Name : Mujahid Ullah

Subject: Android Development

Submitted To Sir Junaid

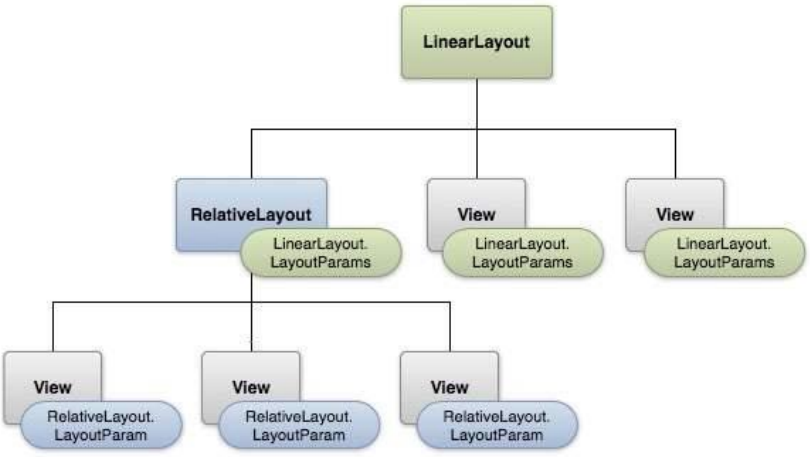
Date : 26-Dec-20

---

1. what is layout ?

The basic building block for user interface is a View object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.The ViewGroup is a subclass of View and provides invisible container that hold other Views or other ViewGroups and define their layout properties.

At third level we have different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using View/ViewGroup objects or you can declare your layout using simple XML file main_layout.xml which is located in the res/layout folder of your project.



This tutorial is more about creating your GUI based on layouts defined in XML file. A layout may contain any type of widgets such as buttons, labels, textboxes, and so on. Following is a simple example of XML file having 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" >

   <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a TextView" />

      <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a Button" />

   <!-- More GUI components go here  -->
</LinearLayout>
```

Once your layout has created, you can load the layout resource from your application code, in your Activity.onCreate() callback implementation as shown below −

```
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
}
```

Android Layout Types: There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.

Table Of Layout and Their Description

Sr.No	Layout & Description

1	**Linear Layout:**
	LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
	Example:
	`<?xml version="1.0" encoding="utf-8"?>`
	`<LinearLayout`
	`xmlns:android="http://schemas.android.com/apk/res/android"`
	`xmlns:app="http://schemas.android.com/apk/res-auto"`

	xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:layout_height="match_parent" android:orientation="vertical" tools:context=".courses.Chapter_detail"> </LinearLayout>
2	**Relative Layout:** RelativeLayout is a view group that displays child views in relative positions. Example: <?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:orientation="vertical" android:layout_height="match_parent" tools:context=".courses.Chapter_detail"> </RelativeLayout>
3	**Table Layout:** TableLayout is a view that groups views into rows and columns. Example: <?xml version="1.0" encoding="utf-8"?> <TableLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app="http://schemas.android.com/apk/res-auto" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:orientation="vertical" android:layout_height="match_parent" tools:context=".courses.Chapter_detail"> <TableRow android:layout_width="fill_parent" android:layout_height="fill_parent"> <TextView android:text="Time" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_column="1" /> </TableRow> </TableLayout>
4	**Absolute Layout:** AbsoluteLayout enables you to specify the exact location of its children. **Example:** <AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:layout_height="fill_parent"> <Button

	android:layout_width="100dp" android:layout_height="wrap_content" android:text="OK" android:layout_x="50px" android:layout_y="361px" /> <Button android:layout_width="100dp" android:layout_height="wrap_content" android:text="Cancel" android:layout_x="225px" android:layout_y="361px" /> </AbsoluteLayout>
5	**Frame Layout:** The FrameLayout is a placeholder on screen that you can use to display a single view. **Example:** <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:layout_height="fill_parent"> <ImageView android:src="@drawable/ic_launcher" android:scaleType="fitCenter" android:layout_height="250px" android:layout_width="250px"/> <TextView android:text="Frame Demo" android:textSize="30px" android:textStyle="bold" android:layout_height="fill_parent" android:layout_width="fill_parent" android:gravity="center"/> </FrameLayout>
6	**List View:** ListView is a view group that displays a list of scrollable items. **Example** <ListView android:id="@+id/mobile_list" android:layout_width="match_parent" android:layout_height="wrap_content" > </ListView>
7	**Grid View:** GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid. **Example:** <GridView android:id="@+id/mobile_list" android:layout_width="match_parent" android:layout_height="wrap_content" > </GridView>

Detail About Layout Attributes:

Following are the important attributes

Sr.No	Attribute & Description
1	**android:id** android:id="@+id/myId" This is the ID which uniquely identifies the layout.
2	**android:baselineAligned:** android:baselineAligned="true" This must be a boolean value, either "true" or "false" and prevents the layout from aligning its children's baselines.
3	**android:baselineAlignedChildIndex** android:baselineAlignedChildIndex="@integer/config_navAnimTime" When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align.
4	**android:divider** android:divider="@color/cardview_light_background" This is drawable to use as a vertical divider between buttons. You use a color value, in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb".
5	**android:gravity** android:gravity="center" This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
6	**android:orientation** This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal.
7	**android:weightSum** Sum up of child weight
8	**android:layout_width** This is the width of the layout.
9	**android:layout_height** This is the height of the layout
10	**android:layout_marginTop** This is the extra space on the top side of the layout.
11	**android:layout_marginBottom** This is the extra space on the bottom side of the layout.
12	**android:layout_marginLeft** This is the extra space on the left side of the layout.
13	**android:layout_margin** This is the extra space on the all side of the layout.
14	**android:layout_marginRight** This is the extra space on the right side of the layout.
15	**android:layout_x**

	This specifies the x-coordinate of the layout.
16	**android:layout_y** This specifies the y-coordinate of the layout.
17	**android:paddingLeft** This is the left padding filled for the layout.
18	**android:paddingRight** This is the Right padding filled for the layout.
19	**android:paddingTop** This is the Top padding filled for the layout.
20	**android:paddingButtom** This is the Bottom padding filled for the layout.

Here **width and height** are the dimension of the layout/view which can be specified in terms of dp (Density-independent Pixels), sp (Scale-independent Pixels), pt (Points which is 1/72 of an inch), px(Pixels), mm (Millimeters) and finally in (inches).You can specify width and height with exact measurements but more often, you will use one of these constants to set the width or height –

android:layout_width=wrap_content tells your view to size itself to the dimensions required by its content.

android:layout_width=fill_parent tells your view to become as big as its parent view.

Gravity attribute plays important role in positioning the view object and it can take one or more (separated by '|') of the following constant values.

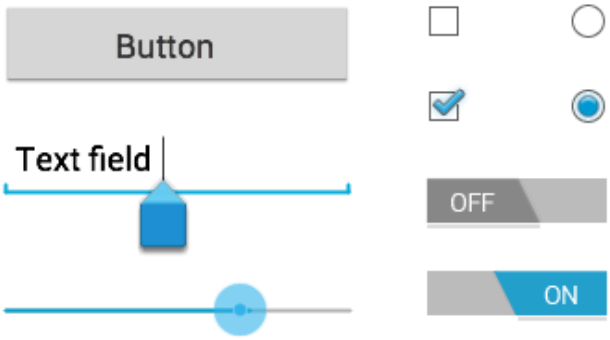
Constant	Value	Description
top	0x30	Push object to the top of its container, not changing its size.
bottom	0x50	Push object to the bottom of its container, not changing its size.
left	0x03	Push object to the left of its container, not changing its size.
right	0x05	Push object to the right of its container, not changing its size.
center_vertical	0x10	Place object in the vertical center of its container, not changing its size.
fill_vertical	0x70	Grow the vertical size of the object if needed so it completely fills its container.
center_horizontal	0x01	Place object in the horizontal center of its container, not changing its size.

fill_horizontal	0x07	Grow the horizontal size of the object if needed so it completely fills its container.
center	0x11	Place the object in the center of its container in both the vertical and horizontal axis, not changing its size.
fill	0x77	Grow the horizontal and vertical size of the object if needed so it completely fills its container.
clip_vertical	0x80	Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip will be based on the vertical gravity: a top gravity will clip the bottom edge, a bottom gravity will clip the top edge, and neither will clip both edges.
clip_horizontal	0x08	Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The clip will be based on the horizontal gravity: a left gravity will clip the right edge, a right gravity will clip the left edge, and neither will clip both edges.
start	0x00800003	Push object to the beginning of its container, not changing its size.
end	0x00800005	Push object to the end of its container, not changing its size.

2. What is UI or android widget ? Enlist all supported ui/widget with breifly explaination, attribute and methods?

In android UI or input controls are the interactive or View components that are used to design the user interface of an application. In android we have a wide variety of UI or input controls available, those are TextView, EditText, Buttons, Checkbox, Progressbar, Spinners, etc.

Following is the pictorial representation of user interface (UI) or input controls in android application.



Generally, in android the user interface of an app is made with a collection of View and ViewGroup objects.The View is a base class for all UI components in android and it is used to create interactive UI components such as TextView, EditText, Checkbox, Radio Button, etc. and it is responsible for event handling and drawing.The ViewGroup is a subclass of View and it will act as a base class for layouts and layout parameters. The ViewGroup will provide invisible containers to hold other Views or ViewGroups and to define the layout properties.To know more about View and ViewGroup in android applications, check this Android View and ViewGroup.

In android, we can define a UI or input controls in two ways, those are

Declare UI elements in XML

Create UI elements at runtime

The android framework will allow us to use either or both of these methods to define our application's UI.

Declare UI Elements in XML

In android, we can create layouts same as web pages in HTML by using default Views and ViewGroups in the XML file. The layout file must contain only one root element, which must be a View or ViewGroup object. Once we define the root element, then we can add additional layout objects or widgets as a child elements to build View hierarchy that defines our layout.

Following is the example of defining UI controls (TextView, EditText, Button) in the XML file (activity_main.xml) using LinearLayout.

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:orientation="vertical"

    android:layout_width="match_parent"

    android:layout_height="match_parent">

    <TextView

        android:id="@+id/fstTxt"

        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="Enter Name" />
  <EditText
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"/>
  <Button
    android:id="@+id/getName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Get Name" />
</LinearLayout>
```

Create UI Element at Runtime

If we want to create UI elements at runtime, we need to create our own custom View and ViewGroup objects programmatically with required layouts.Following is the example of creating UI elements (TextView, EditText, Button) in LinearLayout using custom View and ViewGroup objects in an activity programmatically.

```
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView textView1 = new TextView(this);
    textView1.setText("Name:");
    EditText editText1 = new EditText(this);
    editText1.setText("Enter Name");
    Button button1 = new Button(this);
    button1.setText("Add Name");
    LinearLayout linearLayout = new LinearLayout(this);
```

```
        linearLayout.addView(textView1);
        linearLayout.addView(editText1);
        linearLayout.addView(button1);
        setContentView(linearLayout);
    }
}
```

By creating a custom View and ViewGroups programmatically, we can define UI controls in layouts based on our requirements in android applications

Android Different Types of UI Controls

We have a different type of UI controls available in android to implement the user interface for our android applications.

Following are the commonly used UI or input controls in android applications.

1. TextView
2. EditText
3. AutoCompleteTextView
4. Button
5. ImageButton
6. ToggleButton
7. CheckBox
8. RadioButton
9. RadioGroup
10. ProgressBar
11. Spinner
12. TimePicker
13. DatePicker
14. SeekBar
15. AlertDialog
16. Switch
17. RatingBar.

I. **Android TextView:** In android, TextView is a user interface control that is used to display the text to the user.To know more about TextView control check this, Android TextView with Examples.

Example Android TextView:

Following is the sample way to define TextView control in XML layout file in android application.

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:text="Welcome to Tutlane"
    android:textColor="#86AD33"
    android:textSize="20dp"
    android:textStyle="bold" />
```

II. **Android EditText:** In android, EditText is a user interface control which is used to allow the user to enter or modify the text.To know more about EditText, check this Android EditText with Examples.

Example Android EditText:

```
<EditText
    android:id="@+id/txtSub"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Subject"
    android:inputType="text"/>
```

III. **Android AutoCompleteTextView:** In android, AutoCompleteTextView is an editable text view which is used to show the list of suggestions based on the user typing text. The list of suggestions will be shown as a dropdown menu from which the user can choose an item to replace the content of the textbox.To know more about AutoCompleteTextView, check this Android AutoCompleteTextView with Examples.

Example Android AutoCompleteTextView:

```
<AutoCompleteTextView
    android:id="@+id/autoComplete_Country"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

IV. **Android Button:** In android, Button is a user interface control that is used to perform an action when the user clicks or tap on it.To know more about Button in android check this, Android Buttons with Examples.

Example Android Button:

```
<Button
    android:id="@+id/addBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add" />
```

V. **Android Image Button :** In android, Image Button is a user interface control that is used to display a button with an image to perform an action when the user clicks or tap on it.Generally, the Image button in android looks similar as regular Button and perform the actions same as regular button but only difference is for image button we will add an image instead of text.To know more about Image Button in android check this, Android Image Button with Examples.

Example Android Image Button:

```
<ImageButton
    android:id="@+id/addBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/add_icon" />
```

VI. **Android Toggle Button:** In android, Toggle Button is a user interface control that is used to display ON (Checked) or OFF (Unchecked) states as a button with a light indicator.To know more about Toggle Button in android check this,

Android Toggle Button with Examples.

```
<ToggleButton
    android:id="@+id/toggle1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:layout_marginTop="120dp"
    android:checked="true"
    android:textOff="OFF"
    android:textOn="ON"/>
```

VII. **Android CheckBox:** In android, Checkbox is a two-states button that can be either checked or unchecked.To know more about CheckBox in android check this,

Android CheckBox with Examples.

```
<CheckBox
    android:id="@+id/chk1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="Java" />
```

VIII. **Android Radio Button:** In android, Radio Button is a two-states button that can be either checked or unchecked and it cannot be unchecked once it is checked.To know more about Radio Button in android check this,

Android Radio Button with Examples.

```
<RadioGroup
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Java"
        android:checked="true"/>
```

IX. **Android Radio Group:** In android, Radio Group is used to group one or more radio buttons into separate groups based on our requirements.In case if we group radio buttons using the radio group, at a time only one item can be selected from the group of radio buttons.To know more about Radio Group in android check this,

Android Radio Group with Examples.

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Java"/>
</RadioGroup>
```

X. **Android ProgressBar:** In android, ProgressBar is a user interface control which is used to indicate the progress of an operation.To know more about ProgressBar, check this

Android ProgressBar with Examples.

```
<ProgressBar
    android:id="@+id/pBar3"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minHeight="50dp"
    android:minWidth="250dp"
    android:max="100"
    android:indeterminate="true"
    android:progress="1" />
```

XI. **Android Spinner:** In android, Spinner is a drop-down list which allows a user to select one value from the list.To know more about Spinner, check this

Android Spinner with Examples.

```
<Spinner android:id="@+id/spinner1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

XII. **Android TimePicker:** In android, TimePicker is a widget for selecting the time of day, either in 24-hour or AM/PM mode.To know more about TimePicker, check this,

Android TimePicker with Examples.

```
<TimePicker android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

XIII. **Android DatePicker:** In android, DatePicker is a widget for selecting a date.To know more about DatePicker, check this Android DatePicker with Examples.We learn all android user interface (UI) controls in next chapters in detailed manner with examples.

```
<DatePicker android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Ref: https://www.tutlane.com/tutorial/android/android-ui-controls-textview-edittext-radio-button-checkbox