User Interface

User Interface

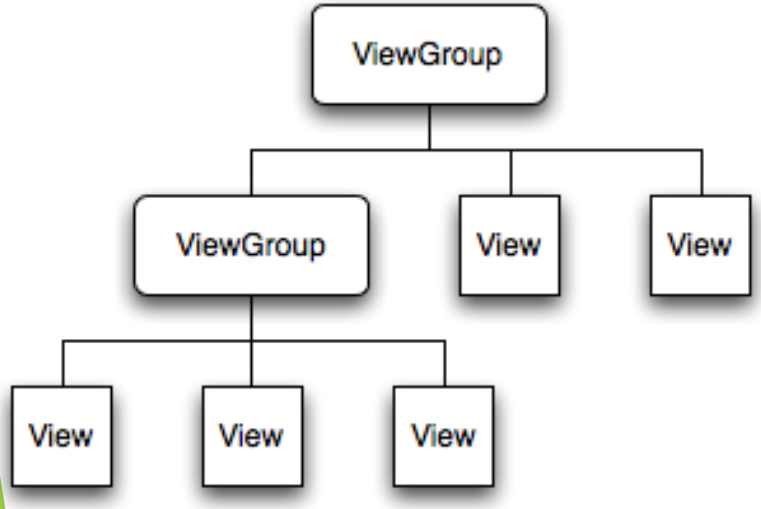
- In an Android application, the user interface is built using View and ViewGroup objects.
- The View class serves as the base for subclasses called "widgets," which offer fully implemented UI objects, like text fields and buttons.
- The ViewGroup class serves as the base for subclasses called "layouts," which offer different kinds of layout architecture, like linear, tabular and relative.

View

- A View object is a data structure whose properties store the layout parameters and content for a specific rectangular area of the screen.
- A View object handles its own measurement, layout, drawing, focus change, scrolling, and key/gesture interactions for the rectangular area of the screen in which it resides.
- As an object in the user interface, a View is also a point of interaction for the user and the receiver of the interaction events.

View Hierarchy

- An Activity's UI using a hierarchy of View and ViewGroup nodes



- In order to attach the view hierarchy tree to the screen for rendering, your Activity must call the setContentView() method and pass a reference to the root node object.
- The root node of the hierarchy requests that its child nodes draw themselves.
- The children may request a size and location within the parent, but the parent object has the final decision on where how big each child can be.

Layout

- Layout can be expressed using the layout XML file.
- Each element in XML is either a View or ViewGroup object.
- The name of an XML element is respective to the Java class that it represents.
  So a <TextView> element creates a TextView in your UI, and a <LinearLayout> element creates a LinearLayout view group.
- When you load a layout resource, the Android system initializes these run-time objects, corresponding to the elements in your layout.
- There are a variety of ways in which you can layout your views, E.g. LinearLayout, RelativeLayout, TableLayout, GridLayout etc.

Layout XML file

```
<?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:id="@+id/text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a TextView" />
      <Button android:id="@+id/button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Hello, I am a Button" />
    </LinearLayout>
```

Widget : Actionable element

- A widget is a View object that serves as an interface for interaction with the user.
- Android provides a set of fully implemented widgets, like buttons, checkboxes, and text-entry fields, so you can quickly build your UI. Some widgets provided by Android are more complex, like a date picker, a clock, and zoom controls.
- You can create own actionable elements i.e. widgets by defining your own View object or by extending and combining existing widgets.

UI Events

- To be informed of UI events, you need to do one of two things:
  - **Define an event listener and register it with the View.**
    The View class contains a collection of nested interfaces named On<*something*>Listener, each with a callback method called On<*something*>().
    For example, View.OnClickListener (for handling "clicks" on a View), View.OnTouchListener (for handling touch screen events in a View), and View.OnKeyListener (for handling device key presses within a View).
  - **Override an existing callback method for the View.**
    E.g. onKeyDown(). onTouchEvent() etc.