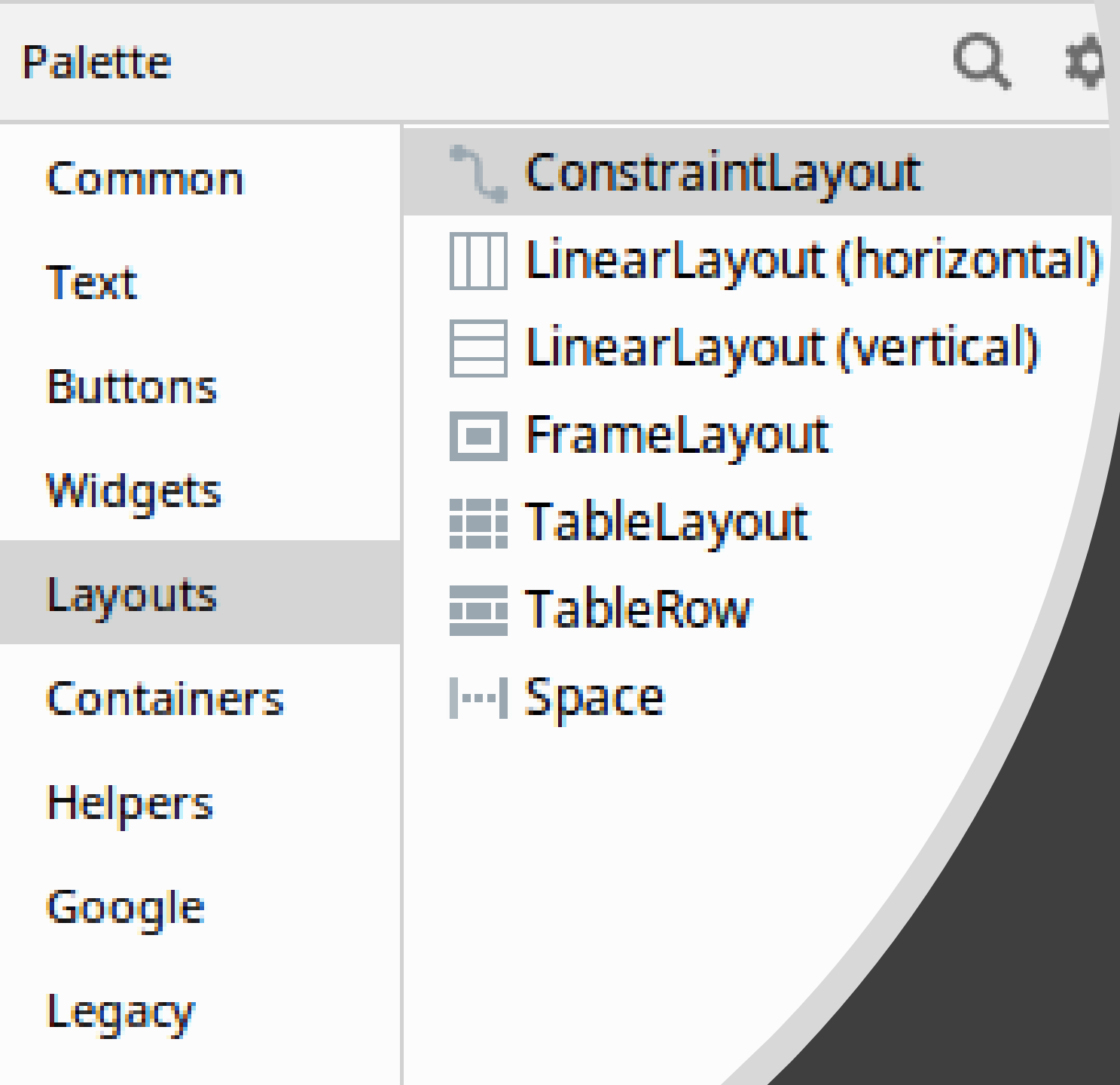
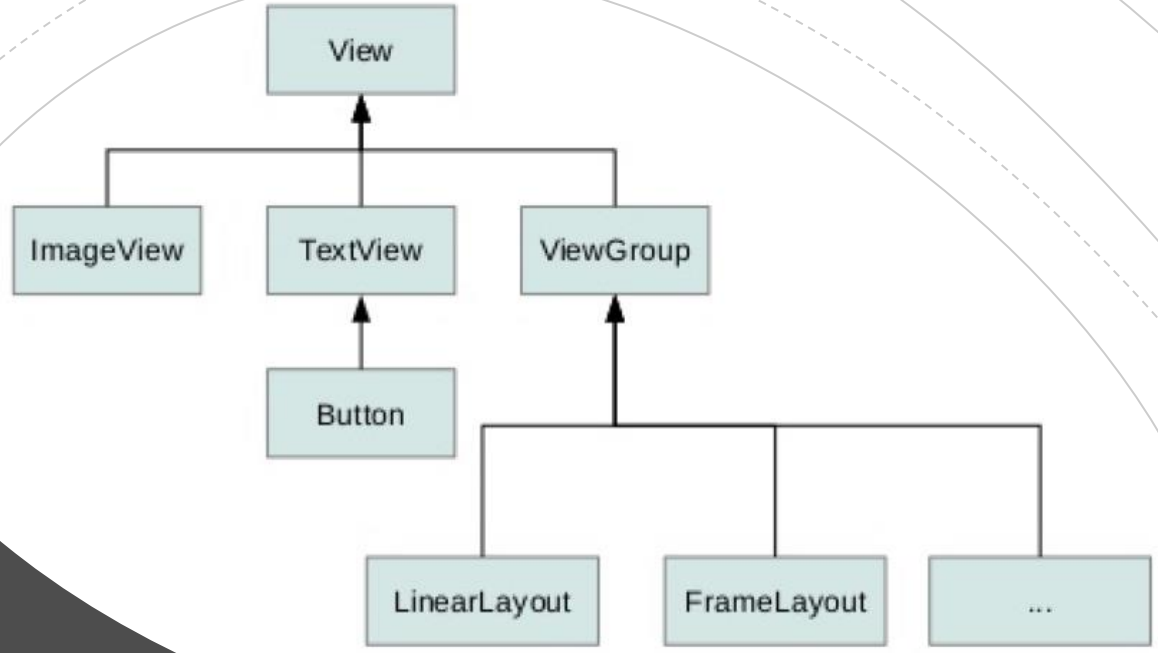


Containers and Layouts

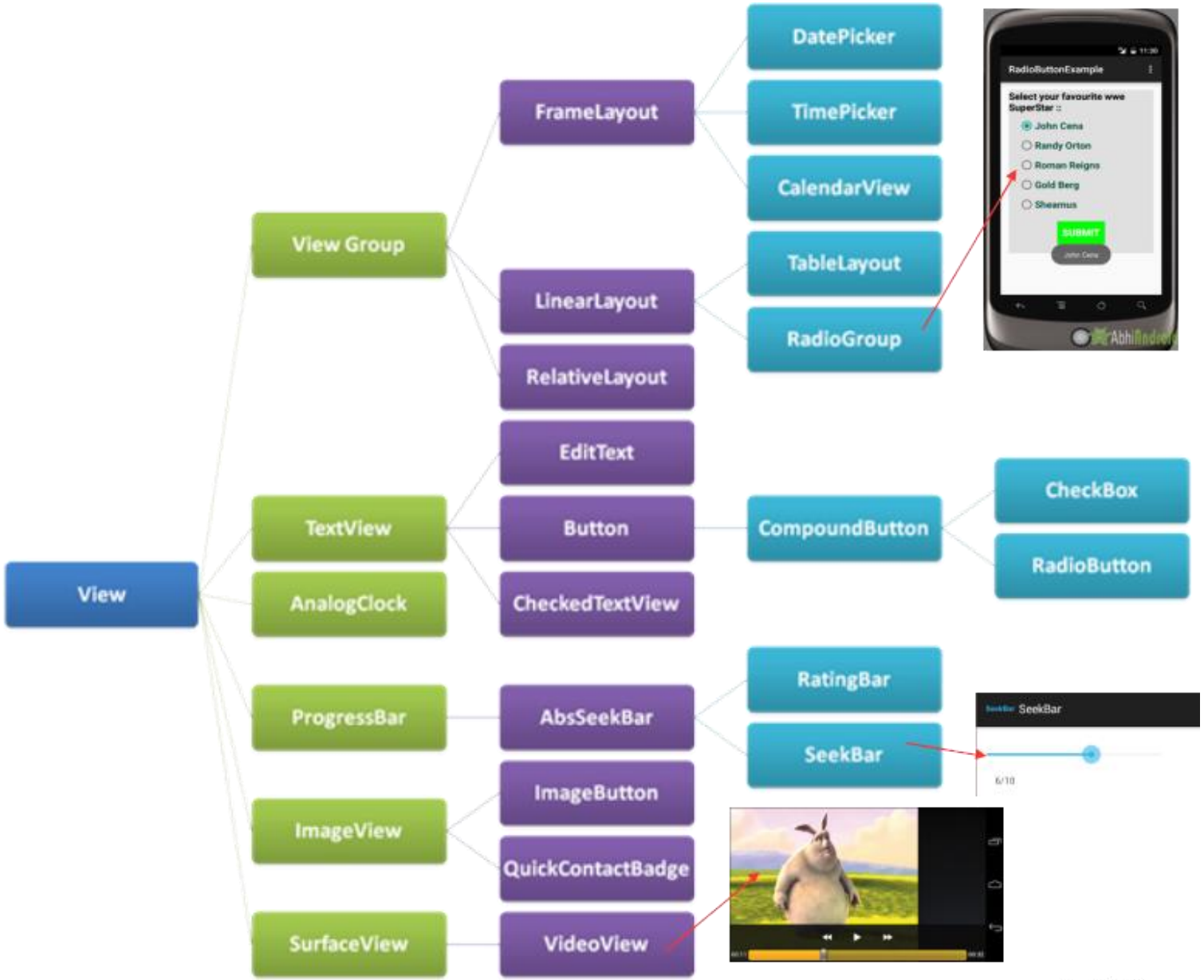
How to change the visual structure for a user interface



View
and ViewGroup

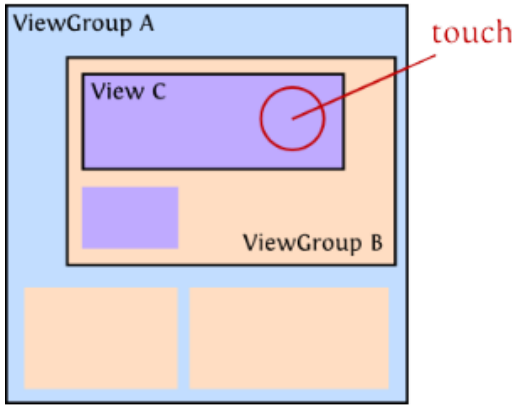
View

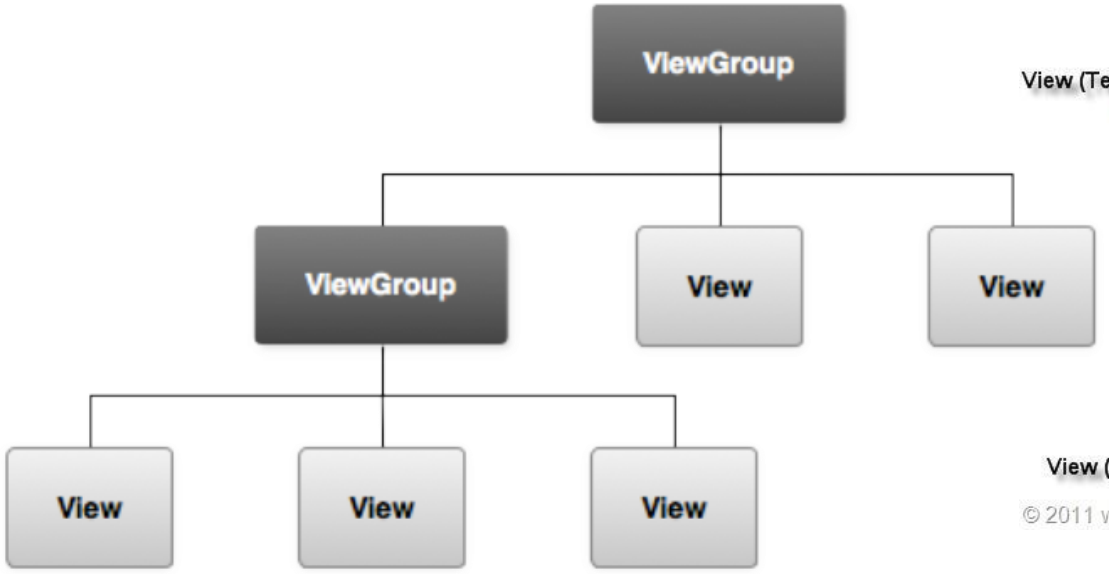
- A View is an object that draws something on the screen that the user can interact with.
- It is the base class (superclass) for all UI elements (controls or widgets).

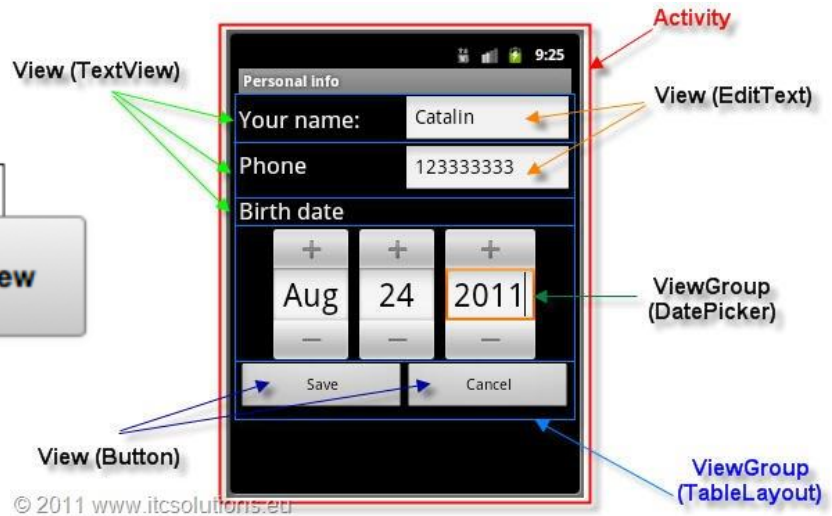


ViewGroup

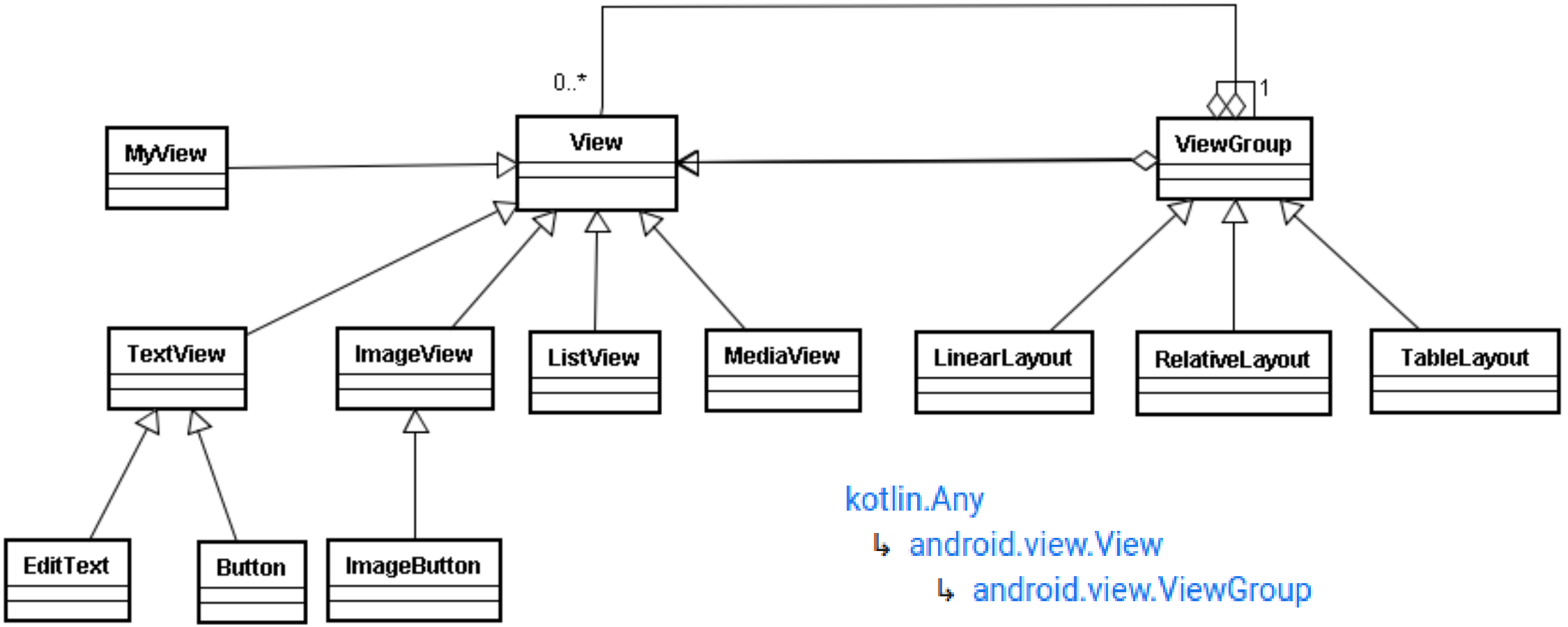
- A ViewGroup is an object that holds other View (and ViewGroup) objects in order to define the layout of the interface.
- LinearLayout, FrameLayout, RelativeLayout, etc. are specialized sub classes of ViewGroup class that layout their child in specific format.







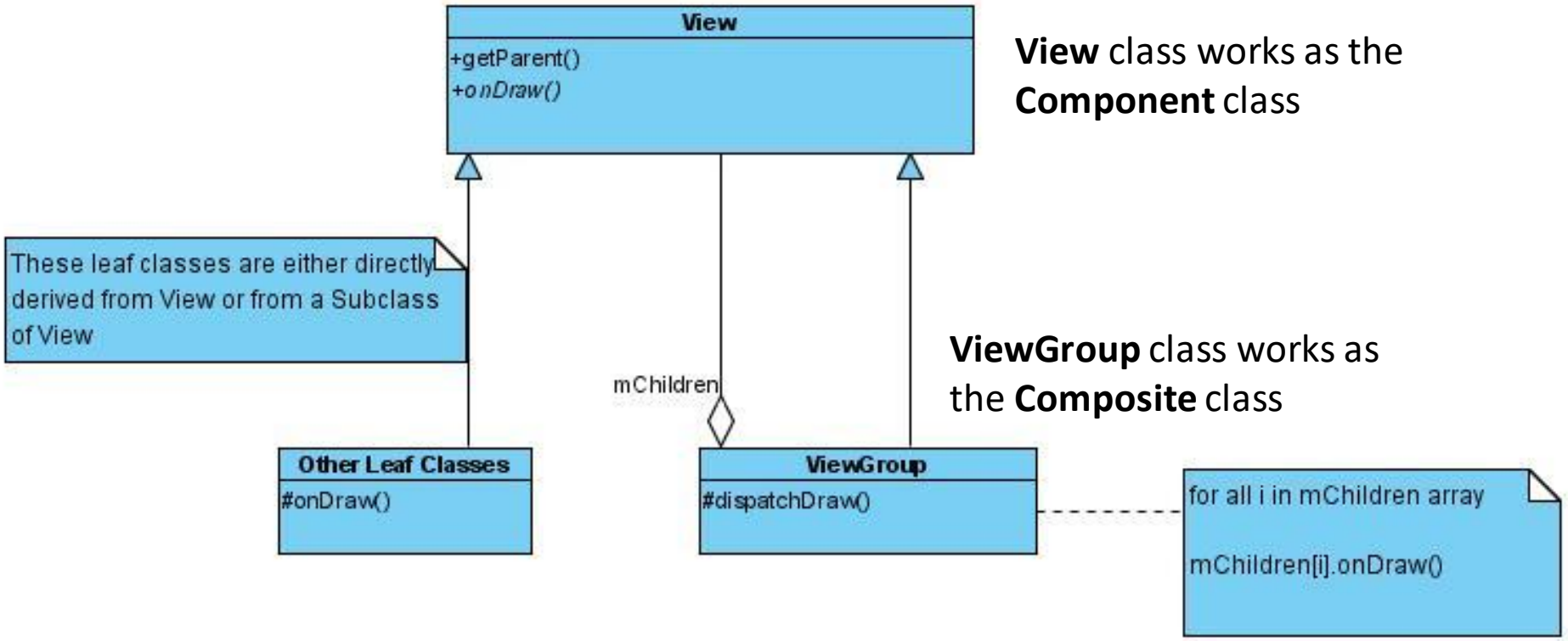
ViewGroup and View



kotlin.Any
↳ android.view.View
↳ android.view.ViewGroup

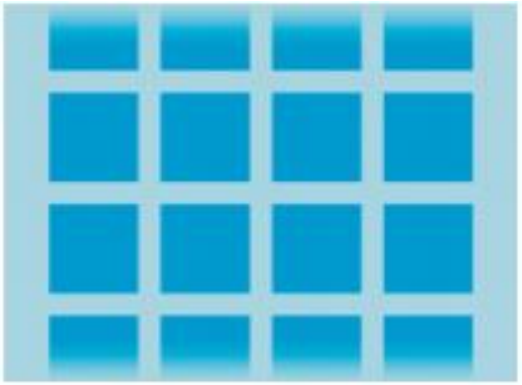
ViewGroup and View as example of Composite Design Pattern

- In Android implementation, the onDraw function in the Component (View) plays the role of **Operation** function.



LinearLayout

RelativeLayout

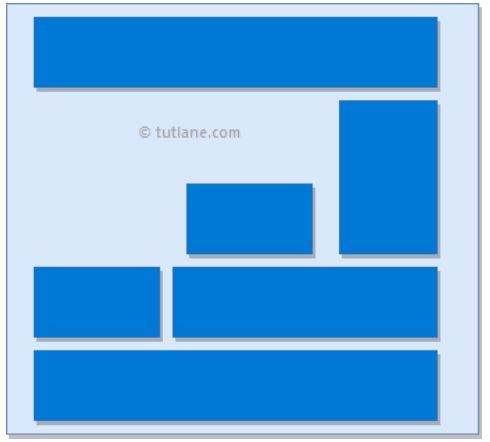


GridLayout

TableLayout

Layout

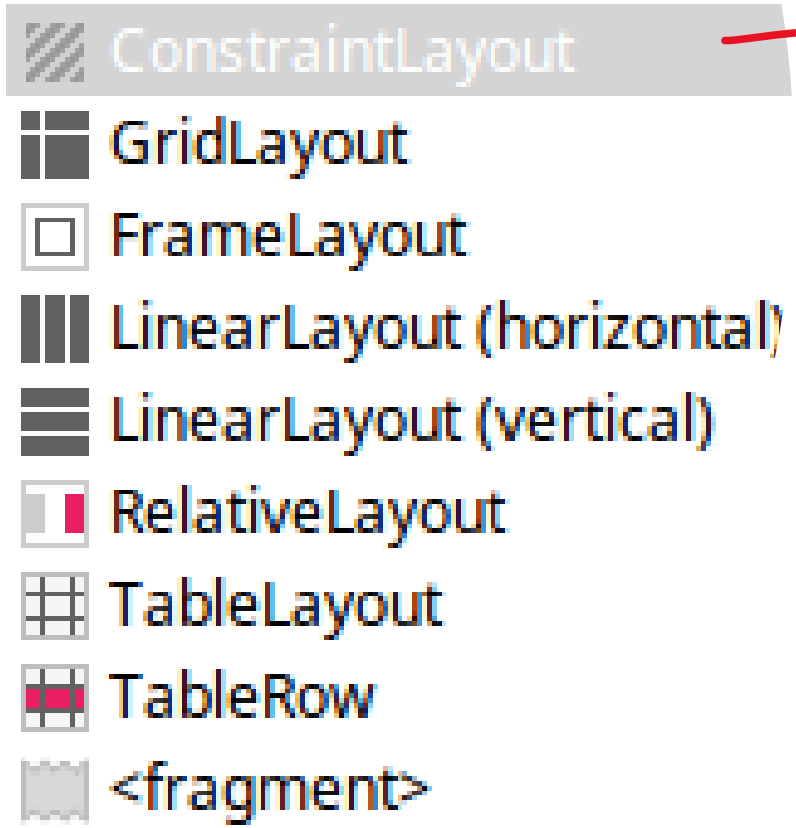
Layout

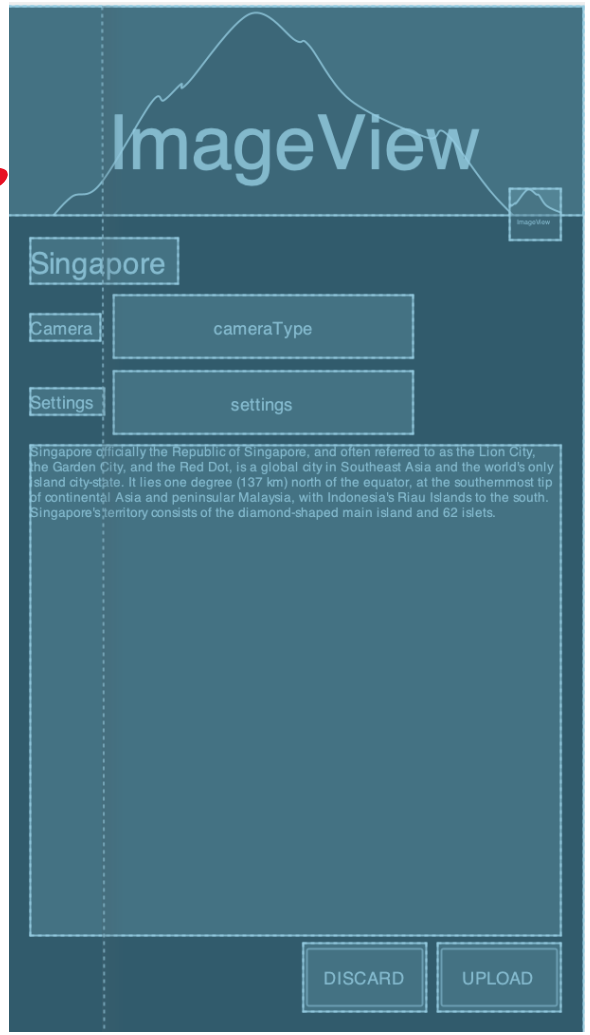


- A layout defines the visual structure for a user interface, such as the UI for an activity or app widget.

- How to define a layout:
  - **code**: instantiate View objects in code and start building a tree
  - **resource**: define it as a layout resource via XML.
- The name of an XML element for a view is respective to the Android class it represents
  - the <TextView> element creates a **TextView** object
  - the <LinearLayout> element creates a **LinearLayout** view group.

Common layouts

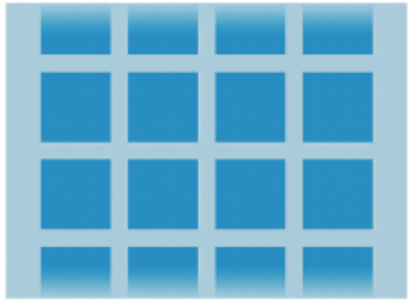




Layouts with an Adapter

- When the content for your **layout** is dynamic or not predetermined, you can use a layout that subclasses AdapterView to populate the layout with views at runtime.
- These layouts use an Adapter to bind data to its layout.

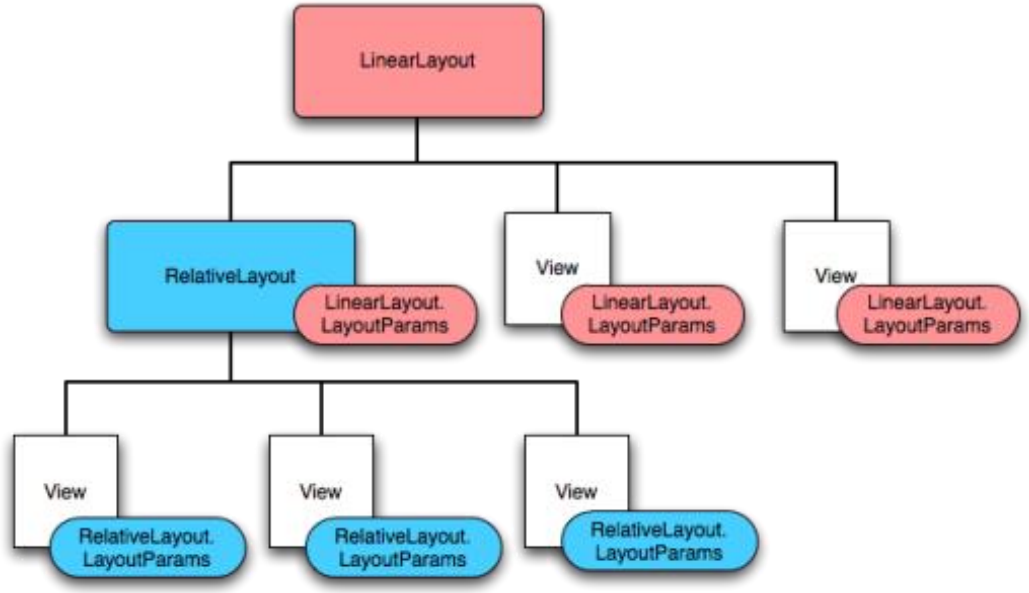
`ListView`: displays a scrolling single column list.



`GridView`: displays a scrolling grid of columns and rows.

XML Layout Attributes

- Each Layout class has an inner class called **LayoutParams** that defines general XML parameters that layout uses.
- These parameters are always named android:layout_blah, and usually have to do with sizes and margins.



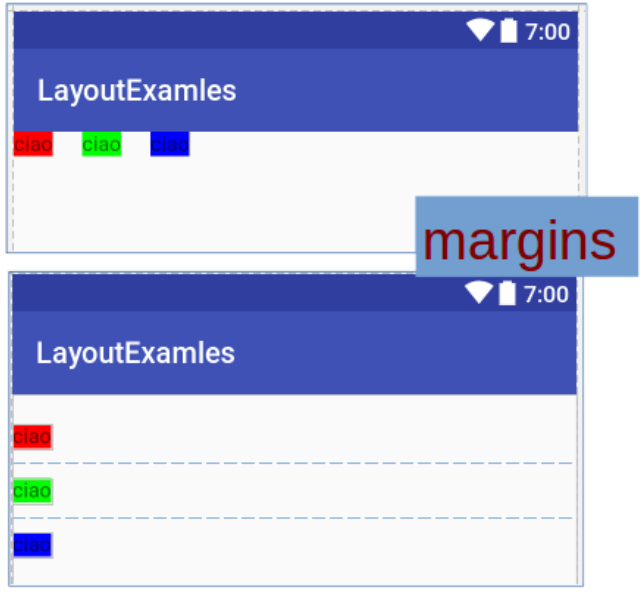

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        android:id="@+id/button"
        android:background="#358a32" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2"
        android:id="@+id/button2"
        android:background="#0058b6" />
</LinearLayout>
```

Commonly Used Attributes

- **Size** – android:layout_height, android:layout_width, android:layout_weight
- **Alignment** – android:layout_gravity, android:gravity
- **Margins** (blank space outside) – android:layout_marginBottom, android:layout_marginTop, android:layout_marginLeft, android:layout_marginRight
- **Padding** (blank space inside) – android:paddingBottom, android:paddingTop, android:paddingLeft, android:paddingRight
- **ID** – android:id
- **Colors** – android:background, android:textColor (e.g., for TextView or Button)
- **Click handler** – android:onClick

margins and **paddings**

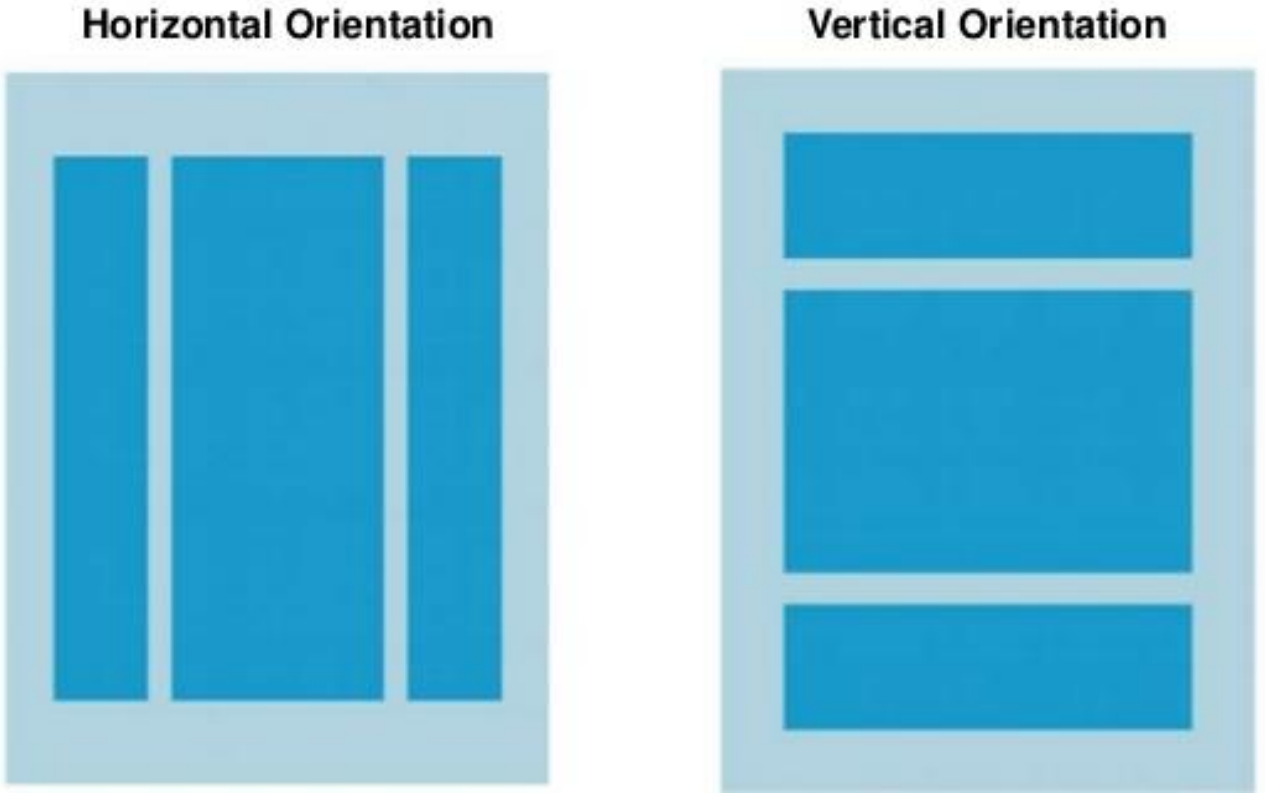
- Two layout options that could lead to similar effects, but generally have different application: margins and paddings.
- Both define **free space**, but margins work outside an element boundaries and paddings inside an element.



ViewGroup and Layout

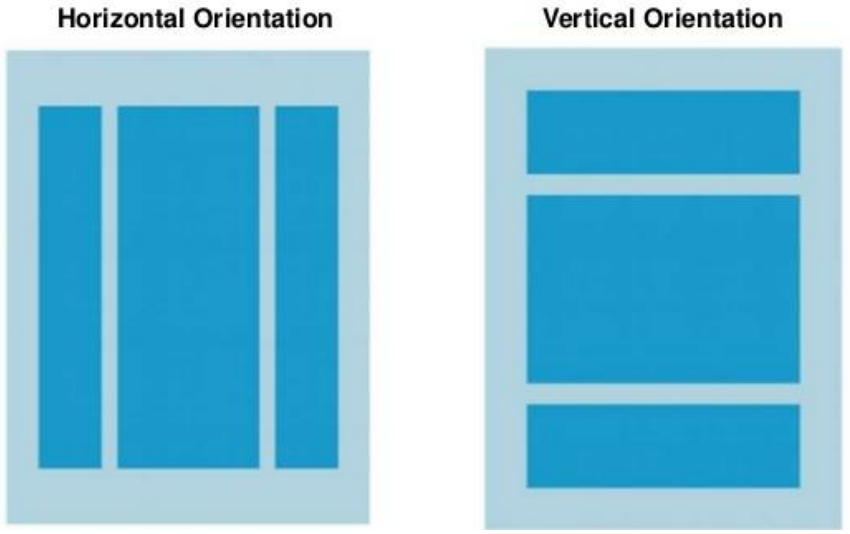
- Every view needs to specify:
  - android:layout_height
  - android:layout_width
- The value is a dimension or one of:
  - wrap_content: sets the size of a View to the minimum required to contain the contents it displays
  - match_parent or ~~fill_parent~~: expands the View to match the available space within the parent container.
  - 0dp: height or width when set to "0dp", means you want to fill all the available space for height or width (are mostly used in combination with "weight")

Linear layout



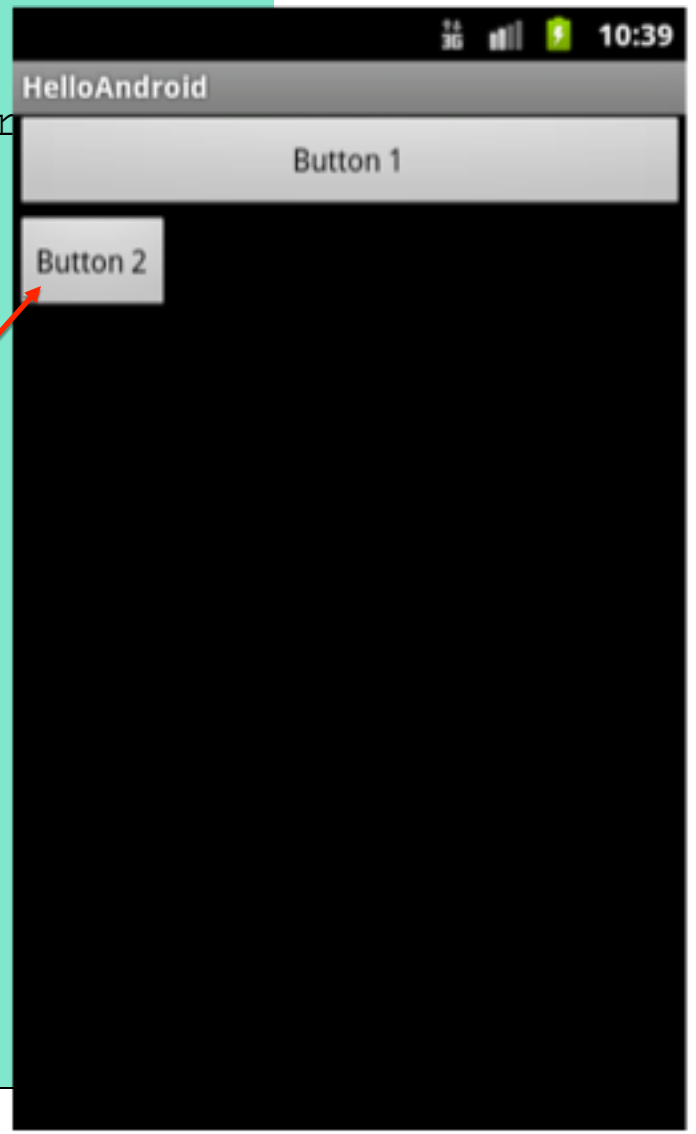
Linear Layout

- Dispose views on a single row or column:
  - **in XML**: depending on the attribute android:layout_orientation (horizontal|vertical).
  - **in code**: using the `setOrientation(int orientation)` with VERTICAL and HORIZONTAL (int constants of LinearLayout).



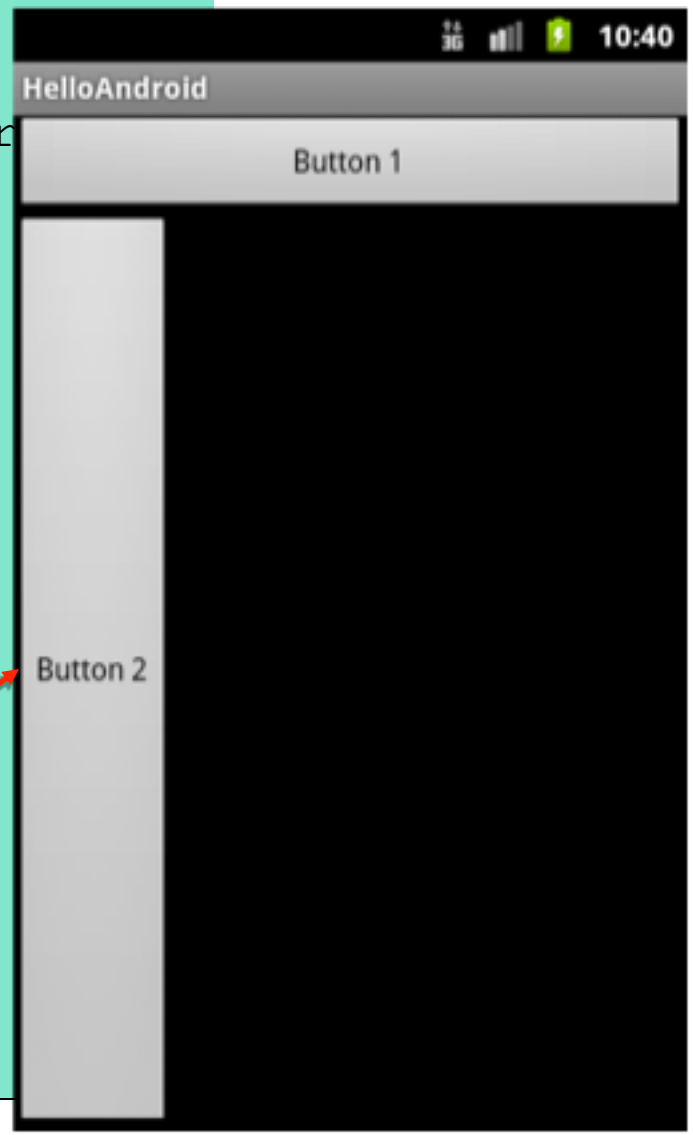
LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/andr
    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="wrap_content"
        android:text="@string/buttonString2" />
</LinearLayout>
```



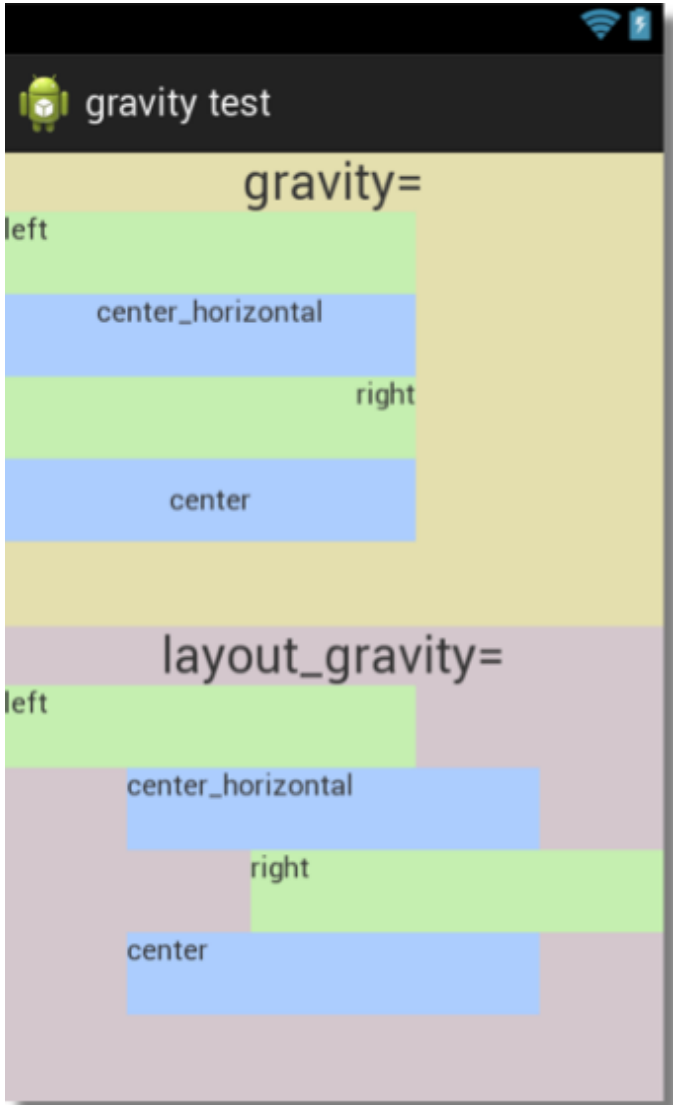
LinearLayout

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



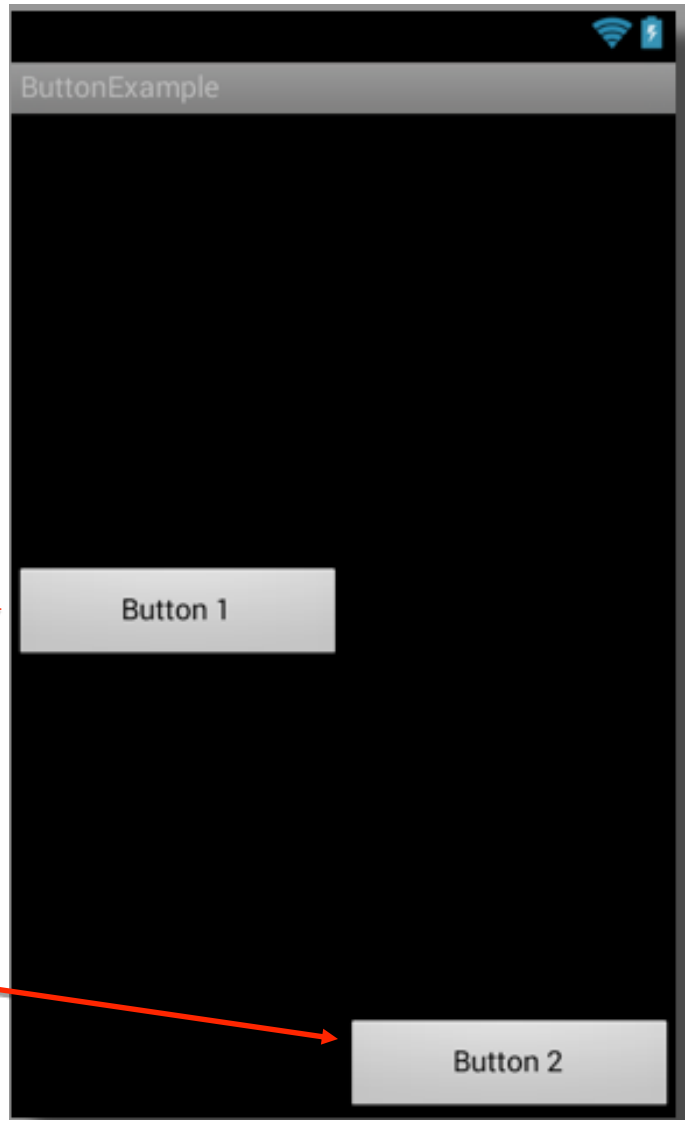
Gravity

- `gravity`: specifies how an object should position its content, on both the X and Y axes, within its own bounds (`top, bottom, right, left,…`) .
- `layout_gravity`: positions the view with respect to its parent (i.e. what the view is contained in).
  - A LinearLayout respects margins between children and the gravity (right, center, or left alignment) of each child.



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="horizontal" >
  <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1"
        android:layout_weight="1"
        android:layout_gravity="center_vertical" />
  <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:layout_weight="1"
        android:text="@string/buttonString2" />

</LinearLayout>
```

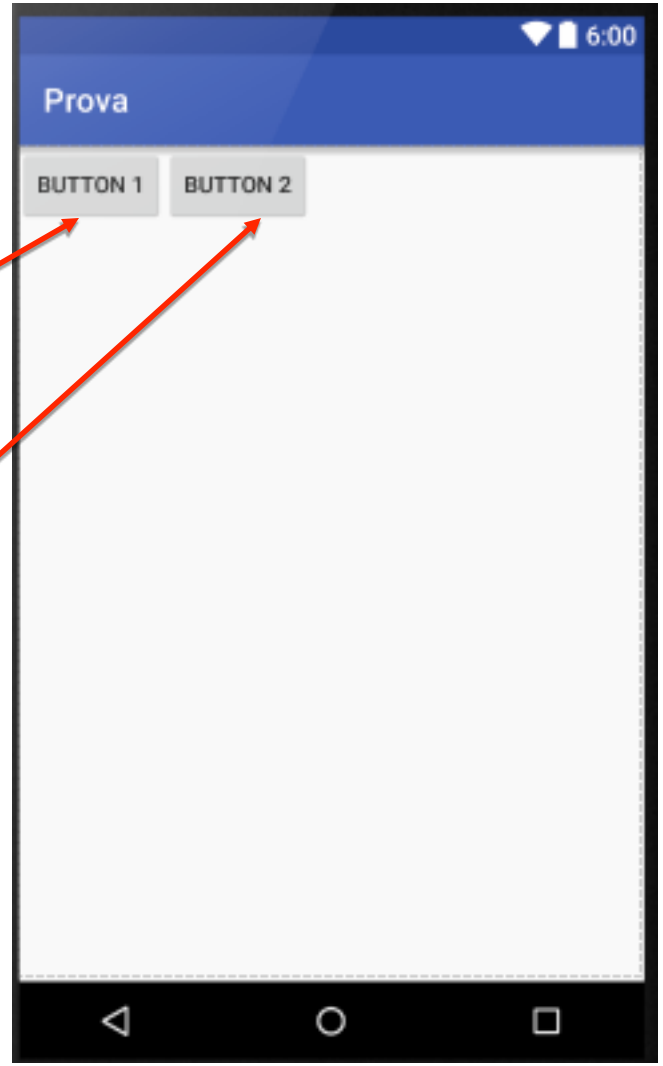


Linear Layout

- Also supports assigning a ***weight*** to ***individual children*** with the `android:layout_weight` attribute.
- This attribute assigns an ***"importance"*** value to a view in terms of ***how much space it should occupy*** on the screen.
  - A larger weight value allows it to expand to fill any remaining space in the parent view.
  - Child views can specify a weight value, and then any remaining space in the view group is assigned to children in the proportion of their declared weight.
  - Default weight is zero.

LinearLayout

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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1"
        android:layout_weight="0" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString2"
        android:layout_weight="0" />

</LinearLayout>
```



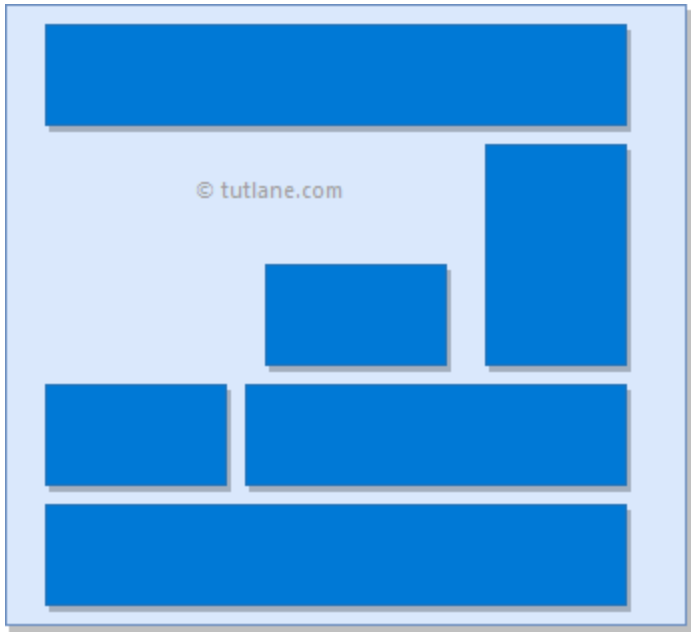
LinearLayout

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

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

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString2"
        android:layout_weight="1" />

</LinearLayout>
```

LinearLayout

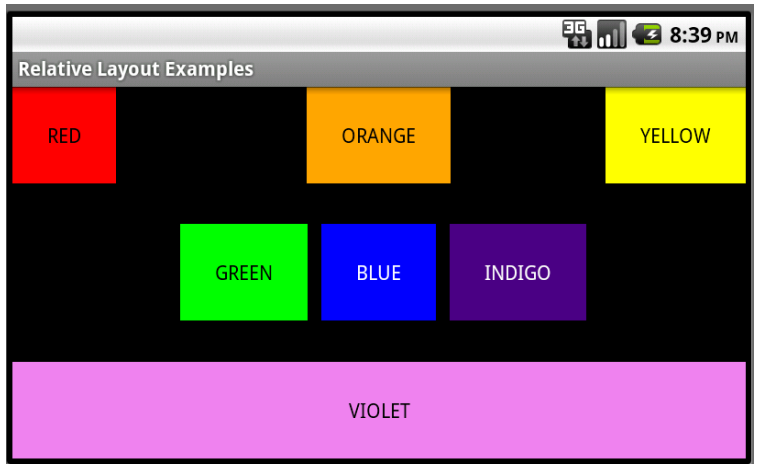
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout …>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString1"
        android:layout_weight="2" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/buttonString2"
        android:layout_weight="1" />

</LinearLayout>
```

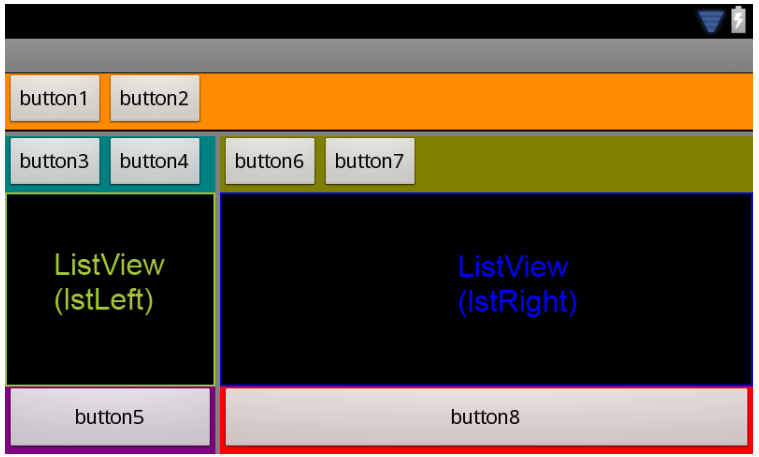
Relative layout



Relative Layout

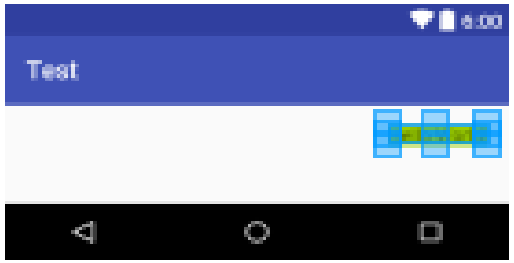
- Child views specify their position **relative** to the parent view or to each other (specified by ID).
- So you can **align** two elements by right border, or make one **below** another, **centered** in the screen, centered left, and so on.
- By default, all child views are drawn at the top-left of the layout, so you must define the position of each view using the various layout properties available from `RelativeLayout.LayoutParams`.

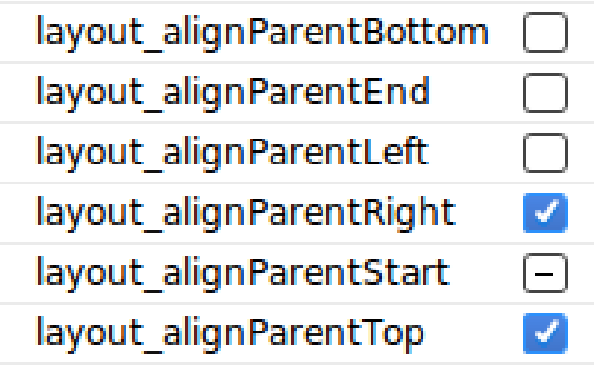




Relative Layout: some properties

- `android:layout_alignParentTop`: If "true", makes the top edge of this view match the top edge of the parent (attribute also for `Bottom`, `Right`, `Left`).
- `android:layout_centerVertical`: If "true", centers this child vertically within its parent (also `Horizontal`).
- `android:layout_below`: Positions the top edge of this view below the view specified with a ***resource ID*** (also `above`).
- `android:layout_toRightOf`: Positions the left edge of this view to the right of the view specified with a ***resource ID***. (also, `toLeftOf`, `toEndOf`, `toStartOf`).





Layout ID

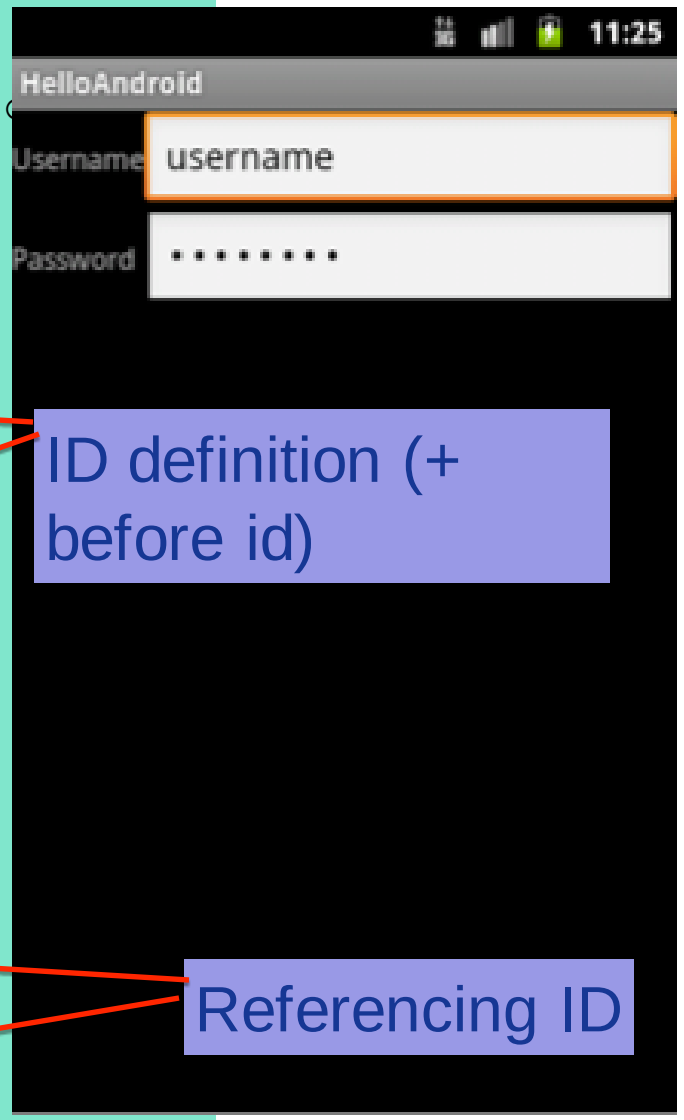
- Any ***View object may have an integer ID associated with it***, to uniquely identify the View within the tree.
  - It is assigned in the layout XML file as a string, in the `id` attribute.
  - Referenced as an integer via the `R.java` file.
- The syntax for an ID, inside an XML tag is:

```
android:id="@+id/my_button"
```

  - `@` indicates that the **XML parser** should parse and expand the rest of the ID string and identify it as an ID resource.
  - `+` means that this is a **new resource** name that must be created and added to our resources.
  - Referenced in XML without the plus-symbol.

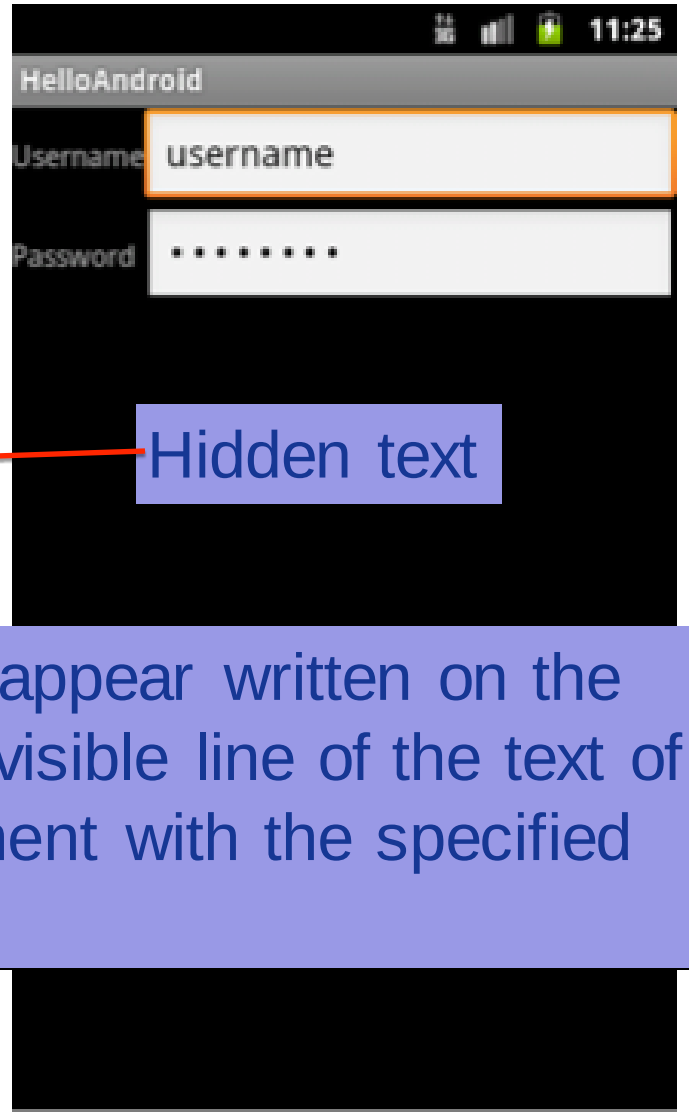
RelativeLayout

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

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

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



RelativeLayout

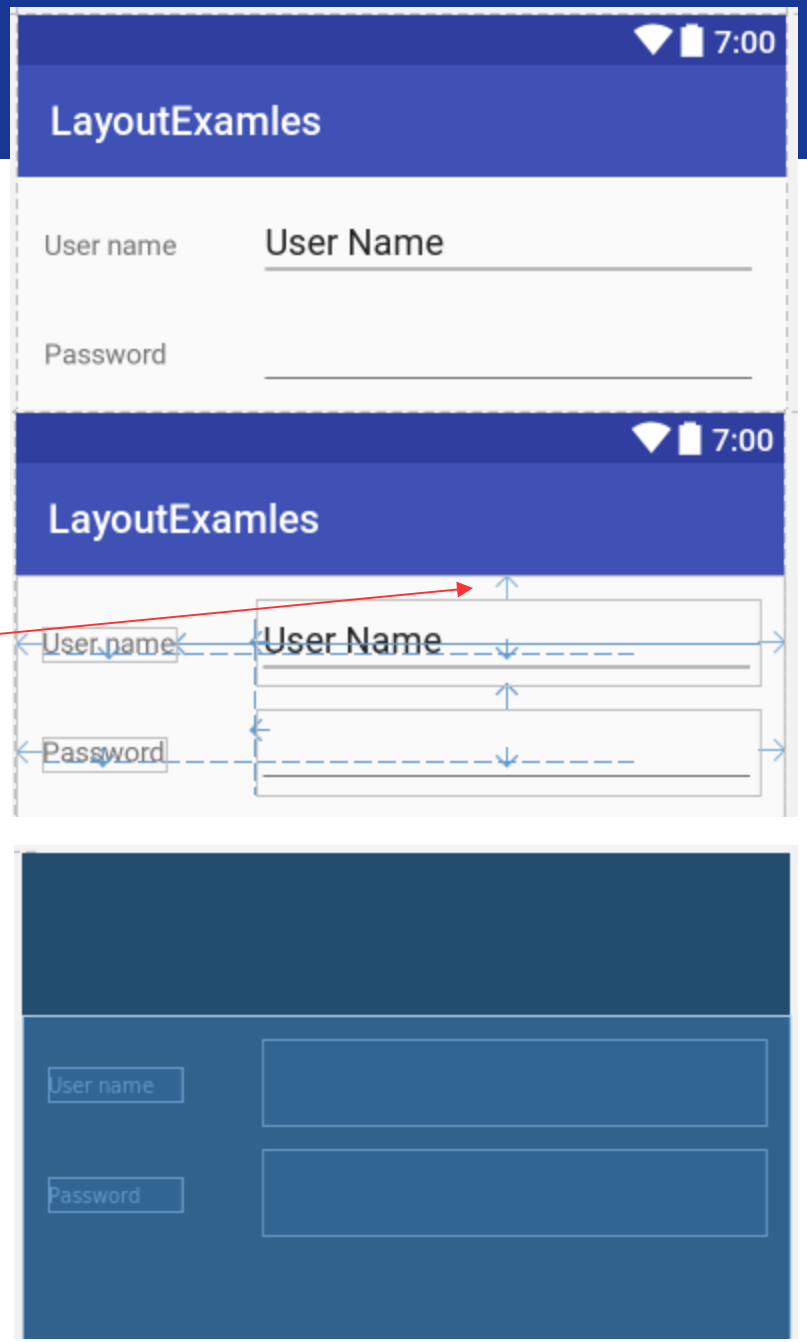
```
<EditText
    android:id="@+id/password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/username"
    android:layout_alignParentRight="true"
    android:layout_below="@id/username"
    android:layout_toRightOf="@id/usernameLabel"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

<TextView
    android:id="@+id/passwordLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@id/passwo
    android:text="@string/pwd" />
```



the text appear written on the same invisible line of the text of the element with the specified ID.

RelativeLayout: margins and align

```
<?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:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="User Name"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:layout_toEndOf="@+id/username_label"
    android:layout_marginEnd="13dp"
    android:layout_marginTop="12dp"
    android:layout_marginStart="40dp" />
  <TextView
    android:id="@id/username_label"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="13dp"
    android:layout_alignBaseline="@id/username"
    android:layout_alignParentLeft="true"
    android:text="User name" />
...
</RelativeLayout>
```



RelativeLayout: margins and align

```
<?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/edit_password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@id/username"
    android:layout_marginEnd="13dp"
    android:layout_marginTop="12dp"
    android:inputType="textPassword"
    android:layout_alignStart="@id/username" />
  <TextView
    android:id="@+id/password_label"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="13dp"
    android:layout_alignBaseline="@id/edit_password"
    android:layout_alignParentLeft="true"
    android:text="Password" />
</RelativeLayout>
```

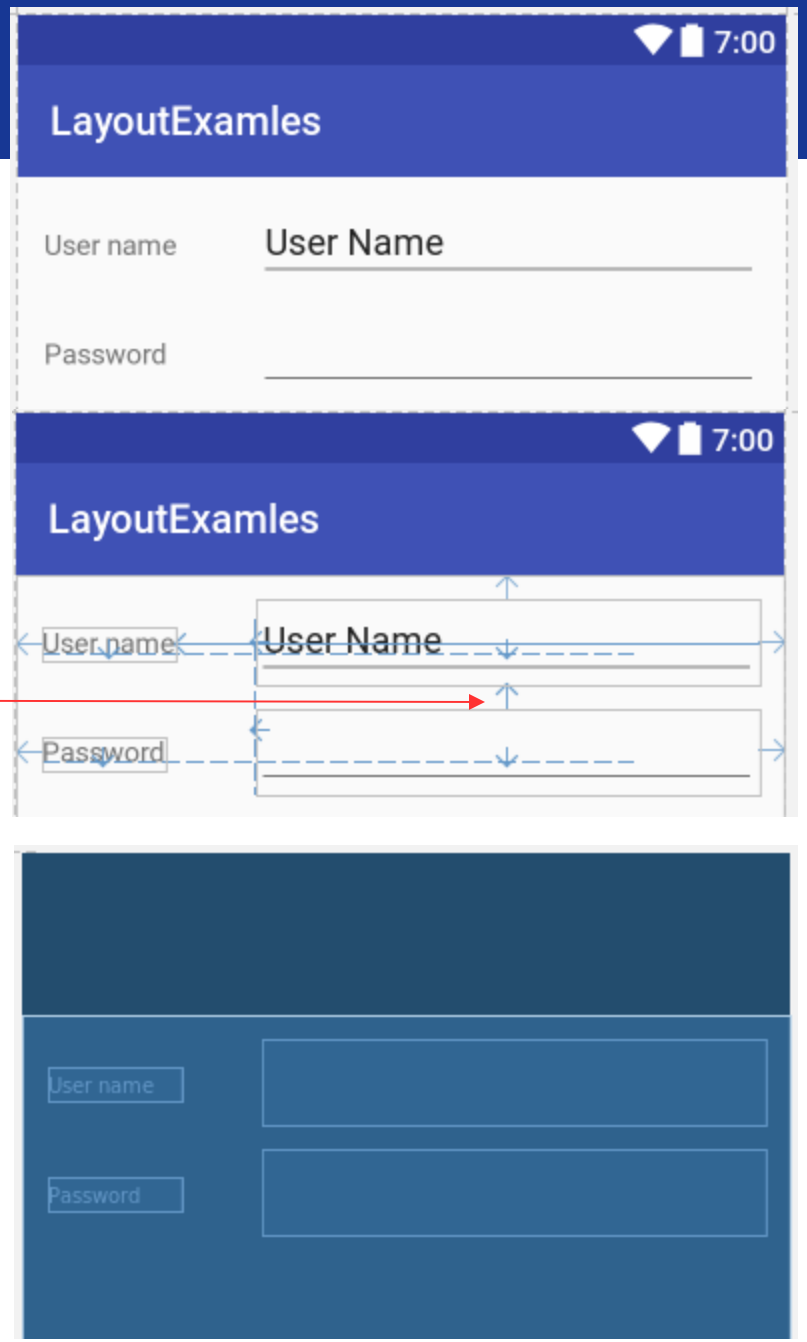


Table layout

TableLayout

- `TableLayout` positions its children into ***rows and columns*** (they do not display border lines for their rows, columns, or cells).
  - The table will have as many columns as the row with the most cells.
  - A table can leave cells empty, but cells cannot span columns.
- `TableRow` objects are the child views of a TableLayout (each TableRow defines a single row in the table).
  - Each row has zero or more cells, each of which is defined by any kind of other View.
  - A cell can be any `View` object.
  - A cell can be `ViewGroup` object (e.g., you can nest another `TableLayout` as a cell).

TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout … >
    <TableRow><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" />
    </TableRow>
    <TableRow><Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
           android:text="@string/buttonString3" />
    </TableRow>
</TableLayout>
```