Layout and Event Listeners

Part 1:

Start a new project. (API Level 14 – Android 4.0 Ice-cream Sandwich).

Part 2:

Layouts

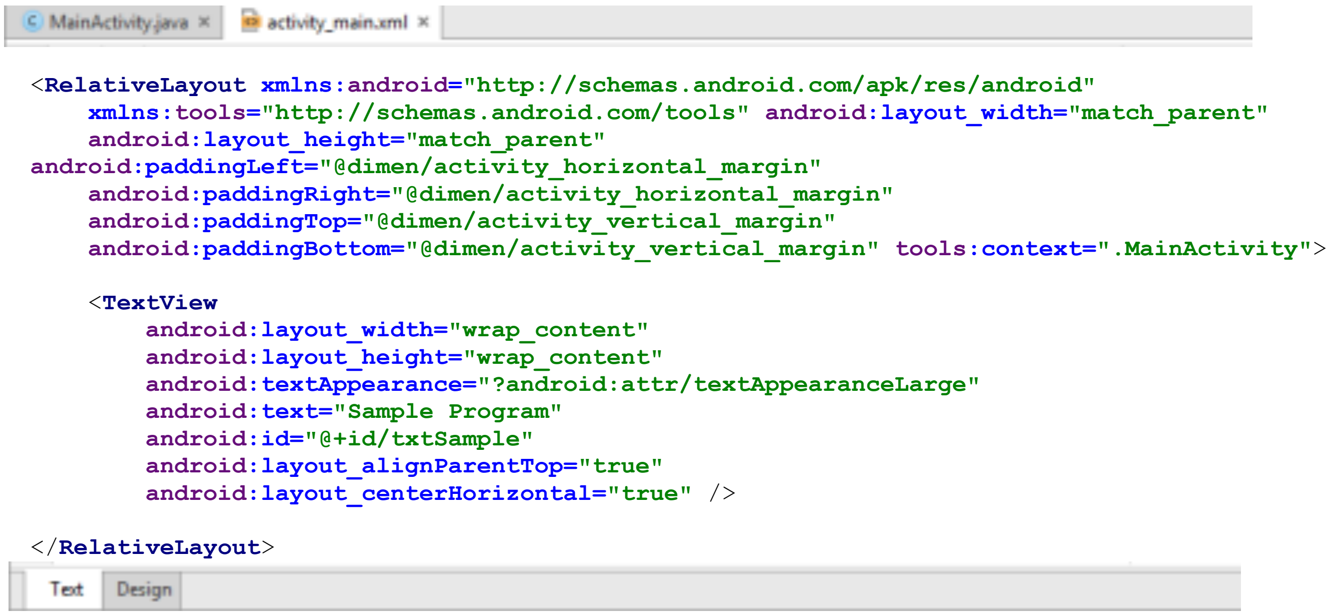
1. Relative Layout

As in name this layout positions elements relative to the adjacent elements.

It uses the following attributes for each element to position them:

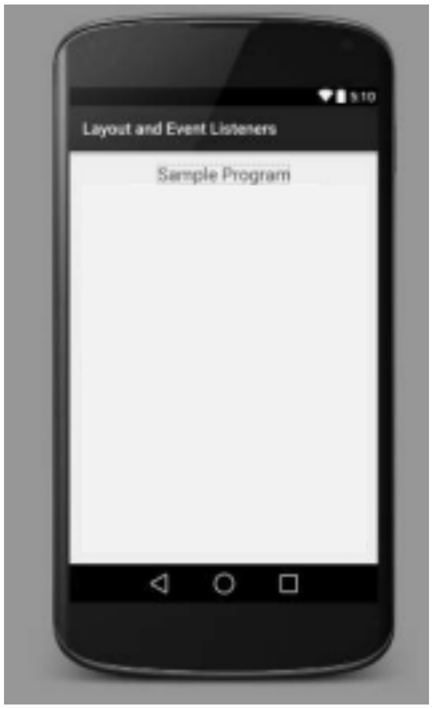
- layout:alignEnd
- layout:alignStart
- layout:toEndOf
- layout:toStartOf
- layout:alignParent
- layout:centreInParent

We will create a TextView (Large) inside the parent Relative layout by editing the xml code:



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Sample Program"
        android:id="@+id/txtSample"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Now we have created a TextView.



2. Linear Layout

Linear layout are two types Horizontal and Vertical.

Horizontal/Vertical is set using the **orientation** attribute.

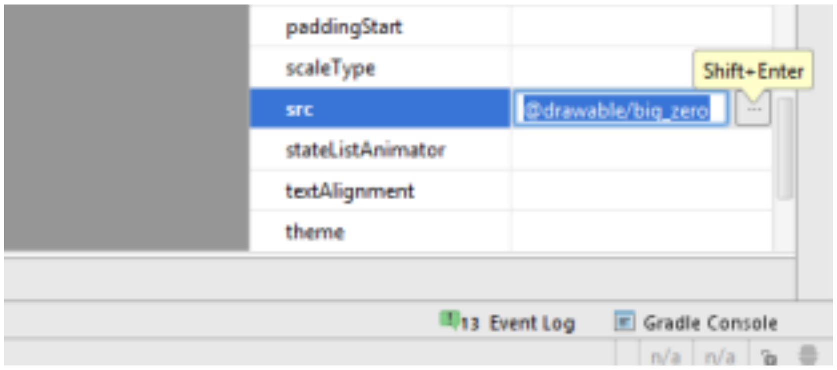
In such layout the elements are arranged in order top-to-bottom or left-to-right.

Let's add a Linear Layout now. (Now you can use the drag and drop layout editor). Change orientation to Vertical.

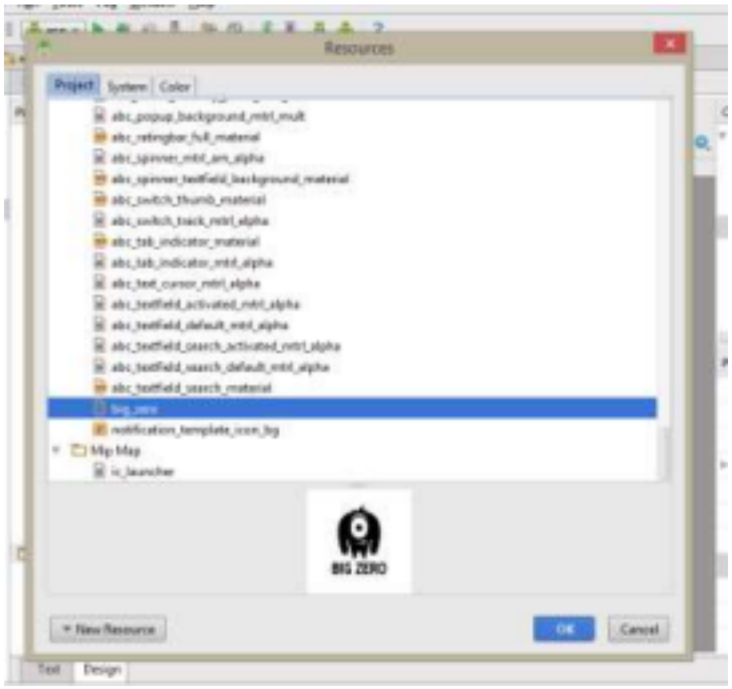
Now add an ImageView to the Linear Layout.

Import an image to the drawable directory. (Just as we have imported font-face in previous chapter).

Set the **src** attribute to the drawable we imported. (Click the browse button and select the file from Drawable directory).



paddingStart
scaleType
src @drawable/big_zero
stateListAnimator
textAlignment
theme



Resources
Project
System
Color
abc_popup_background_mtrl_mult
abc_ratingbar_full_material
abc_spinner_mtrl_am_alpha
abc_spinner_textfield_background_material
abc_switch_thumb_material
abc_switch_track_mtrl_alpha
abc_tab_indicator_material
abc_tab_indicator_mtrl_alpha
abc_text_cursor_mtrl_alpha
abc_textfield_activated_mtrl_alpha
abc_textfield_default_mtrl_alpha
abc_textfield_search_activated_mtrl_alpha
abc_textfield_search_default_mtrl_alpha
abc_textfield_search_material
big_zero
notification_template_icon_bg
Mip Map
ic_launcher
BIG ZERO
New Resource
OK
Cancel
Text
Design

3. **Table Layout:**

As we all know table layout uses rows and columns to position elements.

Add table layout inside the linear layout. Table layout uses TableRow layout to create rows.

Add a TableRow to the TableLayout. Add two Buttons to the TableRow.

Change the Id's of the two Buttons to ***btnClick*** and ***btnLongClick*** respectively.

we will use these buttons to implement event listeners

Change **text** to *Click Me!* and *Long Click Me!* also.

Select one of the buttons from the component tree. Pay attention to **Properties** window. You can see **layout:span** and **layout:column** attributes. The table layout uses these attributes to position elements.

If the values are unset then uses default values (span=1 and column increments according to order of placement).

4. Grid Layout

This is a very useful layout. This layout has order as well as freedom.

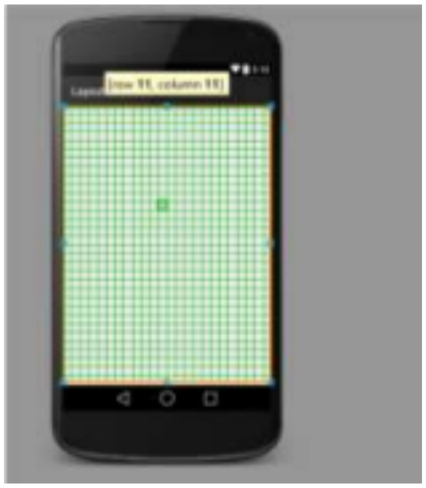
This layout uses orderly grids with rows and columns , span and spaces.

Add a **GridLayout** below the table layout.

Now drag and drop a Button to the GridLayout.

You'll see a green grid with many blocks.

<u>Example:</u>



Select this button and you can see that it uses attributes **layout:column, layout:row, layout:rowSpan, layout:columnSpan.** These are the attributes to position to items in GridLayout.

Change the rowSpan to 3. resize the button

Add one more button and a text field.

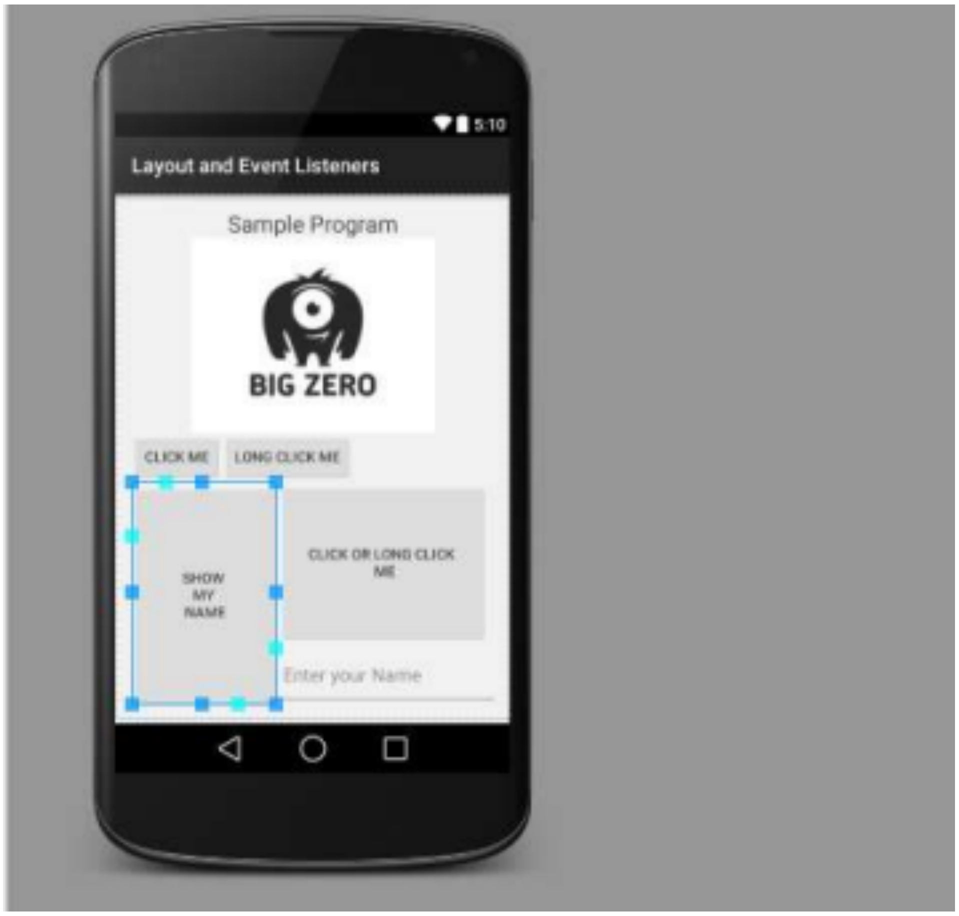
Edit the xml file to position them correctly:

```
<GridLayout
    android:layout_width="match_parent"
    android:layout_height="fill_parent"
    android:id="@+id/lytGrid">

    <Button
        android:layout_width="202dp"
        android:layout_height="156dp"
        android:text="Click or Long CLick \n Me"
        android:id="@+id/btnAll"
        android:layout_column="3"
        android:layout_row="0"
        android:layout_columnSpan="1"
        android:layout_rowSpan="2" />
    <Button
        android:layout_width="143dp"
        android:layout_height="match_parent"
        android:text="Show \nMy \nName"
        android:id="@+id/btnShowName"
        android:layout_row="1"
        android:layout_column="2"
        android:layout_rowSpan="3" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="62dp"
        android:id="@+id/txtName"
        android:layout_row="3"
        android:layout_column="3"
        android:hint="Enter your Name"
        android:layout_columnSpan="1"
        android:layout_rowSpan="1" />

</GridLayout>
```

As you see we changed the id's of the elements since we are going to use these to implement event listeners as well.

Final XML:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Sample Program"
        android:id="@+id/txtSample"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:layout_below="@+id/txtSample"
        android:layout_centerHorizontal="true">


        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/imageView"
            android:layout_gravity="center_horizontal"
            android:src="@drawable/big_zero" />

        <TableLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <TableRow
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:id="@+id/tableRow"
                android:orientation="horizontal">

                <Button
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Click Me"
                    android:id="@+id/btnClick" />

                <Button
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Long Click Me"
                    android:id="@+id/btnLongClick" />
            </TableRow>
        </TableLayout>

        <GridLayout
            android:layout_width="match_parent"
            android:layout_height="fill_parent"
            android:id="@+id/lytGrid">

            <Button
                android:layout_width="202dp"
                android:layout_height="156dp"
                android:text="Click or Long CLick \n Me"
                android:id="@+id/btnAll"
                android:layout_column="3"
                android:layout_row="0"
                android:layout_columnSpan="1"
                android:layout_rowSpan="2" />

            <Button
                android:layout_width="143dp"
                android:layout_height="match_parent"
                android:text="Show \nMy \nName"
                android:id="@+id/btnShowName"
                android:layout_row="1"
                android:layout_column="2"
                android:layout_rowSpan="3" />

            <EditText
                android:layout_width="match_parent"
                android:layout_height="62dp"
                android:id="@+id/txtName"
                android:layout_row="3"
                android:layout_column="3"
                android:hint="Enter your Name"
                android:layout_columnSpan="1"
                android:layout_rowSpan="1" />

        </GridLayout>
    </LinearLayout>
</RelativeLayout>
```


5:10
Layout and Event Listeners
Sample Program
BIG ZERO
CLICK ME
LONG CLICK ME
SHOW MY NAME
CLICK OR LONG CLICK ME
Enter your Name

Part 3: Event Listeners

We will discuss about two most commonly used event listeners – **onClickEventListener()** and **onLongClickEventListener().**

Step 1:

First we need to define some variables for each items in the UI.

```
Button clickBtn, longClickBtn, allBtn, btnShow;
TextView sample;
EditText nameTxt;
```

Step 2:

Assign the UI elements to these variables using findViewById()

```
clickBtn = (Button) findViewById(R.id.btnClick);
longClickBtn = (Button) findViewById(R.id.btnLongClick);
allBtn = (Button) findViewById(R.id.btnAll);
btnShow = (Button) findViewById(R.id.btnShowName);
sample = (TextView) findViewById(R.id.txtSample);
nameTxt = (EditText) findViewById(R.id.txtName);
```

Step 3:

Implementing the onClick Listener:

```
/*Simple Click onClick Listener*/
clickBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(getApplicationContext(), "Hai fella!", Toast.LENGTH_SHORT).show();
    }
});
```

-This event listener toasts a message Hai Fella when the “Click Me!” button is clicked

Step 4:

Implementing the onLongClick Listener

```
/*Implement Long Click Listener*/
longClickBtn.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {

        Toast.makeText(getApplicationContext(), "Hai there!",
Toast.LENGTH_SHORT).show();
        return false;
    }
});
```

- This event listener toasts a message “Hai there!” when the “Long Click Me!” button is clicked and held.

Step 5:

Implementing onClick and onLongClick Event on the same button.

```
allBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(getApplicationContext(), "You Just Clicked Me!",
Toast.LENGTH_SHORT).show();
    }
});

allBtn.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        Toast.makeText(getApplicationContext(), "You clicked me for so long!",
Toast.LENGTH_SHORT).show();
        return false;
    }
});
```

The button defined by the variable **allBtn** will toast two different messages when clicked and long-clicked i.e, *“You just Clicked Me!”* when clicked and *“You clicked me for so long!”* when long clicked.

Step 6:

Reading a data from a text field and writing it to a text view using event listeners.

```
btnShow.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        sample.setText(nameTxt.getText().toString());
    }
});
```

This event reads the value of the **nameTxt** text field and writes it to **sample** TextView.

That’s all for this chapter. Hope it helps.