# Views & ViewGroups:Basic Calculator App

**Prashanth B S**[1]

[1]Department of Information Science & Engineering, Nitte Meenakshi Institute of Technology, Yelahanka - 560064, Bengaluru

## 1 Views and ViewGroup

In Android Layout is used to describe the user interface for an app or activity, and it stores the UI elements that will be visible to the user. An android app's user interface is made up of a series of View and ViewGroup elements. In most cases, android apps will have one or more operations, each of which is a single screen of the app. Multiple UI components will be present in the operations, and those UI components will be instances of the View and ViewGroup subclasses[1]. View is a basic building block of UI (User Interface) in android. A view is a small rectangular box that responds to user inputs. Eg: EditText, Button, CheckBox, etc. ViewGroup is an invisible container of other views (child views) and other ViewGroup. Eg: LinearLayout is a ViewGroup that can contain other views in it. ViewGroup is a special kind of view that is extended from View as its base class. ViewGroup is the base class for layouts.

### 1.1 View

The View class is the base class or we can say that it is the superclass for all the GUI components in android. View refer to the android.view.View class, which is the base class of all UI classes. android.view.View class is the root of the UI class hierarchy. So from an object point of view, all UI objects are View objects.

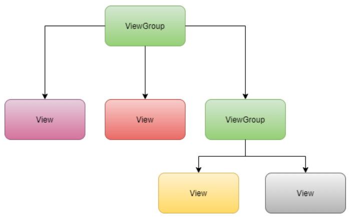
### 1.2 ViewGroup

The ViewGroup class is a subclass of the View class. And also it will act as a base class for layouts and layouts parameters. The ViewGroup will provide an invisible container to hold other Views or ViewGroups and to define the layout properties. For example, Linear Layout is the ViewGroup that contains UI controls like Button, TextView, etc., and other layouts also. A Viewgroup can also contain another viewgroups as well. The following figure 1a shows the relationship between the view and viewgroups.

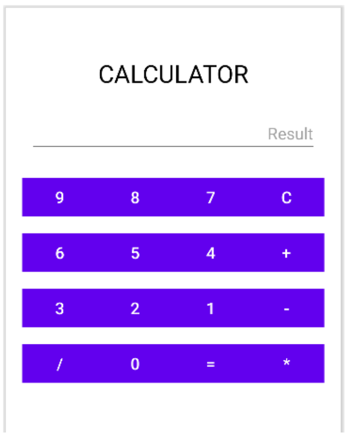
## 2 Exercise-I

**Create a Basic Calculator App(Binary) to demonstrate the usage of views and viewgroups.**
The application Basic calculator works only for binary operators and is not suited for the complex expressions. To evaluate the expression we use custom logic in conjunction with regex. The app is implemented using LinearLayout(ViewGroup) which is a parent for the Views such as Button, EditText, TextView and another LinearLayout as well. The front end of the calculator is as shown in the figure 1b,

[1]https://www.geeksforgeeks.org/difference-between-view-and-viewgroup-in-android/



(a) View and ViewGroups



(b) Calculator Design

Figure 1: Views for Calculator

The following are the steps,

- Follow the design shown in figure 2 for the activity_main.xml file and change the root element of the layout to LinearLayout and set the orientation to vertical as shown below,

```
<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"
    tools:context=".MainActivity"
    android:orientation="vertical"
    android:background="@color/white">
</LinearLayout>
```

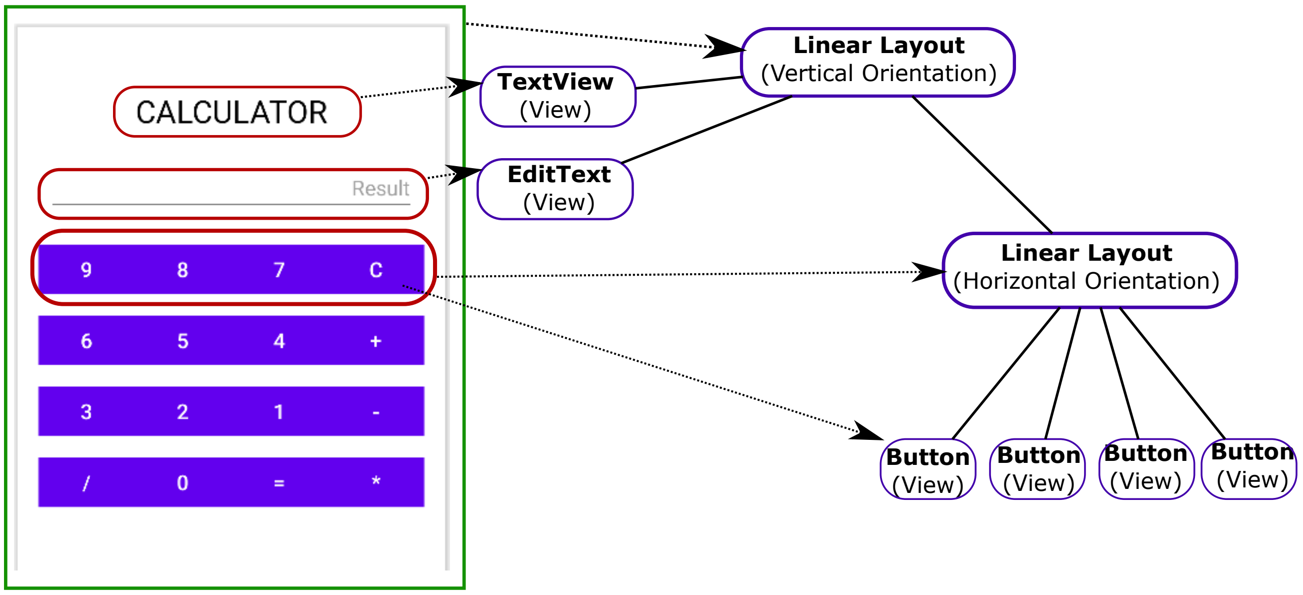


Figure 2: Layout Visualized

- Add an TextView(To display the name of the app), EditText(where the result of the calculation is shown as shown below,

```
<TextView
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:id="@+id/label"
        android:text="CALCULATOR"
        android:textSize="30dp"
        android:textColor="@color/black"
        android:gravity="center"
        />

<EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:hint="Result"
        android:gravity="end"
        android:textSize="20dp"
        android:id="@+id/res"
        />
```

- To implement the next set of widgets, we define another LinearLayout with horizontal orientation under the parent layout after EditText which will act as a container for the first row buttons such as 9,8,7 and C Buttons. Define the buttons and use *layout_weight* attribute to share the width of the layout to four buttons(0.25 each). The implementation is as shown below,

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:layout_marginBottom="20dp">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="9"
            android:textSize="20dp"
            android:id="@+id/nine"
            android:background="@color/white"
            android:layout_weight="0.25"
            />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="8"
            android:textSize="20dp"
            android:background="@color/white"
            android:id="@+id/eight"
            android:layout_weight="0.25"
            />
        <Button
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="7"
            android:textSize="20dp"
            android:background="@color/white"
            android:id="@+id/seven"
            android:layout_weight="0.25"
            />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="C"
            android:textSize="20dp"
            android:background="@color/white"
            android:id="@+id/clear"
            android:layout_weight="0.25"
            />
</LinearLayout>
```

- On the similar lines, Repeat the above steps to implement the rest of the rows along with the respective buttons complying to the UI design shown in 1b.
- Open the Mainactivity.java make it implement View.OnClickListener, for which one needs to override a function that is onClick(View v).

```java
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
// logic of onCreate()
//....
@Override
public void onClick(View view)
{
   //Logic to Handle various buttons
}
//...
```

- In the Mainactivity.java, Instantiate JAVA object class for each of the views defined in the layout as follows,

```java
Button one, two, three, four, five, six, seven, eight,nine;
Button plus, minus, div, sub, clear, equals;
EditText result;
String operatorPressed = " "; // keeps track of the operator pressed
```

- Inside the onCreate(), for each of the button defined above, equate the XML object with JAVA object using $findViewById(R.id.x)$ function as shown below,

```java
result = findViewById(R.id.expression); //EditText
one = findViewById(R.id.one) ;
two = findViewById(R.id.two) ; //Buttons
 // do it for the rest of the buttons for numbers and operators as well
```

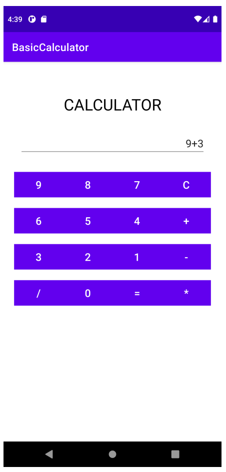
- Set the event listener for all of the buttons using this pointer as shown below,

```java
// Buttons
one.setOnClickListener(this);
two.setOnClickListener(this);
three.setOnClickListener(this);
four.setOnClickListener(this);
five.setOnClickListener(this);
six.setOnClickListener(this);
seven.setOnClickListener(this);
eight.setOnClickListener(this);
nine.setOnClickListener(this);
// operators
plus.setOnClickListener(this);
minus.setOnClickListener(this);
mult.setOnClickListener(this);
div.setOnClickListener(this);
equals.setOnClickListener(this);
clear.setOnClickListener(this);
```

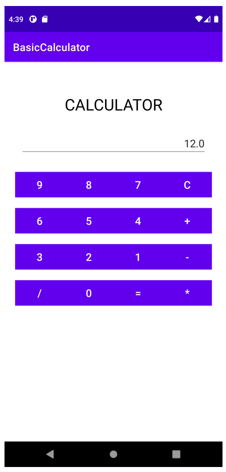
- Inside on the onClick override function, use the View.getID() and switch it to address the various buttons and perform the relevant actions(self commented)

```java
double finalResult = 0.0;
      switch(view.getId())
      {
          case R.id.one: res.append("1");
              break;
          case R.id.two: res.append("2");
              break;
          // case for rest of the button 3-9 here
          case R.id.plus: res.append("+");
              operatorPressed="+";
              break;
          case R.id.minus: res.append("-");
              operatorPressed="-";
              break;
          case R.id.mult: res.append("*");
              operatorPressed="*";
              break;
          case R.id.div: res.append("/");
              operatorPressed="/";
              break;
          case R.id.equals: finalResult=
              evaluateExpression(res.getText().toString(),operatorPressed);
              res.setText(String.valueOf(finalResult));
              break;
              // evaluateExpression function is the function we use to compute the
                  expression it takes the expression and operatorpressed as input and
                  returns a string
          default:return;
      }
```

- Implement te evaluateExpression as follows, explanation is in comments



(a) Calculator Expression input



(b) Result obtained on computation

Figure 3: Results of the Basic Calculator App

```
private double evaluateExpression(String res, String operatorPressed)
   {
       String[] tokens = res.split("\\+|-|\\*|\\/"); // split for +, -, *,/ operator
       // After split tokens[0] = first half of the string, tokens[1] = second half
           of the string
       double firstOperand = Double.parseDouble(tokens[0]); //convert string to double
       double secondOperand = Double.parseDouble(tokens[1]);
       switch(operatorPressed) // Switch the operator, compute and returs the result
           back to onclick that sets the editeext object to the result
       {
           case "+": return firstOperand + secondOperand;
           case "-": return firstOperand - secondOperand;
           case "*": return firstOperand * secondOperand;
           case "/": return firstOperand / secondOperand;
           default: return 0;
       }
   }
```

## 3 Output

The results of the app is shown in the figure 3. Figure 3a shows the landing page where as 3b shows the app screen after computing the expression.