Unit -4
Android Activity [4 Hrs]

The Activity Life Cycle

Every instance of Activity has a lifecycle. During this lifecycle, an activity transitions between three possible states: running, paused, and stopped. For each transition, there is an Activity method that notifies the activity of the change in its state.

As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle. The Activity class provides a number of callbacks that allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.

Within the lifecycle callback methods, you can declare how your activity behaves when the user leaves and re-enters the activity. For example, if you're building a streaming video player, you might pause the video and terminate the network connection when the user switches to another app. When the user returns, you can reconnect to the network and allow the user to resume the video from the same spot. In other words, each callback allows you to perform specific work that's appropriate to a given change of state. Doing the right work at the right time and handling transitions properly make your app more robust and performant. For example, good implementation of the lifecycle callbacks can help ensure that your app avoids:

- Crashing if the user receives a phone call or switches to another app while using your app.
- Consuming valuable system resources when the user is not actively using it.
- Losing the user's progress if they leave your app and return to it at a later time.
- Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). The system invokes each of these callbacks as an activity enters a new state.

Figure 4-1 shows the activity lifecycle, states, and methods.

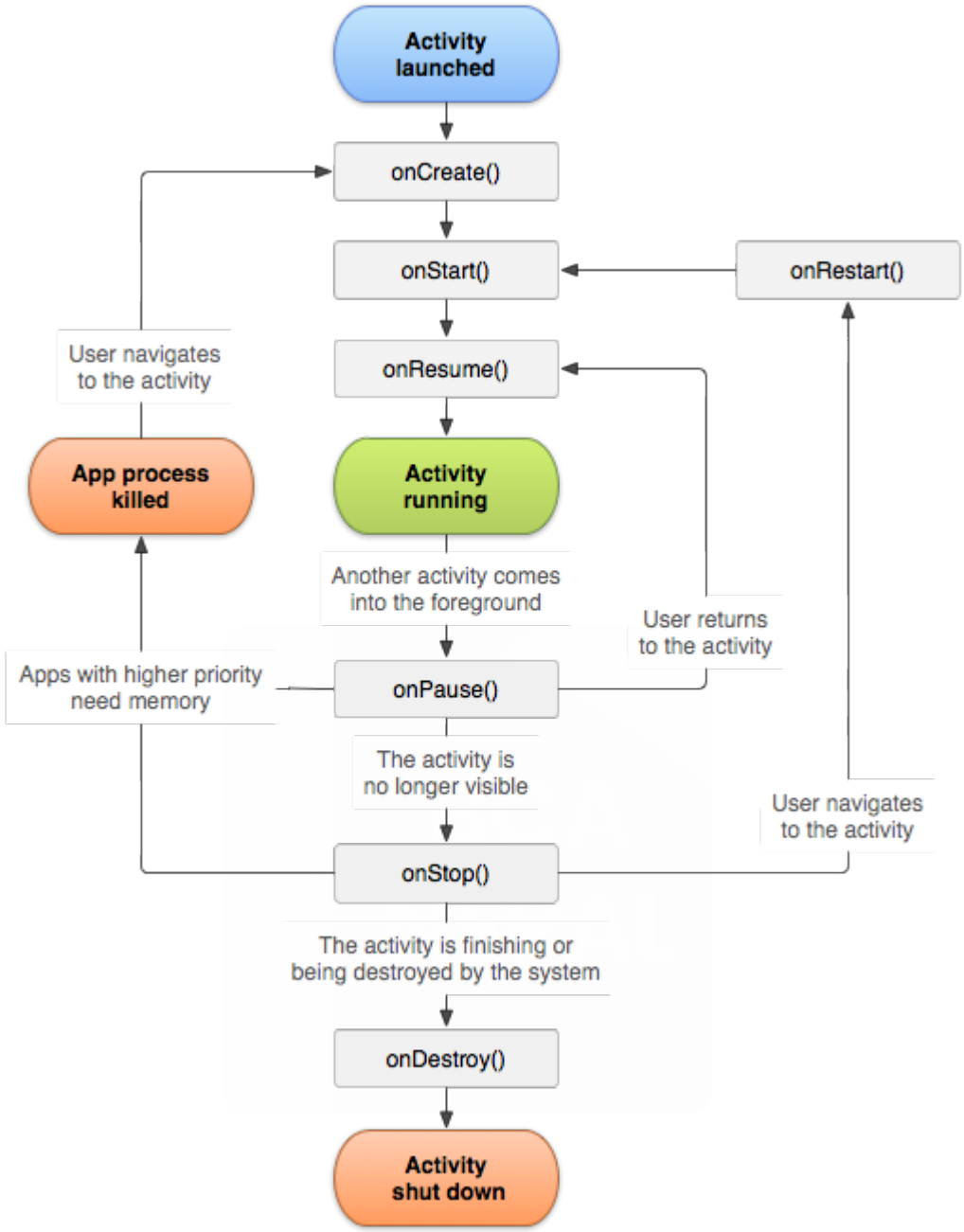


Figure 4-1. A simplified illustration of the activity lifecycle

onCreate()

You must implement this callback, which fires when the system first creates the activity. On activity creation, the activity enters the Created state. In the onCreate() method, you perform basic application startup logic that should happen only once for the entire life of the activity.

```
@Override
protected void onCreate(Bundle b) {
    super.onCreate(b);
    //your stuffs
}
```

onStart()

When the activity enters the Started state, the system invokes this callback. The onStart() call makes the activity visible to the user, as the app prepares for the activity to enter the foreground and become interactive. For example, this method is where the app initializes the code that maintains the UI.

The onStart() method completes very quickly and, as with the Created state, the activity does not stay resident in the Started state. Once this callback finishes, the activity enters the Resumed state, and the system invokes the onResume() method.

```
@Override
protected void onStart()
{
    super.onStart();
    //your stuffs
}
```

onResume()

When the activity enters the Resumed state, it comes to the foreground, and then the system invokes the onResume() callback. This is the state in which the app interacts with the user. The app stays in this state until something happens to take focus away from the app. Such an event might be, for instance, receiving a phone call, the user's navigating to another activity, or the device screen's turning off.

```
@Override
protected void onResume() {
    super.onResume();
    //your stuffs
}
```

onPause()

The system calls this method as the first indication that the user is leaving your activity (though it does not always mean the activity is being destroyed); it indicates that the activity is no longer in the foreground (though it may still be visible if the user is in multi-window mode). Use the onPause() method to pause or adjust operations that should not continue (or should continue in moderation) while the Activity is in the Paused state, and that you expect to resume shortly. There are several reasons why an activity may enter this state. For example:

- Some event interrupts app execution, as described in the onResume() section. This is the most common case.
- In Android 7.0 (API level 24) or higher, multiple apps run in multi-window mode. Because only one of the apps (windows) has focus at any time, the system pauses all of the other apps.
- A new, semi-transparent activity (such as a dialog) opens. As long as the activity is still partially visible but not in focus, it remains paused.

```
@Override
protected void onPause() {
    super.onPause();
    //your stuffs
}
```

onStop()

When your activity is no longer visible to the user, it has entered the Stopped state, and the system invokes the onStop() callback. This may occur, for example, when a newly launched activity covers the entire screen. The system may also call onStop() when the activity has finished running, and is about to be terminated.
You should also use onStop() to perform relatively CPU-intensive shutdown operations. For example, if you can't find a more opportune time to save information to a database, you might do so during onStop().

```
@Override
protected void onStop() {
    super.onStop();
    //your stuffs
}
```

onDestroy()

onDestroy() is called before the activity is destroyed. The system invokes this callback either because:

- the activity is finishing (due to the user completely dismissing the activity or due to finish() being called on the activity), or
- the system is temporarily destroying the activity due to a configuration change (such as device rotation or multi-window mode)

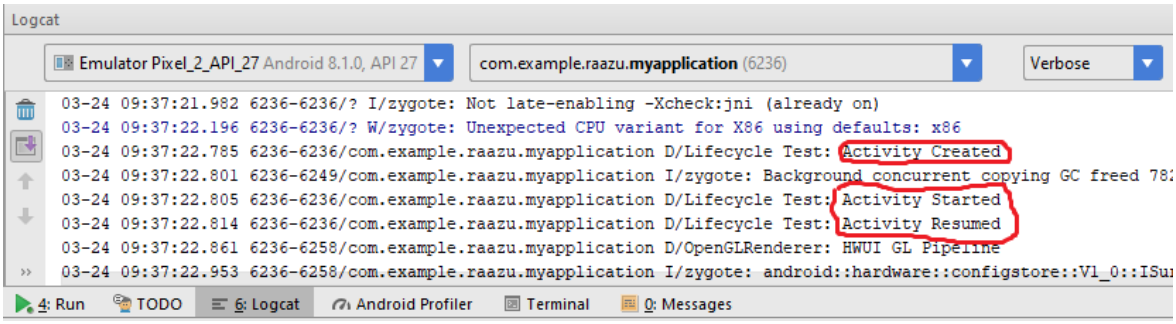
If the activity is finishing, onDestroy() is the final lifecycle callback the activity receives. If onDestroy() is called as the result of a configuration change, the system immediately creates a new activity instance and then calls onCreate() on that new instance in the new configuration. The onDestroy() callback should release all resources that have not yet been released by earlier callbacks such as onStop().

```
@Override
protected void onDestroy() {
    super.onDestroy();
    //your stuffs
}
```

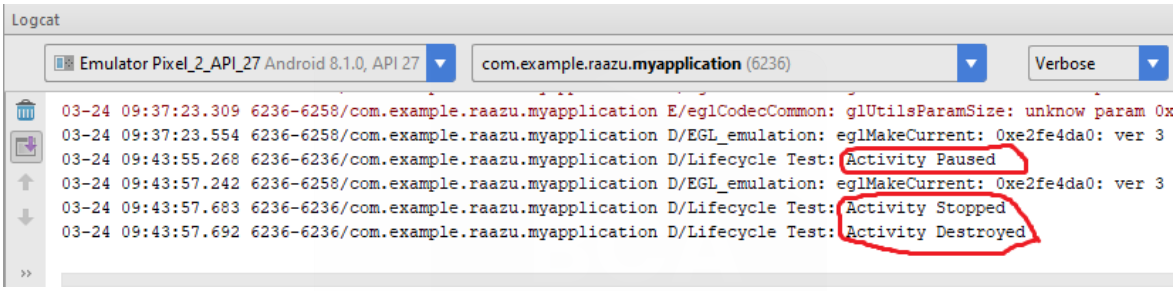
Following example will demonstrate lifecycle of android:

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

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //displaying message in Log
        Log.d("Lifecycle Test", "Activity Created");
    }

    @Override
    protected void onStart()
    {
        super.onStart();
        Log.d("Lifecycle Test", "Activity Started");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.d("Lifecycle Test", "Activity Resumed");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d("Lifecycle Test", "Activity Restarted");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d("Lifecycle Test", "Activity Paused");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d("Lifecycle Test", "Activity Stopped");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("Lifecycle Test", "Activity Destroyed");
    }

}
```

Now go to Logcat present inside Android Monitor: Scroll up and you will notice three methods which were called: Activity Created, Activity started and Activity resumed.



Now press the back button on the Emulator and exit the App. Go to Logcat again and scroll down to bottom. You will see 3 more methods were called: Activity paused, Activity stopped and Activity is being destroyed.



Creating Multiple Activities

So far we have created and tested only a single activity. Now we are going to create multiple activities and also going to learn about linking different activities.
For this purpose, I am going create two activities having the name "**FirstActivity**" and "**SecondActivity**". **FirstActivity** contains **TextView** and **Button**. On the other hand, **SecondActivity** contains a **TextView**.

FirstActivity.java

```
import android.app.Activity;
import android.os.Bundle;

public class FirstActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.first_activity);
    }
}
```

SecondActivity.java

```
import android.app.Activity;
import android.os.Bundle;

public class SecondActivity extends Activity {
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.second_activity);
    }
}
```

first_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is First Activity"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_gravity="center"
        android:layout_marginTop="100dp"
        android:id="@+id/text1"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me"
        android:textSize="20sp"
        android:layout_gravity="center"
        android:id="@+id/button1"
        />

</LinearLayout>
```

second_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Second Activity"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_gravity="center"
        android:layout_marginTop="100dp"
        android:id="@+id/text2"
        />

</LinearLayout>
```

Above xml code produces following design:

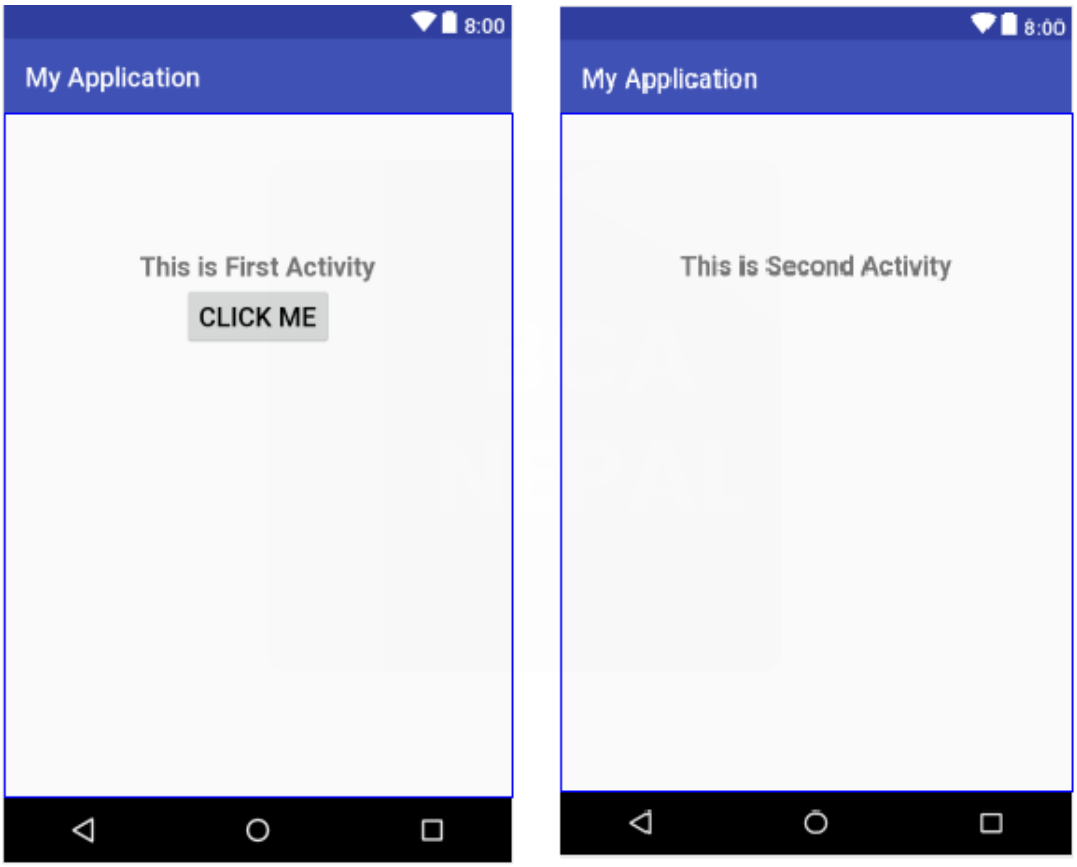


Figure 4-2. Representing FirstActivity and SecondActivity

Declaring Activities in Manifest

Since we have already discussed about declaring activity in manifest file. Previously we got only single activity to be declared in manifest file. But now we are going to discuss about the process of declaring both the activities "**FirstActivity**" and "**SecondActivity**" in manifest file.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.raazu.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".FirstActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
             <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>

        <activity android:name=".SecondActivity"/>

    </application>

</manifest>
```

In above manifest file, **FirstActivity** contains <intent-filter> so, **FirstActivity** will be launched first when we run our application.

Connecting Activities with Intents

An intent is an object that a component can use to communicate with the OS. The only components you have seen so far are activities, but there are also services, broadcast receivers, and content providers.

Intents are multi-purpose communication tools, and the Intent class provides different constructors depending on what you are using the intent to do.

In this case, you are using an intent to tell the **ActivityManager** which activity to start, so you will use this constructor:

public Intent(Context packageContext, Class<?> cls)

The Class object specifies the activity that the **ActivityManager** should start. The **Context** object tells the **ActivityManager** which package the **Class** object can be found in.

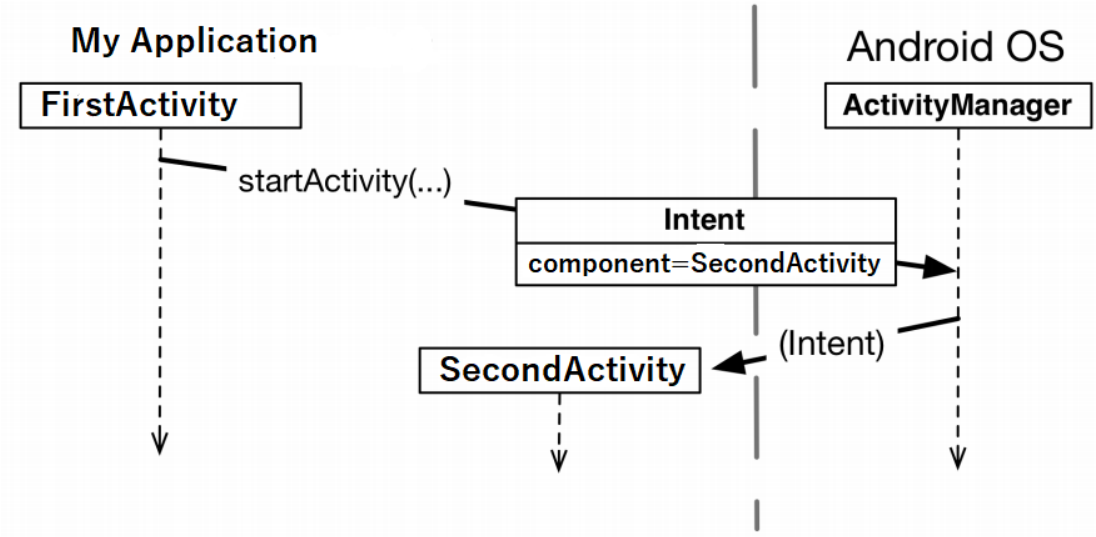


*Figure 4-3. The intent: telling **ActivityManager** what to do*

Following code snippet opens SecondActivity using Intent when user clicks Button in FirstActivity:

```
Button btn=findViewById(R.id.button1);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i=new Intent(FirstActivity.this,SecondActivity.class);
        startActivity(i);
    }
});
```

Passing Data Between Activities

Between activities data can be passed and received by using **Intent extras**. **Extras are arbitrary data that the calling activity can include with an intent**. The OS forwards the intent to the recipient activity, which can then access the extra and retrieve the data.

An extra is structured as a key-value pair. To add an extra to an intent, you use **Intent.putExtra(…)**. In particular, you will be calling

public Intent putExtra(String name, boolean value)

Intent.putExtra(…) comes in many flavors, but it always has two arguments. The first argument is always a String key, and the second argument is the value, whose type will vary.

Let's learn this with the help of an example. Suppose if we want to pass data like id, name and address of student from FirstActivity to SecondActivity then we can use Intent extras as follows:

FirstActivity.java

```
public class FirstActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.first_activity);

        Button btn=findViewById(R.id.button1);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i=new Intent(FirstActivity.this,
                            SecondActivity.class);
                //passing data using putExtra
                i.putExtra("id",001);
                i.putExtra("name","Ram");
                i.putExtra("address","KTM");
                startActivity(i);
            }
        });
    }
}
```

SecondActivity.java

```
public class SecondActivity extends Activity {
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.second_activity);

        //receiving data
        Intent i=getIntent();
        int id=i.getIntExtra("id",0);
        //second argument is default value
        String name=i.getStringExtra("name");
        String address=i.getStringExtra("address");

        //Displaying received data in TextView
        TextView txt=findViewById(R.id.text2);

        txt.setText("Id="+id+"\n"+"Name="+name+"\n"+"Address="+
                    address);
    }
}
```

Following output will be displayed in **SecondActivity**.

Figure 4-4. Output displayed in SecondActivity

Getting Result Back form Child Activity

In previous example, we pass data from FirstActivity to SecondActivity. Now if we want to pass data from SecondActivity back to FirstActivity then we should use following procedure:

- First open SecondActivity form FirstActivity using following Activity method:

  public void startActivityForResult(Intent intent, int requestCode)

  The first parameter is the same intent as before. The second parameter is the request code. The request code is a user-defined integer that is sent to the child activity and then received back by the parent. It is used when an activity starts more than one type of child activity and needs to tell who is reporting back.

- Now pass data from SecondActivity to FirstActivity using following methods. There are two methods you can call in the child activity to send data back to the parent:

  public final void setResult(int resultCode)
  public final void setResult(int resultCode, Intent data)

- And finally, receive data in FirstActivity by overriding **onActivityResult(…)** method.

Following example will demonstrate this procedure. Here we are adding a Button in **SecondActivity**. When user clicks this Button a message "Hello First Activity!" will be sent to FirstActivity. This message will be displayed in **TextView** of FirstActivity.

FirstActivity.java

```
public class FirstActivity extends Activity {
    TextView txt;
    Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.first_activity);

        txt=findViewById(R.id.text1);
        btn=findViewById(R.id.button1);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i=new Intent(FirstActivity.this,
                            SecondActivity.class);
                //starting activity with result code 2
                startActivityForResult(i,2);
            }
        });
    }

    // Call Back method  to get the Message form other Activity
    @Override
    protected void onActivityResult(int requestCode, int
              resultCode, Intent data)
    {
        super.onActivityResult(requestCode, resultCode, data);
        // check if the request code is same as what is passed
                    here it is 2
        if(requestCode==2)
        {
            String message=data.getStringExtra("message");
            txt.setText(message);
        }
    }
}
```

SecondActivity.java

```
public class SecondActivity extends Activity {
    Button btn;
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.second_activity);

        btn=findViewById(R.id.button2);
```

```
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i=new Intent();
                i.putExtra("message","Hello First Activity!");
                setResult(2,i);
                finish();//finishing activity
            }
        });

    }
}
```

first_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is First Activity"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_gravity="center"
        android:layout_marginTop="100dp"
        android:id="@+id/text1"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to Second"
        android:textSize="20sp"
        android:layout_gravity="center"
        android:id="@+id/button1"
        />

</LinearLayout>
```

second_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is Second Activity"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_gravity="center"
        android:layout_marginTop="100dp"
        android:id="@+id/text2"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to First"
        android:textSize="20sp"
        android:layout_gravity="center"
        android:id="@+id/button2"
        />

</LinearLayout>
```

After clicking Button in SecondActivity following output will be produced in FirstActivity:

Figure 4-5. Output displayed in FirstActivity after Button click in SecondActivity

Getting and Setting Data to/from Layout File

To demonstrate this topic, now we are going to create an activity containing some important widgets like TextView, Button, EditText, RadioButton and Spinner. First we try to get all data that we have inputted using these widgets. After getting all data we will display all received data in a TextView.

first_activity.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">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Student Form"
        android:textSize="20sp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:id="@+id/txtForm" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/txtForm"
        android:layout_margin="5dp"
        android:hint="Enter Student Id"
        android:id="@+id/edtId" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtId"
        android:layout_margin="5dp"
        android:hint="Enter Student Name"
        android:id="@+id/edtName" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gender:"
        android:textSize="18sp"
        android:id="@+id/txtGender"
        android:layout_below="@+id/edtName"
        android:layout_marginLeft="10dp"
        android:layout_margin="5dp" />

    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtName"
        android:id="@+id/radGroup"
```

```
        android:orientation="horizontal"
        android:layout_toRightOf="@+id/txtGender">

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male"
            android:id="@+id/radMale" />

        <RadioButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female"
            android:id="@+id/radFemale" />
    </RadioGroup>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Program:"
        android:textSize="18sp"
        android:layout_below="@+id/radGroup"
        android:layout_margin="5dp"
        android:id="@+id/txtProgram" />

    <Spinner
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:entries="@array/programs"
        android:layout_below="@+id/radGroup"
        android:layout_toRightOf="@+id/txtProgram"
        android:layout_marginTop="5dp"
        android:id="@+id/spProgram" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/txtProgram"
        android:layout_marginTop="10dp"
        android:text="Submit"
        android:id="@+id/btnSubmit"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Result"
        android:textSize="18sp"
        android:id="@+id/txtResult"
        android:layout_below="@+id/btnSubmit"
        android:layout_margin="5dp"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

FirstActivity.java

```
public class FirstActivity extends Activity {
    EditText edtId,edtName;
    RadioButton radMale,radFemale;
    Spinner spProgram;
    Button btnSubmit;
    TextView txtResult;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.first_activity);

        edtId=findViewById(R.id.edtId);
        edtName=findViewById(R.id.edtName);
        radMale=findViewById(R.id.radMale);
        radFemale=findViewById(R.id.radFemale);
        spProgram=findViewById(R.id.spProgram);
        btnSubmit=findViewById(R.id.btnSubmit);
        txtResult=findViewById(R.id.txtResult);

        btnSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //getting data from edit text
                String id=edtId.getText().toString();
                String name=edtName.getText().toString();
                //getting data from radio button
                String gender="";
                if(radMale.isChecked())
                    gender="Male";
                else
                    gender="Female";
                //getting data from spinner
           String program=spProgram.getSelectedItem().toString();

                //displaying data in text view
          txtResult.setText("Student Id="+id+"\n"+"Student Name="
              +name+"\n"+"Gender="+gender+"\n"+"Program="+program);
            }
        });

    }
}
```

Above code will produce following output:

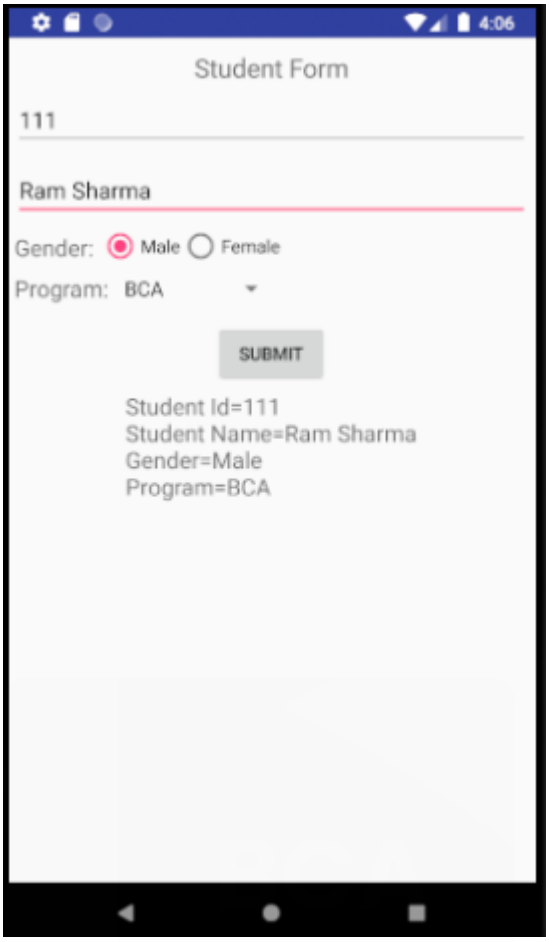


Figure 4-6. Output demonstrating getting and setting data to/from xml file

Now we are going to look another example where user input two numbers using EditText. When user clicks a Button then our application performs addition of two numbers and displays result in a TextView.

first_activity.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">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:hint="Enter First Number"
        android:inputType="number"
        android:id="@+id/edtFirst" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtFirst"
```

```
        android:layout_margin="5dp"
        android:hint="Enter Second Number"
        android:inputType="number"
        android:id="@+id/edtSecond" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtSecond"
        android:layout_marginTop="10dp"
        android:text="Calculate"
        android:id="@+id/btnCalculate"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Result"
        android:textSize="18sp"
        android:id="@+id/txtResult"
        android:layout_below="@+id/btnCalculate"
        android:layout_margin="5dp"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

FirstActivity.java

```
public class FirstActivity extends Activity {
    EditText edtFirst,edtSecond;
    Button btnCalculate;
    TextView txtResult;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.first_activity);

        edtFirst=findViewById(R.id.edtFirst);
        edtSecond=findViewById(R.id.edtSecond);
        btnCalculate=findViewById(R.id.btnCalculate);
        txtResult=findViewById(R.id.txtResult);

        btnCalculate.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View view) {
                //getting data from edit text
                int first,second,result;
                first=Integer.parseInt(edtFirst.getText().toString());
                second=Integer.parseInt(edtSecond.getText().toString());
                result=first+second;
                //displaying result in TextView
                txtResult.setText("Result="+result);
            }
        });
    }
}
```

Above code will produce following output:

Figure 4-7. Output demonstrating addition of two numbers

Exercise

1. What do you mean by android activity? Explain android activity life cycle in detail.
2. How can you declare multiple activities in manifest? Explain.
3. What do you mean by intent? Explain.
4. How can you pass data between multiple activities using intent? Explain.
5. Develop an android application which get result back from a child activity.
6. Develop an android application to calculate simple interest. Your application should contain fields to input principal, rate, time and button for event handling. Calculate and display result in a TextView.
7. Develop an android application to calculate area and perimeter of rectangle. Your application should contain fields to input length and breadth and two buttons for calculating area and perimeter. Calculate and display result in a TextView.
8. Design a signup form using any layout of your choice.
    - Your design must include important widgets like TextView, EditText, Button, RadioButton, CheckBox, Spinner etc.
    - When user clicks a Button display inputted data in a TextView.
9. Develop a simple calculator application with two input fields for inputting numbers and four Buttons for performing addition, subtraction, multiplication and division. Display the result in a TextView.
10. Develop an android application to input your name, address, gender and other personal information. Pass and display this information in another activity.