Practical 4

Programs related to different Layouts Linear, Relative, Table.

Linear Layout in Android

LinearLayout is a ViewGroup that is responsible for holding views in it. It is a layout that arranges its children i.e the various views and layouts linearly (one after another) in a single column(vertically) or a single row(horizontally).

Horizontal LinearLayout

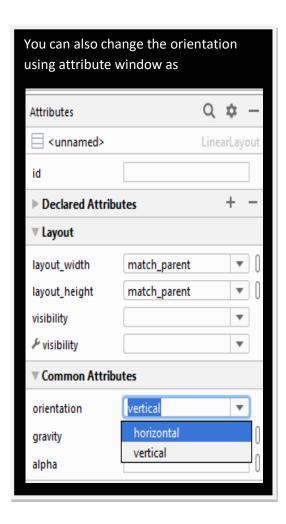
In a horizontal LinearLayout, as the name suggests, the Views defined inside the Linear Layout will be arranged horizontally one after another, like in a row. By default, the orientation is set to horizontal. But its a good practice to explicitly specify the orientation of the linear layout by setting the attribute android:orientation with value horizontal in the LinearLayout tag.

Vertical Linear Layout

In a vertical LinearLayout, as the name suggests, the Views defined inside the Linear Layout are arranged verically one after another, like in a column. And for this we need to mention the android:orientation attribute with value vertical within the LinearLayout tag.

Activity main.xml Remember <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"</p> 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" If you want horizontal layout just change android:orientation="vertical" orientation to horizontal tools:context=".MainActivity"> <EditText android:id="@+id/editText" android:layout width="wrap content" android:layout_height="86dp" android:layout_weight="1" android:ems="10" android:hint="Enter The Text" android:inputType="textPersonName" android:text="Type Here" android:textColor="#FFEB3B" />

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Button"/>
  <TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout height="wrap content"
    android:layout_weight="1"
    android:text="TextView"/>
  <TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout weight="1"
    android:text="TextView"/>
  <TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout weight="1"
    android:text="TextView"/>
</LinearLayout>
```



Note: Here We are getting background as black because we set the background color in style.xml in previous practical

```
<item name="android:background">#000000</item><item name="android:textColor">#FFFFAA</item>
```

Relative Layout

- RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left or center).
- A RelativeLayout is a very powerful utility for designing a user interface because it can eliminate nested view groups and keep your layout hierarchy flat, which improves performance. If you find yourself using several nested LinearLayout groups, you may be able to replace them with a single RelativeLayout.

Activity_main.xml

Remember

```
<?xml version="1.0" encoding="utf-8"?>
< Relative Layout 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=".MainActivity">
  <EditText
    android:id="@+id/name"
    android:layout width="match parent"
    android:layout_height="wrap_content"
    android:hint="Enter Text Here" />
  <Spinner
    android:id="@+id/dates"
    android:layout width="0dp"
    android:layout_height="wrap_content"
    android:layout below="@id/name"
    android:layout_alignParentLeft="true"
    android:layout_toLeftOf="@+id/times"/>
  <Spinner
    android:id="@id/times"
    android:layout width="96dp"
    android:layout height="wrap content"
    android:layout below="@id/name"
    android:layout_alignParentRight="true" />
  <Button
    android:layout width="96dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/times"
    android:layout_alignParentRight="true"
    android:text="Click Here"/>
```

Table Layout

Android TableLayout going to be arranged groups of views into rows and columns. You will use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object.

TableLayout containers 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. Cells can span multiple columns, as they can in HTML. You can span columns by using the span field in the TableRow.LayoutParams class.

```
Main activity.xml
                                                  Remember
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout width="match parent"
 android:layout_height="match_parent"
 android:stretchColumns="1">
  <TableRow>
    <TextView
      android:layout_width="225dp"
      android:layout height="56dp"
      android:padding="3dip"
      android:text="Row1,Col1"/>
    <TextView
      android:layout_height="match_parent"
      android:gravity="right"
      android:padding="3dip"
      android:text="Row1,Col2"/>
  </TableRow>
 <TableRow>
    <TextView
      android:layout width="242dp"
      android:layout height="64dp"
      android:padding="3dip"
      android:text="Row2,Col1"/>
    <TextView
      android:layout_height="match_parent"
      android:gravity="right"
      android:padding="3dip"
      android:text="Row2,Col2"/>
```

T.Y.BSc(I.T) SEM-VI	Advanced Mobile Programm