Lab 4
Intent and Fragment

1. Intent

Android Intent is the *message* that is passed between components such as activities, content providers, broadcast receivers, services etc.

It is generally used with startActivity() method to invoke activity, broadcast receivers etc

Android intents are mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

Types of Android Intents

There are two types of intents in android: implicit and explicit.

1) Implicit Intent

Implicit Intent doesn't specifiy the component. In such case, intent provides information of available components provided by the system that is to be invoked.

For example, you may write the following code to view the webpage.

```
1. Intent intent=new Intent(Intent.ACTION_VIEW);
2. intent.setData(Uri.parse("http://www.google.com"));
3. startActivity(intent);
```

2) Explicit Intent

Explicit Intent specifies the component. In such case, intent provides the external class to be invoked.

```
1. Intent i = new Intent(getApplicationContext(), ActivityTwo.class);
2. startActivity(i);
```

Handling Click events in Button

There are 2 ways to handle the click event in button

- Onclick in xml layout
- Using an OnClickListener

Onclick in XML layout

When the user clicks a button, the Button object receives an on-click event.

To make click event work add `android:onClick` attribute to the Button element in your XML layout. The value for this attribute must be the name of the method you want to call in response to a click event. The Activity hosting the layout must then implement the corresponding method.

NOTE:
If you use this event handler in your code, make sure that you are having that button in your MainActivity. It won't work if you use this event handler in fragment because onClick attribute only works in Activity or MainActivity.

Example:

```
<Button xmlns:android="http:// schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage"
/>
```

In MainActivity class

```
/** Called when the user touches the button */
public void sendMessage(View view)
{
    // Do something in response to button click
}
```

Make sure that your sendMessage method should have the following :

- Be public
- Return void
- Define a View as its only parameter (this will be the View that was clicked)

Using an OnClickListener

You can also declare the click event handler programmatically rather than in an XML layout. This event handler code is mostly preferred because it can be used in both Activities and Fragments.

There are two ways to do this event handler programmatically :

- Implementing View.OnClickListener in your Activity or fragment.
- Creating new anonymous View.OnClickListener.

Implementing View.OnClickListener in your Activity or fragment

To implement `View.OnClickListener` in your Activity or Fragment, you have to override `onClick` method on your class.

Firstly, link the button in xml layout to java by calling `findViewById()` method.
`R.id.button_send` refers the button in XML.

`mButton.setOnClickListener(this);` means that you want to assign listener for your Button “on this instance” this instance represents `OnClickListener` and for this reason your class have to implement that interface.

```
<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"
        tools:context="com.example.sample.MainActivity" >

        <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/button_send" />
</RelativeLayout>
```

MainActivity code:

```
public class MainActivity extends AppCompatActivity
        implements View.OnClickListener {
        private Button mButton;

        @Override
        protected void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);

                mButton = findViewById(R.id.button_send);
                mButton.setOnClickListener(this);
        }
```

```
    @Override
    public void onClick(View view)
    {
        switch (view.getId()) {
        case R.id.button_send:
            // Do something
        }
    }
}
```

If you have more than one button click event, you can use switch case to identify which button is clicked.

Creating Anonymous View.OnClickListener

Link the button from the XML by calling `findViewById()` method and set the onClick listener by using `setOnClickListener()` method.

MainActivity code:

```
public class MainActivity extends AppCompatActivity {


    private Button mButton;


    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        mButton = findViewById(R.id.button_send);
        mButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                // Do something
            }
```

```
        });
    }
}
```

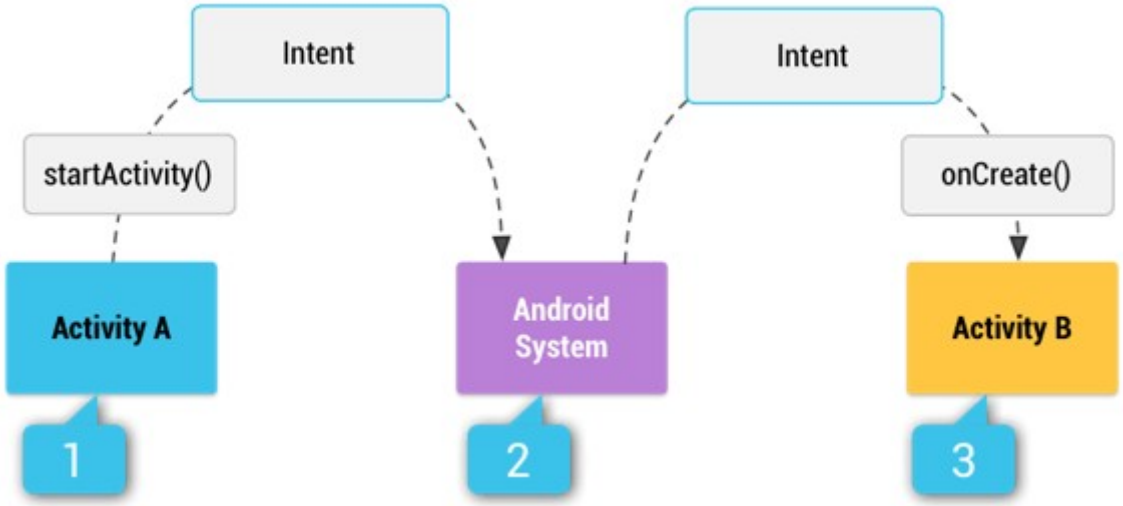
`setOnClickListener` takes an `OnClickListener` object as the parameter. Basically it's creating an anonymous subclass OnClickListener in the parameter.

It's like the same in java when you can create a new thread with an anonymous subclass.

Android Implicit Intents with Examples

In android, **Implicit** Intents won't specify any name of the component to start instead, it declare an action to perform and it allows a component from other apps to handle it. For example, by using implicit intents we can request another app to show the location details of the user or etc.

Following is the pictorial representation of how **Implicit intents** send a request to the android system to start another activity.



If you observe the above image **Activity A** creates an intent with the required action and sends it to an android system using the **startActivity()** method. The android system will search for an intent filter that matches the intent in all apps. Whenever the match found the system starts matching activity (**Activity B**) by invoking the **onCreate()** method.

In android when we create implicit intents, the android system will search for matching components by comparing the contents of intent with intent filters which defined in the **manifest** file of other apps on the device. If the matching component found, the system starts that component and sends it to the Intent object. In case, if multiple intent filters are matched then the system displays a dialog so that the user can pick which app to use.

In android, an **Intent Filter** is an expression in the app's **manifest** file and it is used to specify the type of intents that the component would like to receive. In case if we create an **Intent Filter** for activity, there is a possibility for other apps to start our activity by sending a certain type of intent otherwise the activity can be started only by an explicit intent.

Following is the simple code snippet of **implicit intent** in the android application.

```
Intent intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.google.com"));
startActivity(intent);
```

If you observe above implicit intent we didn't defined any specific name of component to start, instead we defined an action (**ACTION_VIEW**) to open the defined URL (http://www.google.com) in browser within the device.

Example

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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"
   tools:context="com.intents.MainActivity">
   <EditText
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/urlText"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="100dp"
      android:ems="10" />
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/btnNavigate"
      android:layout_below="@+id/urlText"
      android:text="Navigate"
      android:layout_centerHorizontal="true" />
</RelativeLayout>
```

MainActivity.java

```
import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText editText = (EditText)findViewById(R.id.urlText);
        Button btn = (Button) findViewById(R.id.btnNavigate);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String url = editText.getText().toString();
                Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
                startActivity(intent);
            }
        });
    }
}
```

Android Explicit Intents with Examples

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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:id="@+id/fstTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="150dp"
```

```
        android:text="First Number"
        />
    <EditText
        android:id="@+id/firstNum"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10">
    </EditText>
    <TextView
        android:id="@+id/secTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Second Number"
        android:layout_marginLeft="100dp"
        />
    <EditText
        android:id="@+id/secondNum"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10" />
    <Button
        android:id="@+id/addBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:text="Add" />
</LinearLayout>
```

Now we will create another layout resource file **result.xml** in **\src\main\res\layout** path to get the first activity (**activity_main.xml**) details in second activity file for that right click on your layout folder → Go to **New** → select **Layout Resource File** and give name as **result.xml**.

Once we create a new layout resource file **result.xml**, open it and write the code like as shown below

result.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:id="@+id/resultView"
    android:layout_marginLeft="100dp"
    android:layout_marginTop="150dp"/>
</LinearLayout>
```

Now open our main activity file **MainActivity.java** from **\src\main\java\com.intents** path and write the code like as shown below

MainActivity.java

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final EditText firstNum = (EditText)findViewById(R.id.firstNum);
        final EditText secNum = (EditText)findViewById(R.id.secondNum);
        Button btnAdd = (Button)findViewById(R.id.addBtn);
        btnAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int num1 = Integer.parseInt(firstNum.getText().toString());
                int num2 = Integer.parseInt(secNum.getText().toString());
                Intent intent = new Intent(MainActivity.this,ResultActivity.class);
                intent.putExtra("SUM",num1+" + "+num2+" = "+(num1+num2));
                startActivity(intent);
            }
        });
    }
}
```

Now we will create another activity file **ResultActivity.java** in **\src\main\java\com.intents** path to get the first activity (**MainActivity.java**) details in second activity file for that right click on your application folder → Go to **New** → select **Java Class** and give name as **ResultActivity.java**.

Once we create a new activity file **ResultActivity.java**, open it and write the code like as shown below

ResultActivity.java

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;

public class ResultActivity  extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.result);
      TextView result = (TextView)findViewById(R.id.resultView);
      Intent intent = getIntent();
      String addition = (String)intent.getSerializableExtra("SUM");
      result.setText(addition);
   }
}
```

Now we need to add this newly created activity in **ActivityManifest.xml** file in like as shown below.

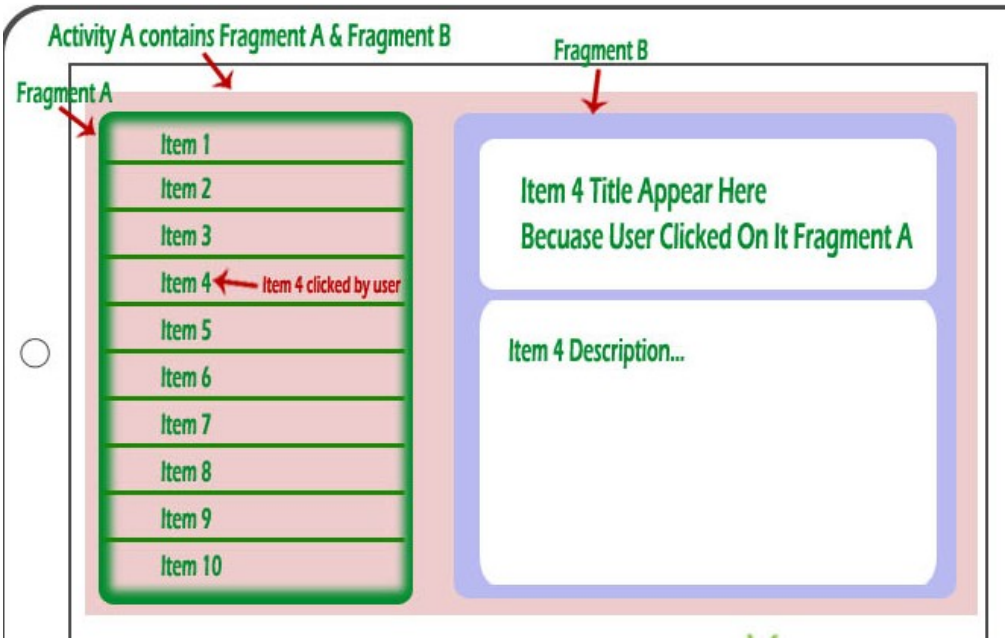
ActivityMainfest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.intents">
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="Explicit Intent - Activity1"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity android:name=".MainActivity">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <activity android:name=".ResultActivity" android:label="Explicit Intent - Activity2">
      </activity>
   </application>
</manifest>
```

If you observe above example we are performing an addition operation in one activity (**MainActivity.java**) and sending those details to another activity (**ResultActivity.java**) and added all the activities in **AndroidManifest.xml** file.

2. Fragment

In Android, Fragment is a part of an activity which enable more modular activity design. It will not be wrong if we say a fragment is a kind of sub-activity. It represents a behaviour or a portion of user interface in an Activity. We can combine multiple Fragments in Single Activity to build a multi panel UI and reuse a Fragment in multiple Activities. We always need to embed Fragment in an activity and the fragment lifecycle is directly affected by the host activity's lifecycle.



We can create Fragments by extending Fragment class or by inserting a Fragment into our Activity layout by declaring the Fragment in the activity's layout file, as a <fragment> element.

Need Of Fragments In Android:
Before the introduction of Fragment's we can only show a single Activity on the screen at one given point of time so we were not able to divide the screen and control different parts separately. With the help of Fragment's we can divide the screens in different parts and controls different parts separately.

By using Fragments we can comprise multiple Fragments in a single Activity. Fragments have their own events, layouts and complete life cycle. It provide flexibility and also removed the limitation of single Activity on the screen at a time.

Basic Fragment Code In XML:

```
<fragment
android:id="@+id/fragments"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

Create A Fragment Class In Android Studio:
For creating a Fragment firstly we extend the Fragment class, then override key lifecycle methods to insert our app logic, similar to the way we would with an Activity class. While creating a Fragment we must use onCreateView() callback to define the layout and in order to run a Fragment.

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.ViewGroup;

public class FirstFragment extends Fragment {
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
// Inflate the layout for this fragment
return inflater.inflate(R.layout.fragment_first, container, false);
}
}
```

Here the inflater parameter is a LayoutInflater used to inflate the layout, container parameter is the parent ViewGroup (from the activity's layout) in which our Fragment layout will be inserted.

The savedInstanceState parameter is a Bundle that provides data about the previous instance of the Fragment. The inflate() method has three arguments first one is the resource layout which we want to inflate, second is the ViewGroup to be the parent of the inflated layout. Passing the container is important in order for the system to apply layout parameters to the root view of the inflated layout, specified by the parent view in which it's going and the third parameter is a boolean value indicating whether the inflated layout should be attached to the ViewGroup (the second parameter) during inflation.

Fragments were added in in Honeycomb version of Android i.e API version 11. There are some primary classes related to Fragment's are:
1. FragmentActivity: The base class for all activities using compatibility based Fragment (and loader) features.

2. Fragment: The base class for all Fragment definitions

3. FragmentManager: The class for interacting with Fragment objects inside an activity

4. FragmentTransaction: The class for performing an atomic set of Fragment operations such as Replace or Add a Fragment.

Fragment Example

Below is the example of Fragment's. In this example we create two Fragments and load them on the click of Button's. We display two Button's and a FrameLayout in our Activity and perform setOnClickListener event on both Button's. On the click of First Button we replace the First Fragment and on click of Second Button we replace the Second Fragment with the layout(FrameLayout). In the both Fragment's we display a TextView and a Button and onclick of Button we display the name of the Fragment with the help of Toast.

Step 1: Create a new project and name it FragmentExample
Step 2: Open res -> layout ->activity_main.xml (or) main.xml and add following code:

```
<LinearLayout 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:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity">
<!-- display two Button's and a FrameLayout to replace the Fragment's  -->
<Button
android:id="@+id/firstFragment"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/button_background_color"
android:text="First Fragment"
android:textColor="@color/white"
android:textSize="20sp" />

<Button
android:id="@+id/secondFragment"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="10dp"
android:background="@color/button_background_color"
android:text="Second Fragment"
android:textColor="@color/white"
android:textSize="20sp" />

<FrameLayout
android:id="@+id/frameLayout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_marginTop="10dp" />
</LinearLayout>
```

Step 3: Open src -> package -> MainActivity.java
In this step we open MainActivity and add the code for initiate the Button's. After that we perform setOnClickListener event on both Button's. On the click of First Button we replace the First Fragment and on click of Second Button we replace the Second Fragment with the layout(FrameLayout). For replacing a Fragment with FrameLayout firstly we create a Fragment Manager and then begin the

transaction using Fragment Transaction and finally replace the Fragment with the layout i.e FrameLayout.

import android.app.Fragment;

```
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

Button firstFragment, secondFragment;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
// get the reference of Button's
firstFragment = (Button) findViewById(R.id.firstFragment);
secondFragment = (Button) findViewById(R.id.secondFragment);

// perform setOnClickListener event on First Button
firstFragment.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// load First Fragment
loadFragment(new FirstFragment());
}
});
// perform setOnClickListener event on Second Button
secondFragment.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// load Second Fragment
loadFragment(new SecondFragment());
}
});

}

private void loadFragment(Fragment fragment) {
// create a FragmentManager
FragmentManager fm = getFragmentManager();
// create a FragmentTransaction to begin the transaction and replace the Fragment
FragmentTransaction fragmentTransaction = fm.beginTransaction();
// replace the FrameLayout with new Fragment
fragmentTransaction.replace(R.id.frameLayout, fragment);
fragmentTransaction.commit(); // save the changes
}
}
```

Step 4: Now we need 2 fragments and 2 xml layouts. So create two fragments by right click on your package folder and create classes and name them as FirstFragment and SecondFragment and add the following code respectively.

FirstFragment.class

In this Fragment firstly we inflate the layout and get the reference of Button. After that we perform setOnClickListener event on Button so whenever a user click on the button a message “First Fragment“ is displayed on the screen by using a Toast.

```
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;

public class FirstFragment extends Fragment {


View view;
Button firstButton;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
// Inflate the layout for this fragment
view = inflater.inflate(R.layout.fragment_first, container, false);
// get the reference of Button
firstButton = (Button) view.findViewById(R.id.firstButton);
// perform setOnClickListener on first Button
firstButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
// display a message by using a Toast
Toast.makeText(getActivity(), "First Fragment", Toast.LENGTH_LONG).show();
}
});
return view;
}
}
```

SecondFragment.class

In this Fragment firstly we inflate the layout and get the reference of Button. After that we perform setOnClickListener event on Button so whenever a user click on the button a message “Second Fragment“ is displayed on the screen by using a Toast.

```
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;


public class SecondFragment extends Fragment {

View view;
Button secondButton;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
// Inflate the layout for this fragment
view = inflater.inflate(R.layout.fragment_second, container, false);
// get the reference of Button
secondButton = (Button) view.findViewById(R.id.secondButton);
// perform setOnClickListener on second Button
secondButton.setOnClickListener(new View.OnClickListener() {
@Override
```

```
public void onClick(View v) {
// display a message by using a Toast
Toast.makeText(getActivity(), "Second Fragment", Toast.LENGTH_LONG).show();
}
});
return view;
}
}
```

Step 5: Now create 2 xml layouts by right clicking on res/layout -> New -> Layout Resource File and name them as fragment_first and fragment_second and add the following code in respective files. Here we will design the basic simple UI by using TextView and Button in both xml's.

fragment_first.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"
tools:context="com.example.fragmentexample.FirstFragment">

<!--TextView and Button displayed in First Fragment -->
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="100dp"
android:text="This is First Fragment"
android:textColor="@color/black"
android:textSize="25sp" />

<Button
android:id="@+id/firstButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:layout_marginLeft="20dp"
android:layout_marginRight="20dp"
android:background="@color/green"
android:text="First Fragment"
android:textColor="@color/white"
android:textSize="20sp"
android:textStyle="bold" />
</RelativeLayout>
```

fragment_second.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"
tools:context="com.example.fragmentexample.SecondFragment">

<!--TextView and Button displayed in Second Fragment -->
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_marginTop="100dp"
android:text="This is Second Fragment"
android:textColor="@color/black"
android:textSize="25sp" />
```

```
<Button
android:id="@+id/secondButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"
android:layout_marginLeft="20dp"
android:layout_marginRight="20dp"
android:background="@color/green"
android:text="Second Fragment"
android:textColor="@color/white"
android:textSize="20sp"
android:textStyle="bold" />

</RelativeLayout>
```

Step 6: Open res ->values ->colors.xml
In this step we define the color's that used in our xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- color's used in our project -->
<color name="black">#000</color>
<color name="green">#0f0</color>
<color name="white">#fff</color>
<color name="button_background_color">#925</color>
</resources>
```

Step 7: Open AndroidManifest.xml

In this step we show the Android Manifest file in which do nothing because we need only one Activitty i.e MainActivity which is already defined in it. In our project we create two Fragment's but we don't need to define the Fragment's in manifest because Fragment is a part of an Activity.

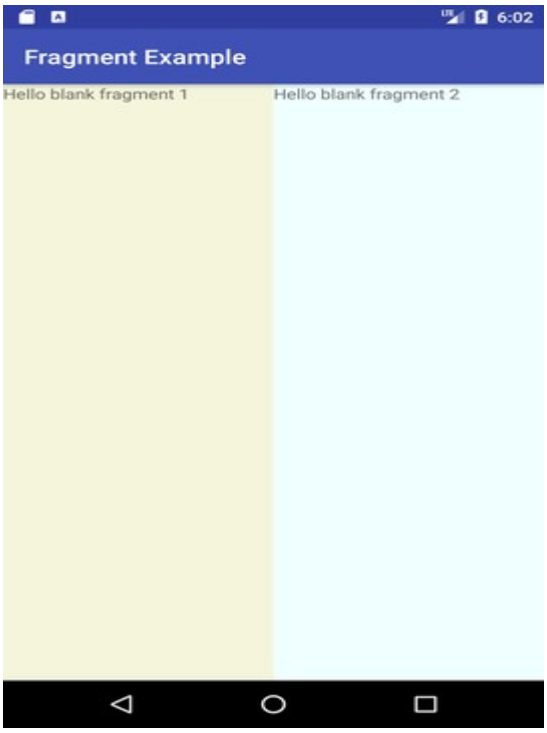
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.fragmentexample" >

<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".MainActivity"
android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

Step 8: Now run the App and you will two button. Clicking on first button shows First Fragment and on click of Second Button shows the Second Fragment which is actually replacing layout(FrameLayout).

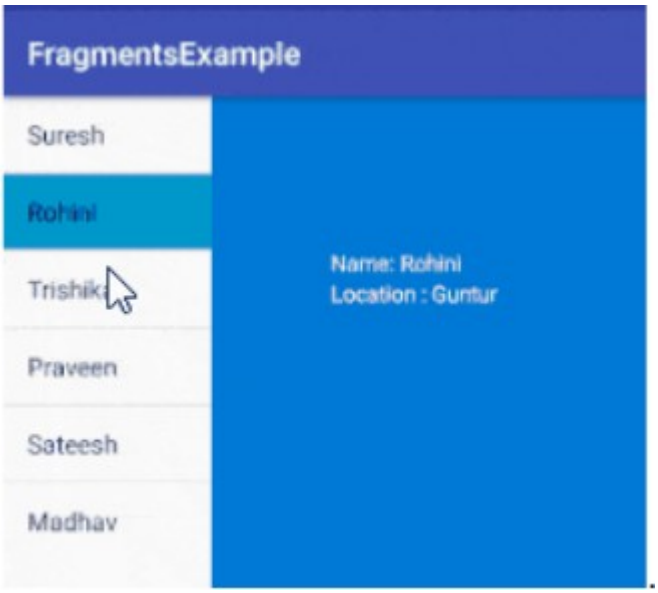

FragmentExample
FIRST FRAGMENT
SECOND FRAGMENT

EXERSICE

1. create two fragments as given below.



2. Create an application which creates a two fragments one with list item, and another shows the details of item selected in first fragment.



3. Create an android application with four buttons like web browser, make calls, show map and choose contact

Operations are

1. By clicking on browser button , it should open “www.google.com” in android built in browser application
2. By Clicking on make call button, it should open the phone dialer with a predefined number.
3. By Clicking on show map button, it should plot a specified location coordinates in the built in google map application.

4. Create an android application with two activities. First activity with a “ENTER” button and second screen with a welcome message. By clicking “ENTER” it should bring the use to the second screen.