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.

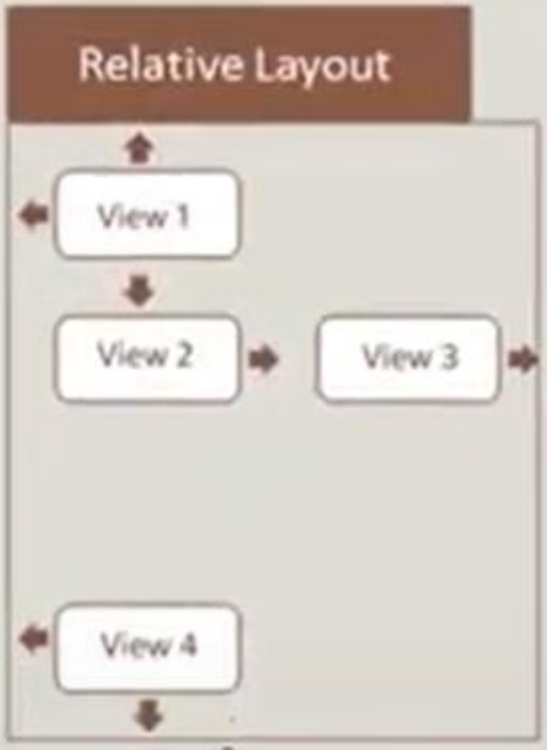
Relative Layout

- The Relative Layout is very flexible layout used in android for custom layout designing. It gives us the flexibility to position our component/view based on the relative or sibling component's position. Just because it allows us to position the component anywhere we want so it is considered as most flexible layout.
- For the same reason Relative layout is the most used layout after the Linear Layout in Android. It allow its child view to position relative to each other or relative to the container or another container.
- In Relative Layout, you can use "above, below, left and right" to arrange the component's position in relation to other component.

RELATIVE LAYOUT

&

ATTRIBUTES



Attributes


Relative Layout
With respect to parent -
• android:layout_alignParentTop
• android:layout_alignParentRight
• android:layout_alignParentBottom
• android:layout_alignParentLeft

Attributes

With respect to parent - (contd.)

- android:layout_centerInParent
- android:layout_centerHorizontal
- android:layout_centerVertical

Attributes

With respect to other views-

- android:layout_alignTop
- android:layout_alignRight
- android:layout_alignBottom
- android:layout_alignLeft

- **Positioning Views**
- RelativeLayout lets 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.
- Some of the many layout properties available to views in a RelativeLayout include:

  android:layout_alignParentTop
  - If "true", makes the top edge of this view match the top edge of the parent.
- android:layout_centerVertical
  - If "true", centers this child vertically within its parent.
- **android:layout_below**
  - Positions the top edge of this view below the view specified with a resource ID.
- **android:layout_toRightOf**
  - Positions the left edge of this view to the right of the view specified with a resource ID.
- The value for each layout property is either a boolean to enable a layout position relative to the parent RelativeLayout or an ID that references another view in the layout against which the view should be positioned.

Difference Between Relative Layout and Linear Layout

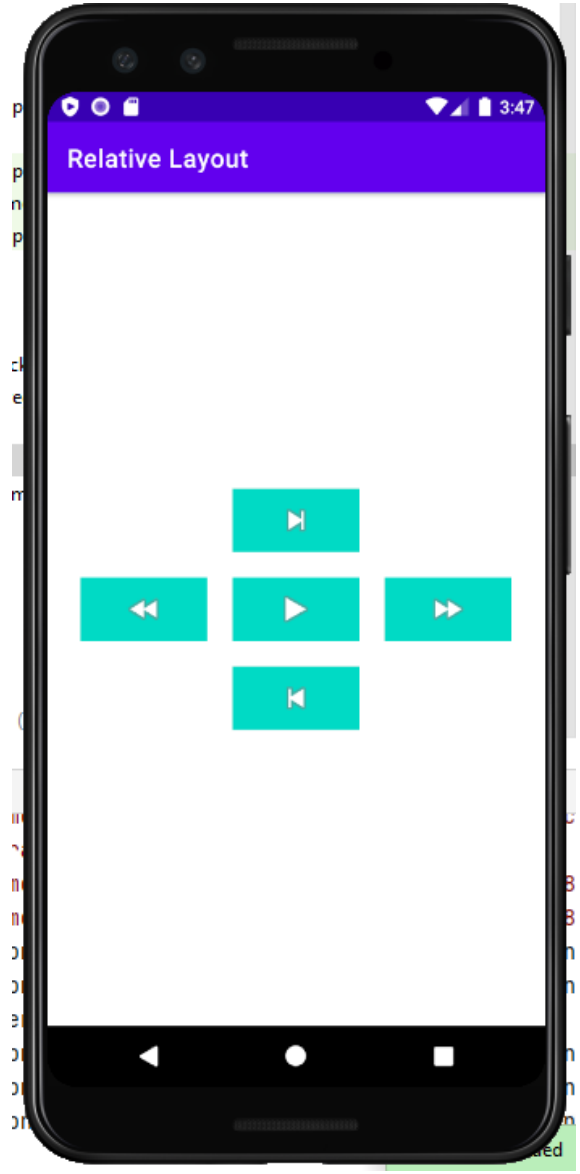
RELATIVE LAYOUT:

- Every element of relative layout arranges itself to the other element or a parent element.
- It is helpful while adding views one next to other etc
- In a relative layout you can give each child a Layout Property that specifies exactly where it should go in relative to the parent or relative to other children.
- Views can be layered on top of each other.

LINEAR LAYOUT:

- In a linear layout, like the name suggests, all the elements are displayed in a linear fashion either vertically or horizontally.
- Either Horizontally or Vertically this behavior is set in android:orientation which is an property of the node linear layout.
  android:orientation=”horizontal” or android:orientation=”vertical”
- Linear layouts put every child, one after the other, in a line, either horizontally or vertically.

Design the following User Interface using Relative Layout



activity_main.xml

```
<?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:layout_height="match_parent"
    android:gravity="center"
    tools:context=".MainActivity">

    <ImageButton
        android:id="@+id/imageButton"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:background="@color/teal_200"
        app:srcCompat="@android:drawable/ic_media_play" />

    <ImageButton
        android:id="@+id/imageButton2"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:layout_above="@+id/imageButton"
        android:layout_marginBottom="20dp"
        android:background="@color/teal_200"
        app:srcCompat="@android:drawable/ic_media_next" />
```

```
<ImageButton
        android:id="@+id/imageButton3"
        android:layout_width="100dp"
        android:layout_height="50dp"
 android:layout_below="@+id/imageButton"
        android:layout_marginTop="20dp"
        android:background="@color/teal_200"
        app:srcCompat="@android:drawable/ic_media_previous" />

    <ImageButton
        android:id="@+id/imageButton4"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:layout_marginRight="20dp"
        android:layout_toLeftOf="@+id/imageButton"
        android:background="@color/teal_200"
        app:srcCompat="@android:drawable/ic_media_ff" />

    <ImageButton
        android:id="@+id/imageButton5"
        android:layout_width="100dp"
        android:layout_height="50dp"
        android:layout_marginLeft="20dp"
        android:layout_toRightOf="@+id/imageButton"
        android:background="@color/teal_200"
        app:srcCompat="@android:drawable/ic_media_previous" />

</RelativeLayout>
```

References

- How to create image asset and add image button
  - https://www.youtube.com/watch?v=Wcdw-4tKgTo
- How to connect android phone to android studio
  - https://www.youtube.com/watch?v=apHlphSgC6o