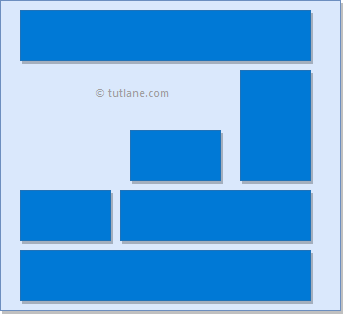
**Android RelativeLayout with Examples**

In android, **RelativeLayout** is a **ViewGroup** which is used to specify the position of child **View** instances relative to each other (Child **A** to the left of Child **B**) or relative to the parent (Aligned to the top of parent).

Following is the pictorial representation of relative layout in android applications.



In android, **RelativeLayout** is very useful to design user interface because by using relative layout we can eliminate the nested view groups and keep our layout hierarchy flat, which improves the performance of application.

**Android Positioning Views in Relative Layout**

As we discussed, in **RelativeLayout** we need to specify the position of child views relative to each other or relative to the parent. In case if we didn’t specify the position of child views, by default all child views are positioned to top-left of the layout.

Following are the some of most useful layout properties available to views in RelativeLayout.

| **Attribute** | **Description** |
| --- | --- |
| layout\_alignParentTop | If it specified “true”, the top edge of view will match the top edge of the parent. |
| layout\_alignParentBottom | If it specified “true”, the bottom edge of view will match the bottom edge of parent. |
| layout\_alignParentLeft | If it specified “true”, the left edge of view will match the left edge of parent. |
| layout\_alignParentRight | If it specified “true”, the right edge of view will match the right edge of the parent. |
| layout\_centerInParent | If it specified “true”, the view will be aligned to the centre of parent. |
| layout\_centerHorizontal | If it specified “true”, the view will be horizontally centre aligned within its parent. |
| layout\_centerVertical | If it specified “true”, the view will be vertically centre aligned within its parent. |
| layout\_above | It accepts another sibling view id and places the view above the specified view id. |
| layout\_below | It accepts another sibling view id and places the view below the specified view id. |
| layout\_toLeftOf | It accepts another sibling view id and places the view left of the specified view id. |
| layout\_toRightOf | It accepts another sibling view id and places the view right of the specified view id. |
| layout\_toStartOf | It accepts another sibling view id and places the view to start of the specified view id. |
| layout\_toEndOf | It accepts another sibling view id and places the view to the end of the specified view id. |

**Android RelativeLayout Example**

Following is the example of creating a **RelativeLayout** with different controls in android application.

Create a new android application using android studio and give names as **RelativeLayout**.

Now open an **activity\_main.xml** file from **\res\layout** path and write the code like as shown below

**activity\_main.xml**

<?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"  
    android:paddingLeft="10dp"  
    android:paddingRight="10dp">  
    <Button  
        android:id="@+id/btn1"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_alignParentLeft="true"  
        android:text="Button1" />  
    <Button  
        android:id="@+id/btn2"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_alignParentRight="true"  
        android:layout\_centerVertical="true"  
        android:text="Button2" />  
    <Button  
        android:id="@+id/btn3"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_alignParentLeft="true"  
        android:layout\_centerVertical="true"  
        android:text="Button3" />  
  
    <Button  
        android:id="@+id/btn4"  
        android:layout\_width="match\_parent"  
        android:layout\_height="wrap\_content"  
        android:layout\_alignParentBottom="true"  
        android:text="Button4" />  
    <Button  
        android:id="@+id/btn5"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_alignBottom="@+id/btn2"  
        android:layout\_centerHorizontal="true"  
        android:text="Button5" />  
    <Button  
        android:id="@+id/btn6"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_above="@+id/btn4"  
        android:layout\_centerHorizontal="true"  
        android:text="Button6" />  
    <Button  
        android:id="@+id/btn7"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:layout\_toEndOf="@+id/btn1"  
        android:layout\_toRightOf="@+id/btn1"  
        android:layout\_alignParentRight="true"  
        android:text="Button7" />  
</RelativeLayout>

Once we are done with creation of layout, we need to load the XML layout resource from our activity **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.tutlane.relativelayout** path and write the code like as shown below.

**MainActivity.java**

import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_main);  
    }  
}

If you observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout\_file\_name**. Here our xml file name is **activity\_main.xml** so we used file name **activity\_main**.

Generally, during the launch of our activity, **onCreate()** callback method will be called by the android framework to get the required layout for an activity.

**Output of Android RelativeLayout Example**

When we run above example using the android virtual device (AVD) we will get a result like as shown below.



This is how we can use RelativeLayout in android applications based on our requirements.