LESSON – 3

Layout, Activity and Basic Components

Agenda

- Activity Life Cycle
- Basic UI Controls
- Layouts
  - Linear Layout
  - Relative Layout
  - Table Layout
- Click Event
- Hands on Examples

Activity Life Cycle

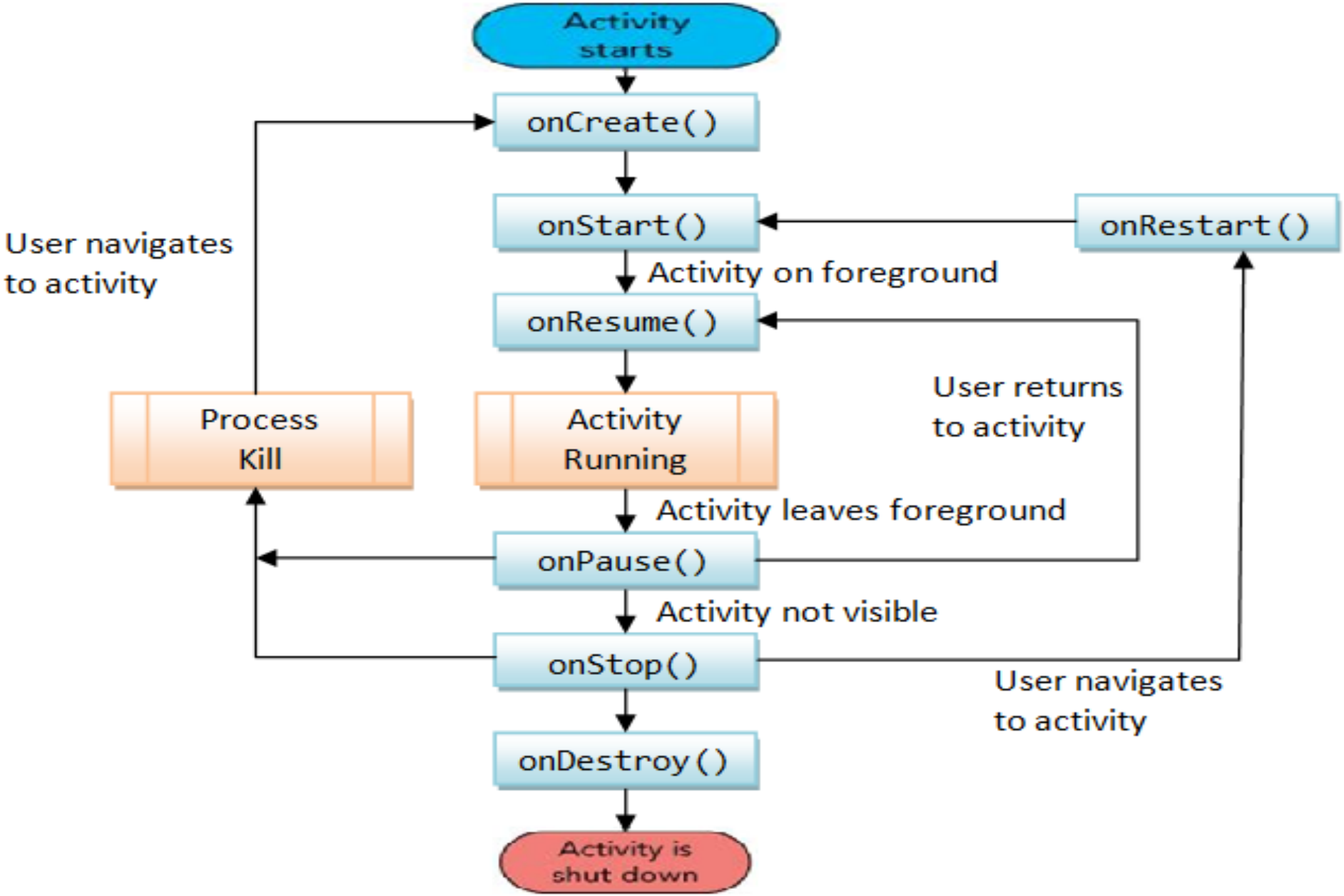
- The activity base class defines a series of events that govern the life cycle of an activity.

The `Activity` class defines the following events:

- `onCreate()`—Called when the activity is first created
- `onStart()`—Called when the activity becomes visible to the user
- `onResume()`—Called when the activity starts interacting with the user
- `onPause()`—Called when the current activity is being paused and the previous activity is being resumed
- `onStop()`—Called when the activity is no longer visible to the user
- `onDestroy()`—Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- `onRestart()`—Called when the activity has been stopped and is restarting again

By default, the activity created for you contains the `onCreate()` event. Within this event handler is the code that helps to display the UI elements of your screen.

Activity Life Cycle



States of Activity Life cycle

1. Active State: [Activity in the Foreground/Running]
   - When an Activity is present at the top of the stack, it is the currently visible and focused Activity and all the user inputs are provided to it.
   - In order to keep this Activity active, Android provides it with all the required resources and also terminates the previous activities if required.
   - When another Activity becomes active, this Activity will be pushed to the paused state.

2. Paused State:

- An Activity in this state is visible, but another Activity will have the focus and is present in the foreground.
- An Activity in the paused state is treated in the same way as it was treated when it was in the active state.
- The only difference is that it will not receive any user input.
- In serious cases, Android terminates a paused Activity to ensure availability of resources to the current Activity.
- An Activity is stopped when it becomes totally obscure.

3. Stopped State: [Activity in the background]
   - In this state, the Activity is not visible. It is however, present in the memory with all the state information.
   - All such activities are now ready for termination when the system requires memory.
   - When an Activity is stopped, it's data and UI information needs to be saved.
   - An Activity becomes inactive when it is closed or exited.
4. Inactive State: [Activity Doesn't exist]
   - When the Activity is no longer in the memory, it is said to be in the inactive state.
   - An Activity goes to the inactive state when it is terminated.
   - All such activities need to be restarted before they are used again.

- Example : LifeCycleAcivityDemo Package from Lesson3

Hands on Example : Life Cycle Activity

Now we are going to Override all these methods to know the activity life cycle and always call up to superclass when implementing these methods. The we are displaying the Log message using Log.i(String TAG,String msg) by import android.util.Log;

```
public class MActivity extends Activity{
    protected void onCreate(Bundle savedInstanceState);
    protected void onStart();
    protected void onRestart();
    protected void onResume();
    protected void onPause();
    protected void onStop();
    protected void onDestroy();
 }
```

- Refer: Lesson3_LogCat Filter Steps.pdf file to learn how to create logcat from Sakai Lectures/Lesson3 folder

MainActivity.java

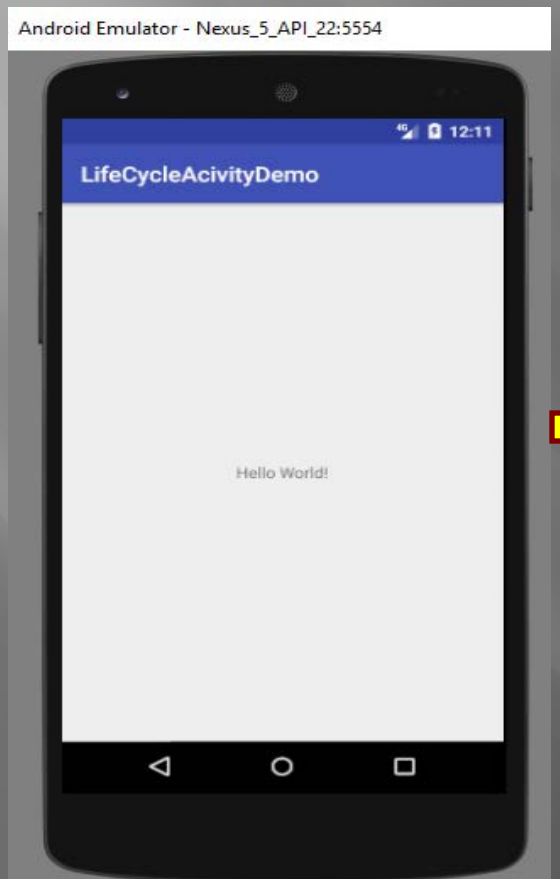
```java
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends
AppCompatActivity {

   public static final String MY_TAG =
"lifecycle";
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.i(MY_TAG,"Method in OnCreate");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.i(MY_TAG,"Method in OnStart");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.i(MY_TAG,"Method in OnResume");
    }
```

```java
@Override
    protected void onPause() {
        super.onPause();
        Log.i(MY_TAG,"Method in OnPause");
    }
    @Override
    protected void onStop() {
        super.onStop();
       Log.i(MY_TAG,"Method in OnStop");
    }
    @Override
    protected void onRestart() {
        super.onRestart();
        Log.i(MY_TAG,"Method in
OnRestart");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
       Log.i(MY_TAG,"Method in
OnDestroy");
    }
}
```

Activity Life Cycle – Screen Shots

- Lifetime of an Activity is from onCreate() to onDestroy()
- Activity is Visible when onStart() to onStop()
- Activity is in Foreground onResume to onPause()

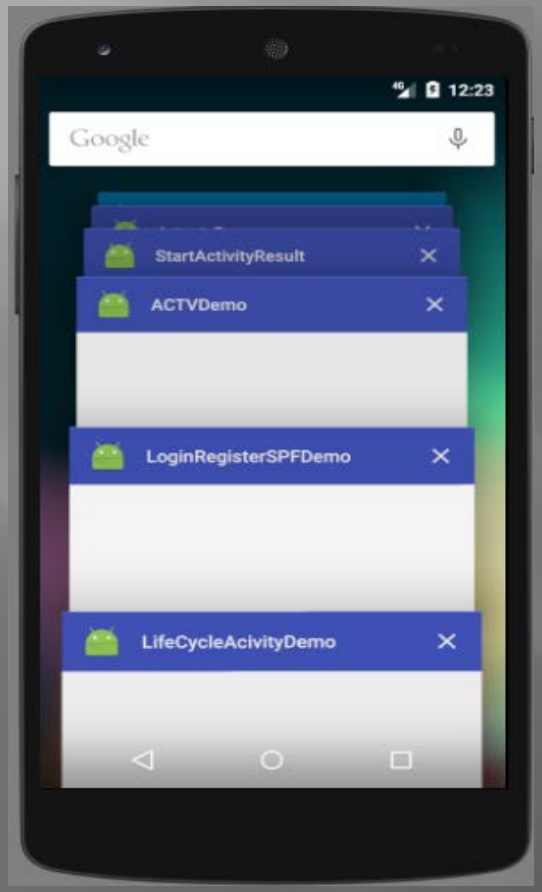
Screen 1 : After running the App, you will get the below screen and the Log message. The Activity is started by invoking 1. onCreate(), 2. onStart() and 3. onResume(). It is visible in the foreground.




```
I/lifecycle: Method in OnCreate
I/lifecycle: Method in OnStart
I/lifecycle: Method in OnResume
```

Activity Life Cycle – Screen Shots

Screen 2 : Now press the highlighted  Pass button from your Emulator. The Activity is paused by invoking
1. OnPause() and 2. onStop(). Now it is not visible and not destroyed.

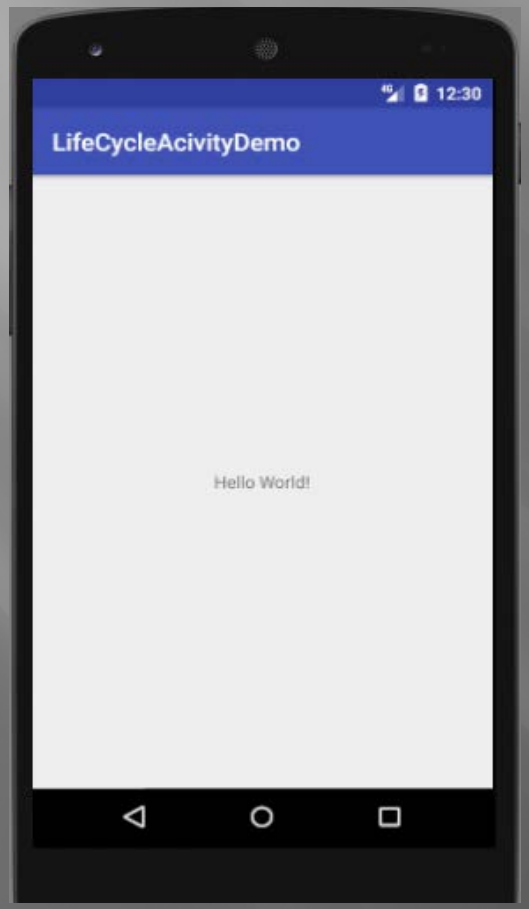



```
I/lifecycle: Method in OnPause
I/lifecycle: Method in OnStop
```

Activity Life Cycle – Screen Shots

Screen 3 : Again start your application by clicking highlighted start button the from your Emulator and reload the same app.

ty is restarted by invoking 1. OnRestart() 2. onStart() and 3. OnResume() itself. Now it is visible and not destroyed.




```
I/lifecycle: Method in OnRestart
I/lifecycle: Method in OnStart
I/lifecycle: Method in OnResume
```

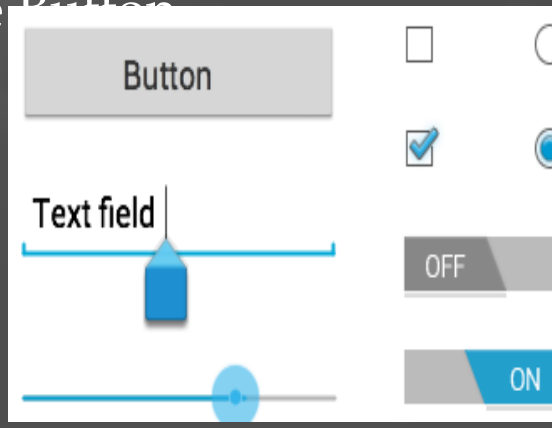
Activity Life Cycle – Screen Shots

Screen 4 : If you press the highlighted  back button from your Emulator your activity is destroyed by invoking 1. onPause() 2. OnStop() and 3. onDestroy(). The lifetime of the activity becomes over.


```
I/lifecycle: Method in OnPause
I/lifecycle: Method in OnStop
I/lifecycle: Method in OnDestroy
```

Basic UI Controls

- Input controls are the interactive components in your app's user interface.
- Android provides a wide variety of controls you can use in your UI, such as buttons, text fields, seek bars, checkboxes, zoom buttons, toggle buttons, and many more.
- Each input control supports a specific set of input events so you can handle events such as when the user enters text or touches a button.
- Adding an input control to your UI is as simple as importing the control class into your .java file and creating an object from the class.
- For instance, to create a Button in code you would type:
- Button myButton = new Button(this);
- Then you can simply press Ctrl-Shift-o to import the Button class.
- import android.widget.Button;
- Refer : **https://developer.android.com/guide/topics/ui/index.html**



Common Controls

Here's a list of some common controls that you can use in your app. Follow the links to learn more about using each one.

> **Note:** Android provides several more controls than are listed here. Browse the `android.widget` package to discover more. If your app requires a specific kind of input control, you can build your own custom components.

Control Type	Description	Related Classes
Button	A push-button that can be pressed, or clicked, by the user to perform an action.	`Button`
Text field	An editable text field. You can use the `AutoCompleteTextView` widget to create a text entry widget that provides auto-complete suggestions	`EditText`, `AutoCompleteTextView`
Checkbox	An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.	`CheckBox`
Radio button	Similar to checkboxes, except that only one option can be selected in the group.	`RadioGroup` `RadioButton`
Toggle button	An on/off button with a light indicator.	`ToggleButton`
Spinner	A drop-down list that allows users to select one value from a set.	`Spinner`
Pickers	A dialog for users to select a single value for a set by using up/down buttons or via a swipe gesture. Use a `DatePicker`code> widget to enter the values for the date (month, day, year) or a `TimePicker` widget to enter the values for a time (hour, minute, AM/PM), which will be formatted automatically for the user's locale.	`DatePicker`, `TimePicker`

- A button consists of text or an icon that communicates what action occurs when the user touches it.

Responding to Click Events

- When the user clicks a button, the Button object receives an on-click event.
- To set a click event handler for a button you can use the button's setOnClickListener(nameOfListener) function like this:
- **// Set the button OnClickListener**
- **myButton.setOnClickListener(myListener);**
- then you can create a class member OnCLickListener object like this:
- **// Create an onClickListener Object**
- **private View.OnClickListener myListener = new View.OnClickListener()**

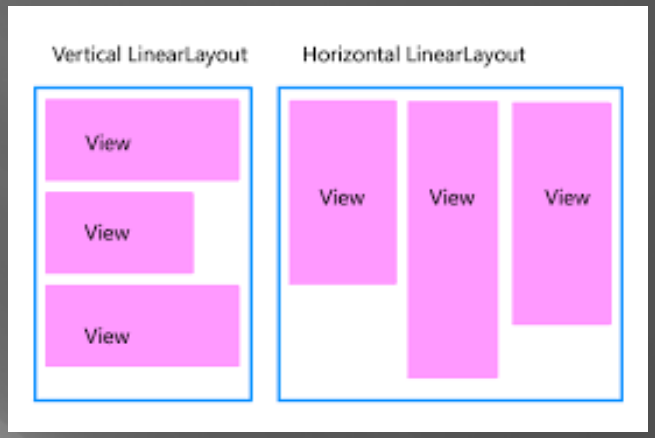
```
{
    public void onClick(View v)
    {
        // Do something here
    }
};
```

Using Built-in Layout Classes

- The types of layouts built into the Android SDK framework include:
  - `LinearLayout`
  - `RelativeLayout`
  - `TableLayout`
  - `FrameLayout`
  - `GridLayout`
- In our course we are going to discuss the first three layouts.
- These layouts are derived from:
  - `android.view.ViewGroup`

Linear Layout

- A `LinearLayout` view organizes its child `View` controls in a single row, or a single column, depending on whether its orientation attribute is set to horizontal or vertical.
- This is a very handy layout method for creating forms.

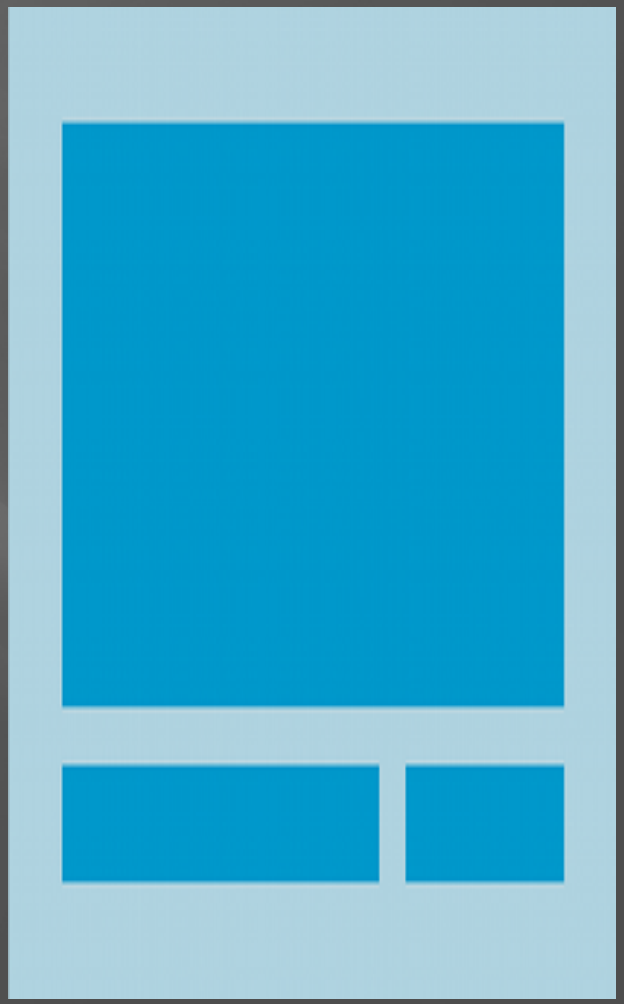


Linear Layout Example

```
<LinearLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent“
    android:orientation=“vertical”>
    <TextView
        android:id="@+id/TextView01"
        android:layout_height="match_parent"
        android:layout_width="match_parent" />
    <Button
    android:id="@+id/Button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content“
    android:text="Press Me"
    android:layout_marginRight="20dp"
    android:layout_marginTop="60dp" />
</LinearLayout>
```

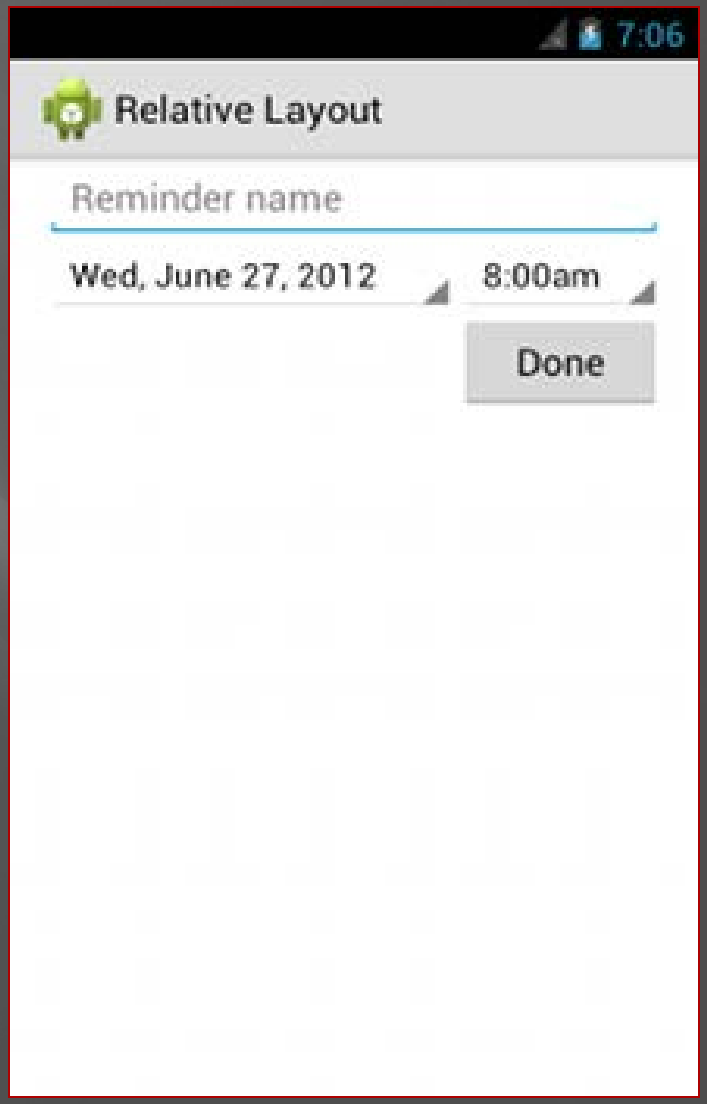
Relative Layout

- The `RelativeLayout` view enables you to specify where the child `View` controls are in relation to each other.
  - For instance, you can set a child `View` to be positioned "above" or "below" or "to the left of" or "to the right of" another `View`.
  - You can also align child `View` controls relative to one another or the parent layout edges.
- Combining `RelativeLayout` attributes can simplify the creation of interesting user interfaces without resorting to multiple layout groups to achieve a desired effect.



RelativeLayout

```
<?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="16dp"
   android:paddingRight="16dp" >
   <EditText
      android:id="@+id/name"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="@string/reminder" />
   <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="@string/done" />
</RelativeLayout>
```



Other Layout

- Look them up on Android Developer site
- They include: TableLayout (think a table), GridLayout, FrameLayout, and MORE!!

TableLayout

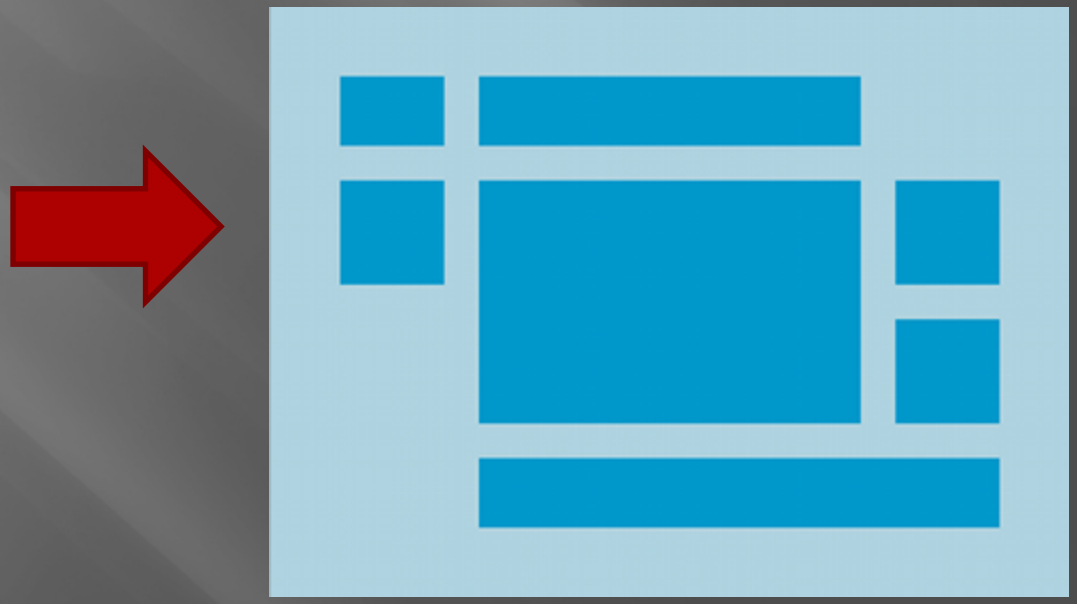
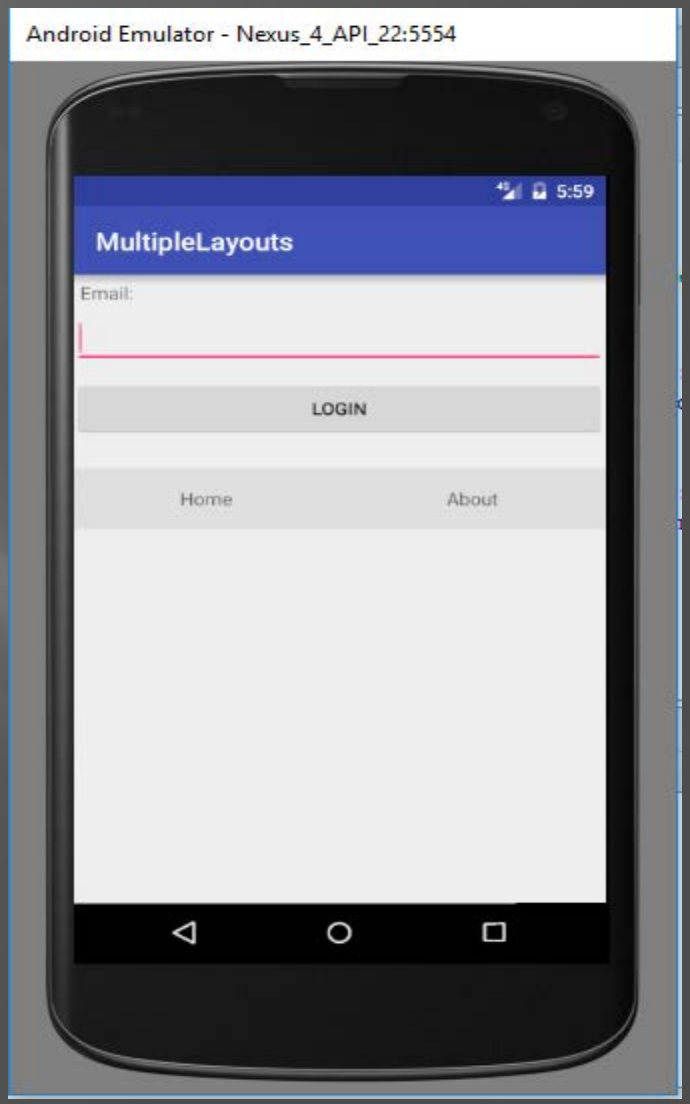


Table Layout Example

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">
    <TableRow android:padding="5dip">
        <TextView
            android:layout_height="wrap_content"
            android:text="New Product Form"
            android:typeface="serif"
            android:layout_span="2"
            android:gravity="center_horizontal"
            android:textSize="20dip" />
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:text="Product Code:"
            android:layout_column="0"/>
        <EditText
            android:id="@+id/prod_code"
            android:layout_height="wrap_content"
            android:layout_column="1"/>
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:text="Product Name:"
            android:layout_column="0"/>
        <EditText
            android:id="@+id/prod_name"
            android:layout_height="wrap_content"/>

    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:text="Product Price:" />
        <EditText
            android:id="@+id/prod_price"
            android:layout_height="wrap_content" />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/add_button"
            android:text="Add Product"
            android:layout_height="wrap_content" />
        <Button
            android:id="@+id/cancel_button"
            android:text="Cancel"
            android:layout_height="wrap_content" />
    </TableRow>
</TableLayout>
```

Nested Layouts

- Combining different layout methods on a single screen can create complex layouts.
- Remember that because a layout contains `View` controls and is, itself, a `View` control, it can contain other layouts.
- Refer Demo Code\Lesson3\MultipleLayouts



```
<?xml version="1.0" encoding="utf-8"?>
<!-- Parent linear layout with vertical orientation -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView android:layout_width="match_parent" android:layout_height="wrap_content"
        android:text="Email:" android:padding="5dip"/>

    <EditText android:layout_width="match_parent" android:layout_height="wrap_content"
        android:layout_marginBottom="10dip"/>

    <Button android:layout_width="match_parent" android:layout_height="wrap_content"
        android:text="Login"/>

    <!-- Child linear layout with horizontal orientation -->
    <LinearLayout android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" android:background="#1111"
        android:layout_marginTop="25dip">

        <TextView android:layout_width="match_parent" android:layout_height="wrap_content"
            android:text="Home" android:padding="15dip" android:layout_weight="1"
            android:gravity="center"/>

        <TextView android:layout_width="match_parent" android:layout_height="wrap_content"
            android:text="About" android:padding="15dip" android:layout_weight="1"
            android:gravity="center"/>

    </LinearLayout>

</LinearLayout>
```

Click Event

- Handle the Click event in two ways
  - In XML
  - In Java Code
- XML Way
  - Configure the following attribute to the UI component

android:onClick="method name"

`android:id="@+id/idname"`

  - If you click the button, it will invoke the specified method from java code. You have to specify your action in java code by creating

public void methodname(View v) { //Implementation}

  - If the method is not available it will throw MethodNotFoundException
  - View is the parent class for all UI Group and components, by using this object you can call objects from xml.

Hands on Example 1 – Birthday Wish – xml way

Problem Requirement : Click the Button to give a Birthday Wish by configuring click event in XML. Once you click the button Hello will be replaced as Happy Birthday The Screen shot shows the activity_main.xml and Emulator screen

activity_main.xml

Activity.java | activity_main.xml

RelativeLayout

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res
    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">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="89dp"
        android:gravity="center"
        android:text="Hello"
        android:textColor="#FF345F"
        android:textSize="28sp"
        android:typeface="monospace" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="166dp"
        android:text="Click Here"
        android:textColor="#FF345F"
        android:textSize="28sp"
        android:onClick="click"/>
</RelativeLayout>
```

Android Emulator - Nexus_4_API_22:5554

2:29

BirthdayWishApp

Hello

CLICK HERE

Hands on Example 1 – Birthday Wish

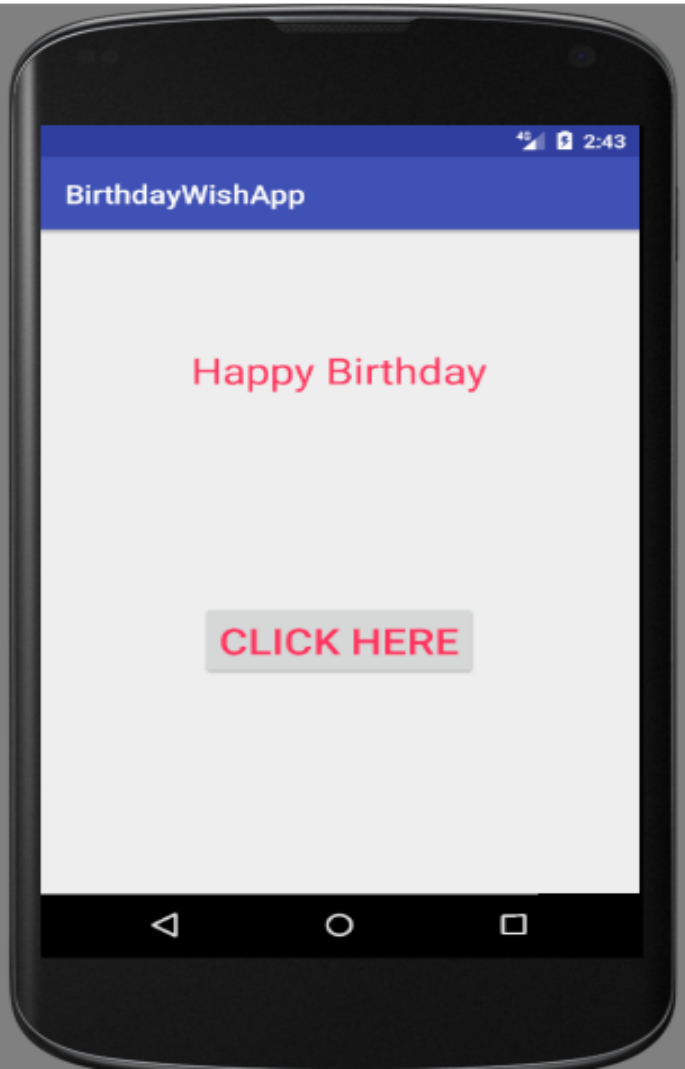
Result Screen: After clicking the CLICK HERE button, you will get the Happy Birthday Message.
The Screen shot shows the MainActivity.java and Emulator output screen

MainActivity | onCreate()

```java
package com.example.rmohanraj.birthdaywishapp;

import ...
public class MainActivity extends AppCompatActivity {
    // Declare UI TextView UI component
    private TextView tv1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Configure the Id
       tv1 = (TextView) findViewById(R.id.textView);
    }
    // Button click event Implementation
    public void click(View view){
      tv1.setText("Happy Birthday");
    }
}
```

Profiler | 4: Run | TODO

Android Emulator - Nexus_4_API_22:5554



Click Event – In Java way

- Java Way using Event Listeners
- An event listener is an interface in the **View** class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.
- onClick()From **View.OnClickListener**. This is called when the user either touches the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses the suitable "enter" key or presses down on the trackball.

```
Component. setOnClickListener(new View.OnClickListener() {
@Override public void onClick(View v) {
// Your implementation
}
});
```

Hands on Example 1 – Birthday Wish using Listener

Remove the line *android:onClick="click"* from activity_main.xml
The Screen shot shows the activity_main.xml and Emulator screen

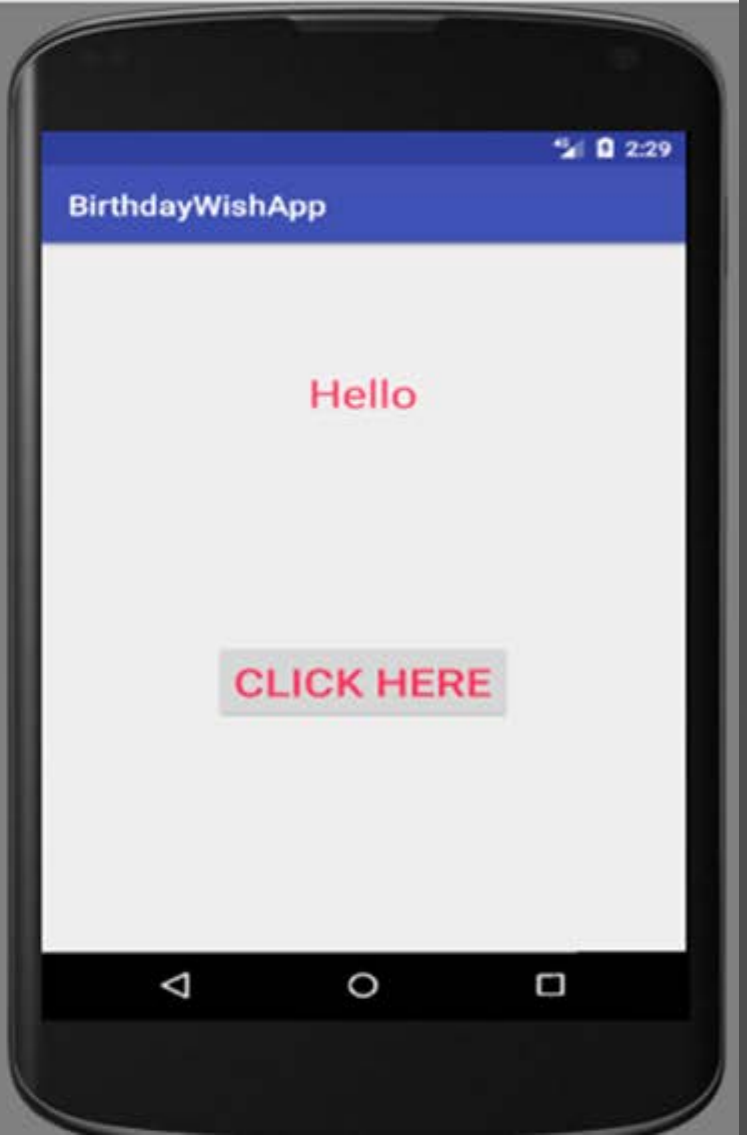
activity_main.xml

Activity.java | activity_main.xml

RelativeLayout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res
    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">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="89dp"
        android:gravity="center"
        android:text="Hello"
        android:textColor="#FF345F"
        android:textSize="28sp"
        android:typeface="monospace" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="166dp"
        android:text="Click Here"
        android:textColor="#FF345F"
        android:textSize="28sp" />

</RelativeLayout>
```

Android Emulator - Nexus_4_API_22:5554



Hands on Example 1 – Birthday Wish using Listener

Result Screen: After clicking the CLICK HERE button, you will get the Happy Birthday Message. **Refer Demo : Lesson3\BirthDayWishApp**

The Screen shot shows the MainActivity.java and Emulator output screen

Run Tools VCS Window Help

app

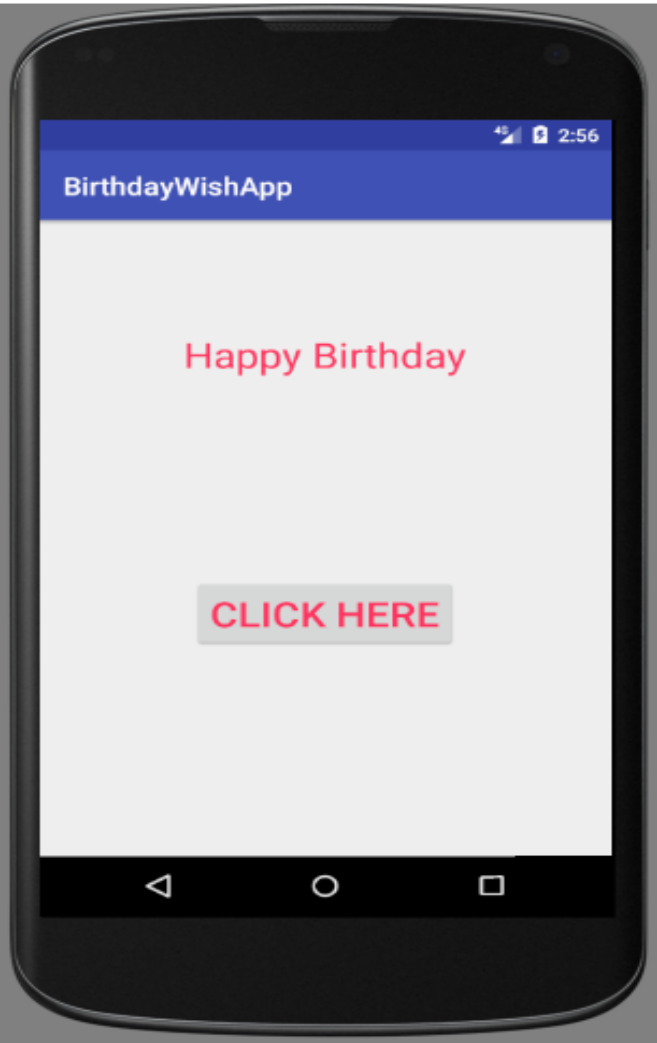
com > example > rmohanraj > birthdaywishapp > MainActivity

MainActivity.java | activity_main.xml

MainActivity | onCreate()

```java
package com.example.rmohanraj.birthdaywishapp;

import ...
public class MainActivity extends AppCompatActivity {
    // Declare UI TextView UI component
    private TextView tv1;
    private Button bt1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Configure the Id
       tv1 = (TextView) findViewById(R.id.textView);
       bt1 =(Button) findViewById(R.id.button);
       // Anonymous Implementation of Button OnClickListener
       bt1.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View view) {
               tv1.setText("Happy Birthday");
           }
       });
    }

}
```

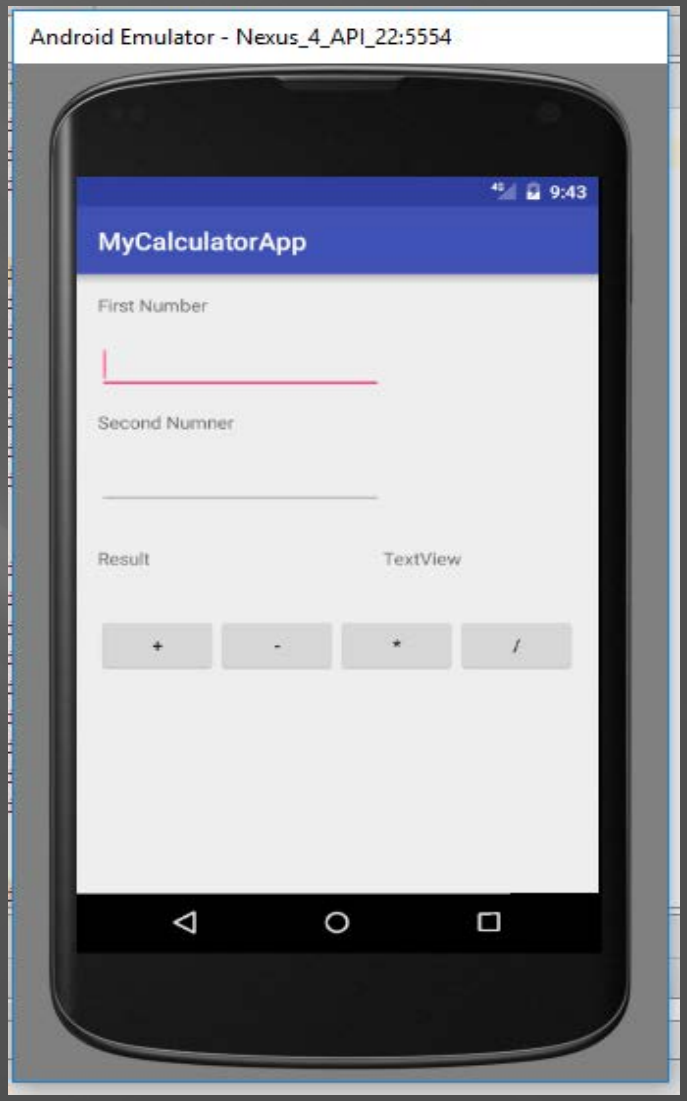
Android Emulator - Nexus_4_API_22:5554



Hands on Example 2 – Simple Calculator

The requirement of this problem is to design a screen as per the screen shot, using nested layouts and performing click action on the operator buttons to display the Result. This screen uses 4 TextView, 2 EditText and 4 Buttons. The toper layout is Relative Layout and all buttons are combined using Linear Layout.

Refer Demo Code: Lesson3\MyCalculatorApp



XML Design(activity_main.xml

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

   <TextView
      android:text="First Number"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/tv1"
      android:layout_alignLeft="@+id/et1"
      android:layout_alignStart="@+id/et1"
      android:layout_alignParentTop="true" />

   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:inputType="number"
      android:ems="10"
      android:layout_marginTop="17dp"
      android:id="@+id/et1"
      android:layout_below="@+id/tv1"
      android:layout_alignLeft="@+id/et2"
      android:layout_alignStart="@+id/et2" />

   <TextView
      android:text="Second Numner"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="15dp"
      android:id="@+id/tv2"
      android:layout_below="@+id/et1"
      android:layout_alignLeft="@+id/et1"
      android:layout_alignStart="@+id/et1" />

   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:inputType="number"
      android:ems="10"
      android:layout_marginTop="16dp"
      android:id="@+id/et2"
      android:layout_below="@+id/tv2"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />

   <TextView
      android:text="Result"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="32dp"
      android:id="@+id/tv3"
      android:layout_below="@+id/et2"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />

   <TextView
      android:text="TextView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/tv4"
      android:layout_alignBaseline="@+id/tv3"
      android:layout_alignBottom="@+id/tv3"
      android:layout_toRightOf="@+id/et2"
      android:layout_toEndOf="@+id/et2" />
```

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="38dp"
    android:layout_below="@+id/tv3"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true">

    <Button
      android:text="+“
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/add“
      android:onClick="click"/>

    <Button
      android:text="-"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/sub"
      android:onClick="click"/>

<Button
  android:text="*"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:id="@+id/mul“
  android:onClick="click"/>

<Button
    android:text="/"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content“
    android:id="@+id/div“
    android:onClick="click"/>

</LinearLayout>

</RelativeLayout>
```

Main Activity.java

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.app.Activity;

public class MainActivity extends Activity implements View.OnClickListener{
    private Button b1,b2,b3,b4;
    private TextView tv;
    private EditText e1,e2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        init();
    }
private void init(){
   b1 = (Button)findViewById(R.id.add);
    b2 = (Button)findViewById(R.id.sub);
    b3 = (Button)findViewById(R.id.mul);
    b4 = (Button)findViewById(R.id.div);
    tv = (TextView)findViewById(R.id.tv4);
    e1 = (EditText) findViewById(R.id.et1);
    e2 = (EditText) findViewById(R.id.et2);
    b1.setOnClickListener(this);
    b2.setOnClickListener(this);
    b3.setOnClickListener(this);
    b4.setOnClickListener(this);
}
```

// Implement using Listener in . java file

```
@Override
    public void onClick(View view) {
        String num1 = e1.getText().toString();
        String num2 = e1.getText().toString();
        switch(view.getId(){
            case R.id.add :
                int addition = Integer.parseInt(num1) + Integer.parseInt(num2);
                tv.setText(String.valueOf(addition));
                break;
            case R.id.sub :
                int minus = Integer.parseInt(num1) - Integer.parseInt(num2);
                tv.setText(String.valueOf(minus));
                break;
            case R.id.mul :
                int mult = Integer.parseInt(num1) * Integer.parseInt(num2);
                tv.setText(String.valueOf(mult));
                break;
            case R.id.div:
                try {
                    int dvd = Integer.parseInt(num1) / Integer.parseInt(num2);
                    tv.setText(String.valueOf(dvd));
                }
                catch(Exception e){
                    tv.setText("Division be Zero");
            }
                break;
        }
    }
}
```