dp will always measure **1 inch regardless of the screen type**
  - dip is an equivalent notation
- **sp (Scale-independent Pixels):**
  - like dp, but also influenced by the font size chosen by the user to use as a unit for character sizes
- **pt (Points)**: 1/72 of an inch
- **px (Pixels)**: corresponds to a pixel, variable according to the devices
- **mm (Millimeters)**: millimeters
- **in (Inches)**: inches

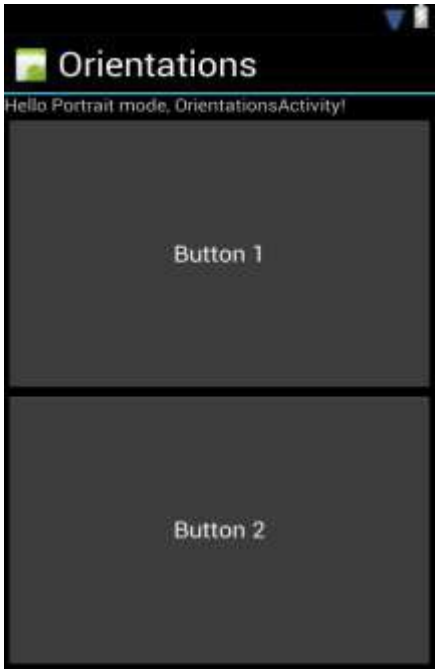
Weight of elements (layout_weight)

Can give a dimension by using the relative weight or percentage that will occupy this element.

android: layout_weight

for these two buttons to take 50% of the space we will have:

```
<Button android:layout_width="match_parent"
    android:layout_height="0px"
    android:text="Button 1"
    android:layout_weight="1" />
<Button android:layout_width="match_parent"
    android:layout_height="0px"
    android:text="Button 2"
    android:layout_weight="1" />
```



Best Practices

- Use layouts and relative dimensions, never absolute, (to remain independent of the device on which the application runs)
- For the size of the components in the layout, use relative values "wrap_content" or "match_parent" ("fill_parent" is equivalent and deprecated since version 8 of the API)
- If the size and position of a component must be specified absolutely (in the case of a game screen, for example), use the units of measurement dp or sp.

Access to the R resource in XML

- The resources are accessible via the syntax:

"**@[package:]type_ressource/nom_ressource"** android:text=*"@string/hello_world"*

- "package" is optional, used to distinguish R and android classes.
- "type" is the type of resource (layout, drawable, string, id)
- "name" is the name of the constant declared in R

- Can generate constants in R with syntax "+",

  "@+[package:]type/name"

  android:id="@+id/lblHello"

Access to R resource in Java

Primary resources

Resources res = **getResources();**

String hw = res.getString(R.string.hello);

Xxx o = res.getXxx(id);

- Views

TextView text = (TextView)**findViewById(R.id.text);**

texte.setText(“ HAH !");

Widgets

- TextView to display text
- EditText to enter text
  - Can enter inputType (password, date ...)
- ImageView to display images
  - There are several ways to link an image with the object. The simplest and use the method setImageResource (R.drawable.monImage);
- Button to display a button
  - In general we will redefine the setOnClickListener (...) method;
- RadioButton usually put in a ButtonGroup
- CheckBox

Layouts (container)

- LinearLayout the Widgets will position themselves under each other or next to others

- ScrollView to allow adding a scroll in our view, if the content exceeds

- RelativeLayout
  - Position relative to the parent or brother
  - Requires reference ID for positioning
  - Component ID must be declared first

- WebWiew to display a web page

- FrameLayout for stacking children (z-index)