BICT- Level III – Semester II
Topic – Mobile Application Development (ICT3233)
Lab Sheet 06

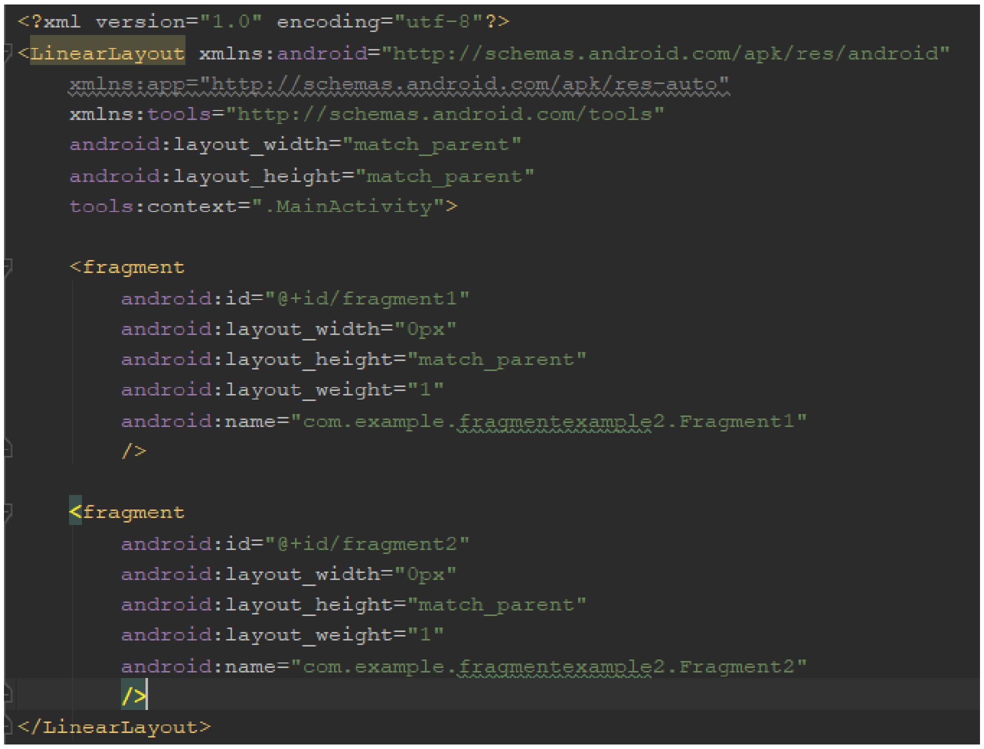
Fragments

- A Fragment represents a reusable portion of your app's UI.
- Fragments are always embedded in Activities i.e., they are added to the layout of activity in which they reside. Multiple fragments can be added to one activity. This task can be carried out in 2 ways:

    1. Statically: Explicitly mention the fragment in the XML file of the activity. This type of fragment can't be replaced during the run time.
    2. Dynamically: **FragmentManager** is used to embed fragments with activities that enable the addition, deletion, or replacement of fragments at run time.

Example 01 – Adding fragments statistically

- Create a new project.
- Open *activity_main.xml* and add the following code.

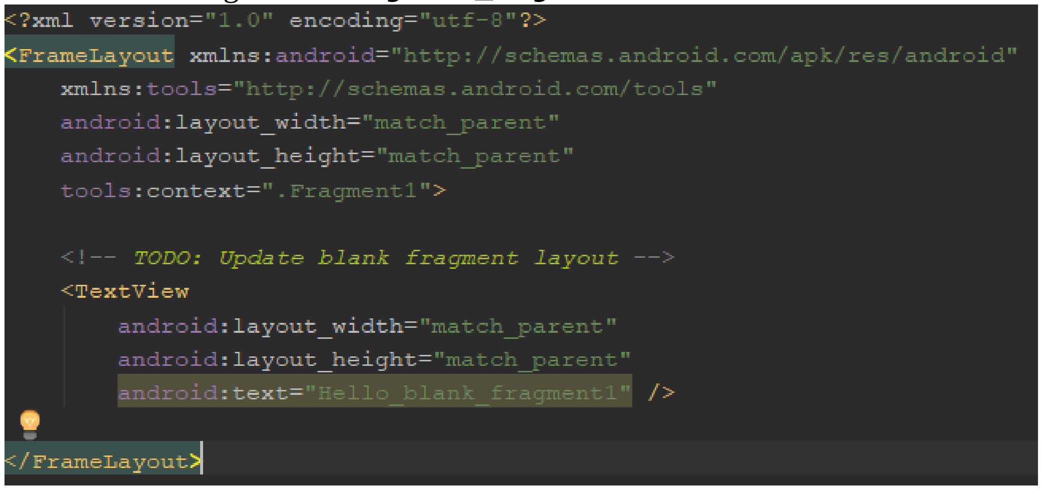


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/fragment1"
        android:layout_width="0px"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:name="com.example.fragmentexample2.Fragment1"
        />

    <fragment
        android:id="@+id/fragment2"
        android:layout_width="0px"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:name="com.example.fragmentexample2.Fragment2"
        />
</LinearLayout>
```

- In here we have embedded two fragments to the main activity's layout (added the fragments statistically).
- Then create two fragments (fragment classes and layouts) in your project.

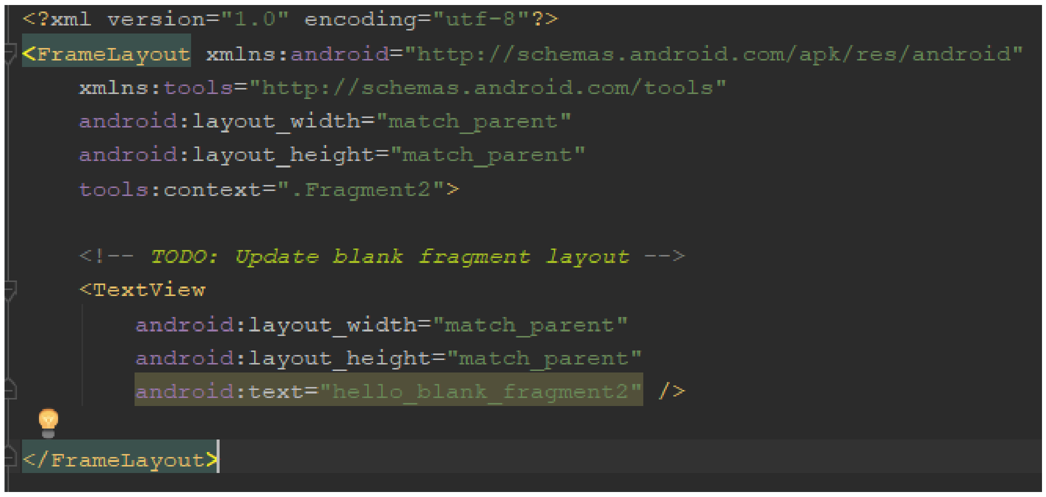
Right click on **layout** sub folder→ New → Fragment → Fragment (Blank)

- Add the following code to *fragment_fragment1.xml*



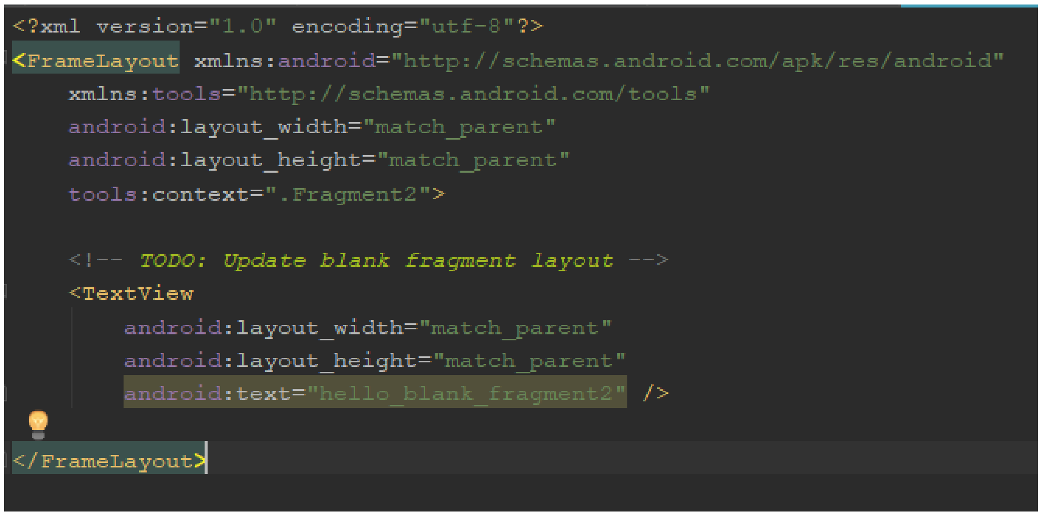
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Fragment1">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Hello_blank_fragment1" />

</FrameLayout>
```

- In here, we have used a FrameLayout and a TextView to display the text **"Hello_blank _fragment1"**.
- Add the following code to *fragment_fragment2.xml*



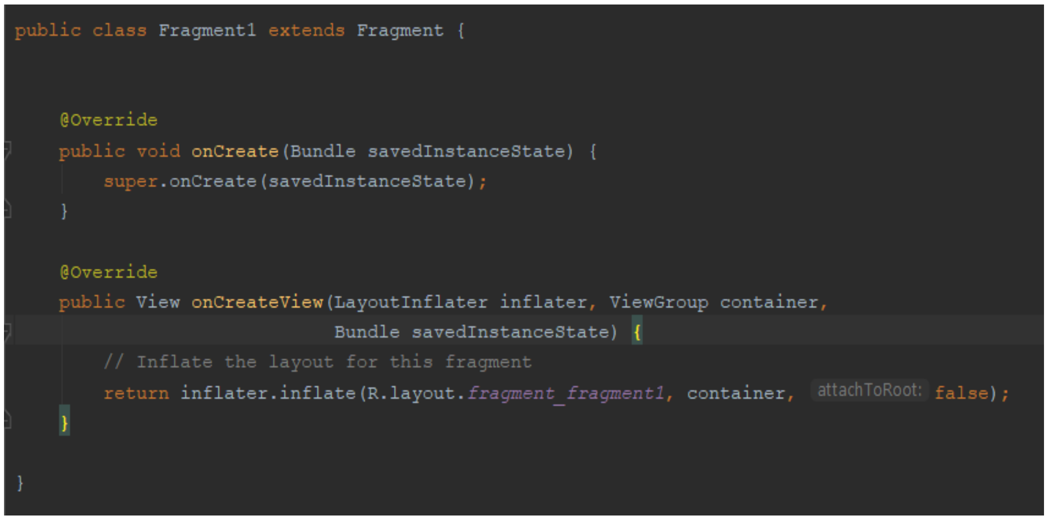
```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Fragment2">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="hello_blank_fragment2" />

</FrameLayout>
```

- In here also we have used a FrameLayout and a TextView to display the text **"Hello_blank _fragment2".**



```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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=".Fragment2">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="hello_blank_fragment2" />

</FrameLayout>
```

- Open *Fragment1.java* and add the following code.

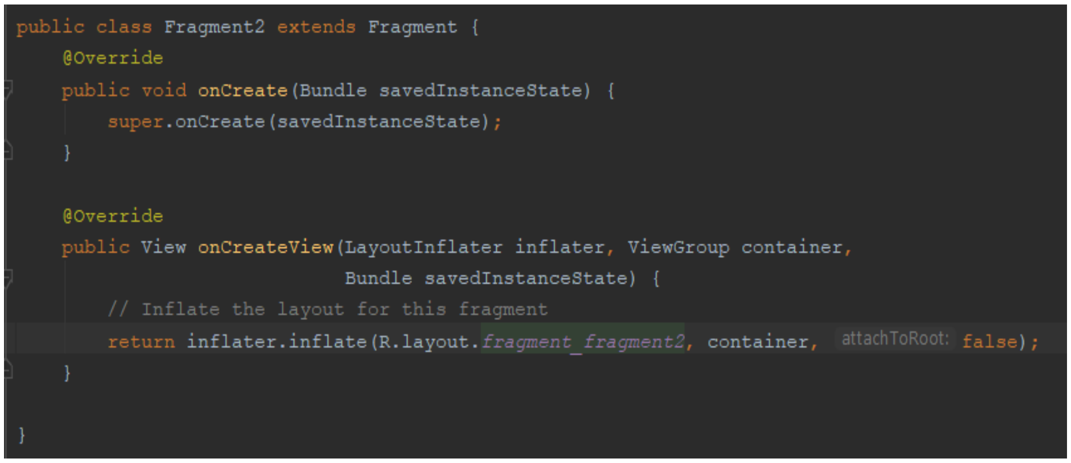


```
public class Fragment1 extends Fragment {


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_fragment1, container, attachToRoot: false);
    }

}
```

- The **onCreateView()** method is called when fragment should create its View object hierarchy, either dynamically or via XML layout inflation.
- You are passing three parameters to **onCreateView()** method.

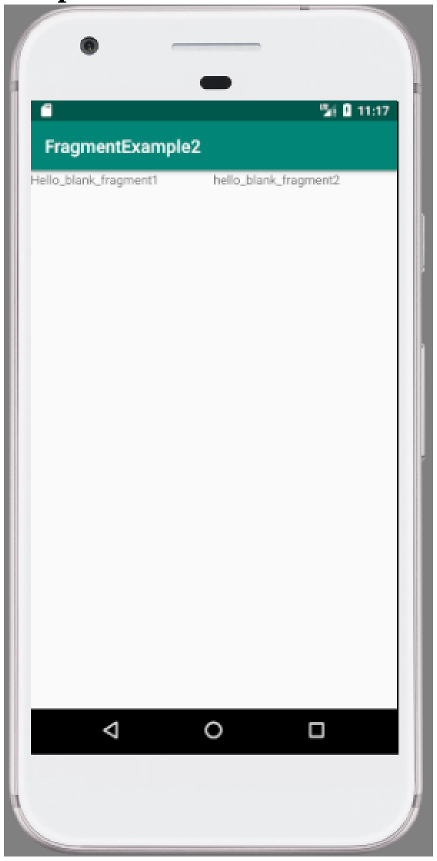
    1. **LayoutInflater inflater** - The LayoutInflater object that can be used to inflate any views in the fragment.
    2. **ViewGroup container** - If non-null, this is the parent view that the fragment's UI should be attached to.
    3. **Bundle savedInstanceState -** If non-null, this fragment is being re-constructed from a previous saved state as given here.

- Open *Fragment2.java* and add the following code.



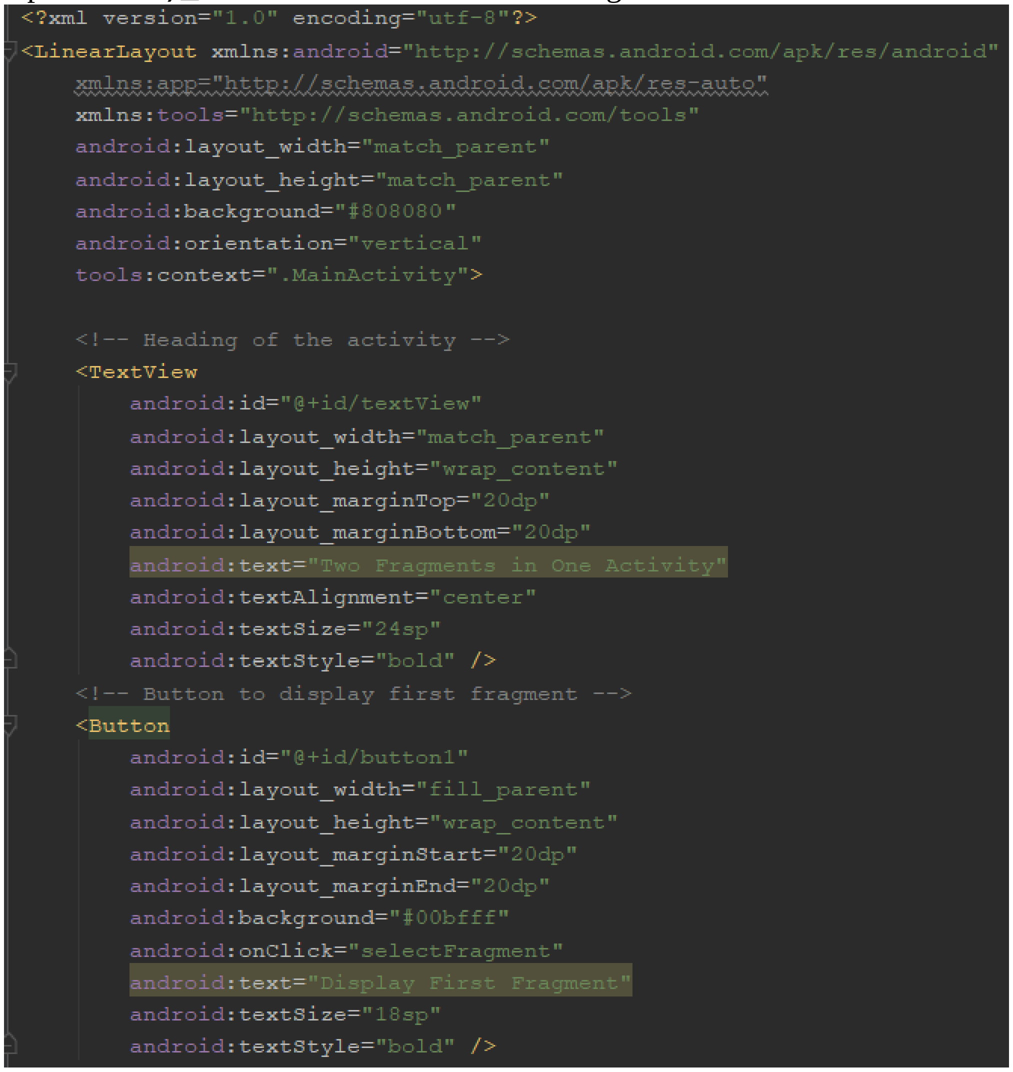
```
public class Fragment2 extends Fragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_fragment2, container,  attachToRoot: false);
    }

}
```

Output:

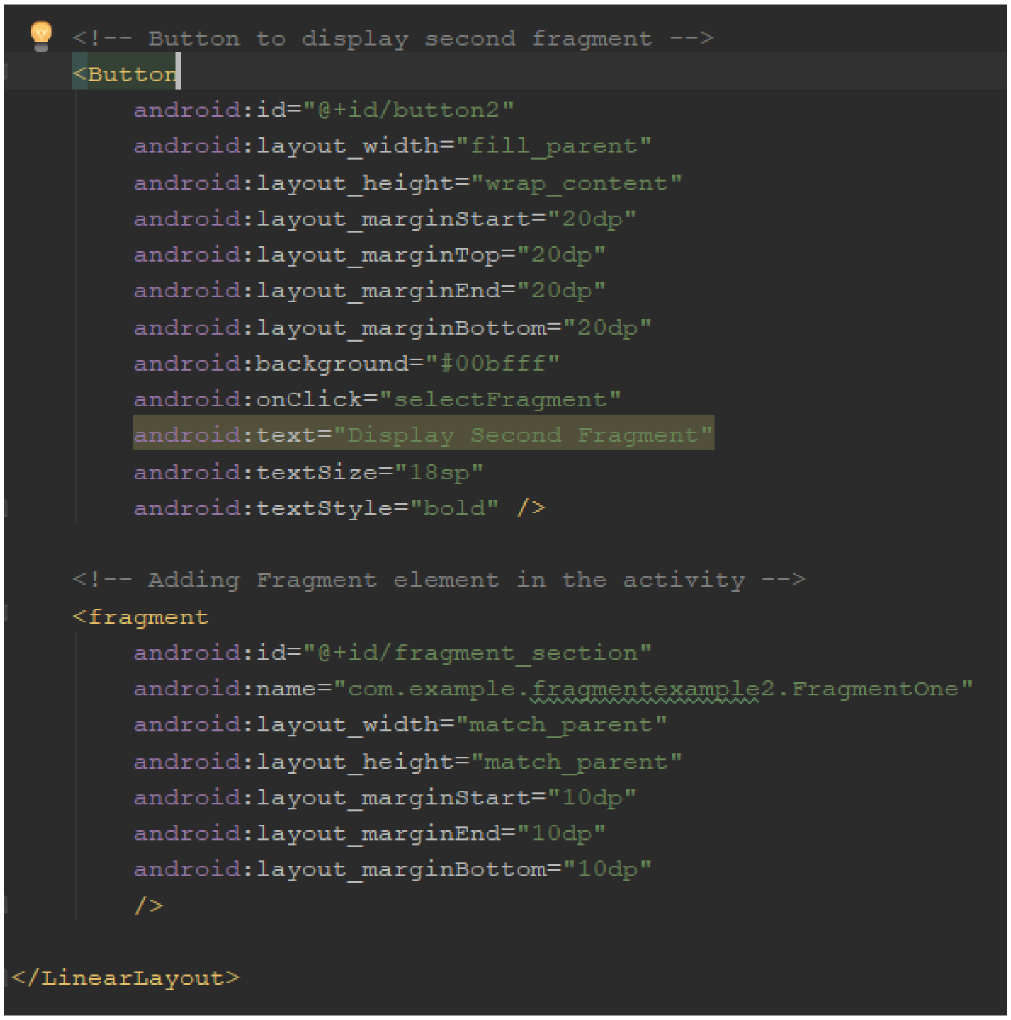


Example 02 – Adding fragments dynamically

- Create a new project.
- Open *activity_main.xml* and add the following code.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#808080"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <!-- Heading of the activity -->
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginBottom="20dp"
        android:text="Two Fragments in One Activity"
        android:textAlignment="center"
        android:textSize="24sp"
        android:textStyle="bold" />
    <!-- Button to display first fragment -->
    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginEnd="20dp"
        android:background="#00bfff"
        android:onClick="selectFragment"
        android:text="Display First Fragment"
        android:textSize="18sp"
        android:textStyle="bold" />
```



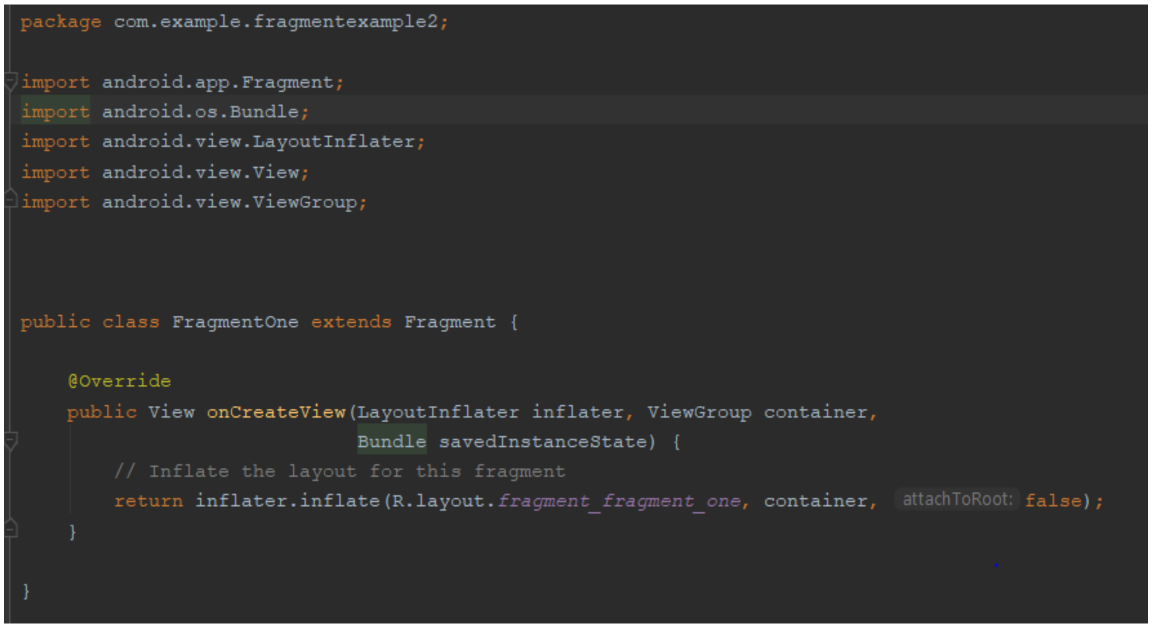
```xml
    <!-- Button to display second fragment -->
    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="20dp"
        android:layout_marginBottom="20dp"
        android:background="#00bfff"
        android:onClick="selectFragment"
        android:text="Display Second Fragment"
        android:textSize="18sp"
        android:textStyle="bold" />

    <!-- Adding Fragment element in the activity -->
    <fragment
        android:id="@+id/fragment_section"
        android:name="com.example.fragmentexample2.FragmentOne"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginStart="10dp"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="10dp"
        />

</LinearLayout>
```

- In here we have added 2 buttons to which will be used to switch between the 2 fragments.
- Further, we have added the fragment element and that area will be used to display the fragment.

*The **android:name** tag under the <fragment> element is containing the file name of default fragment which is to be displayed when activity opens.*

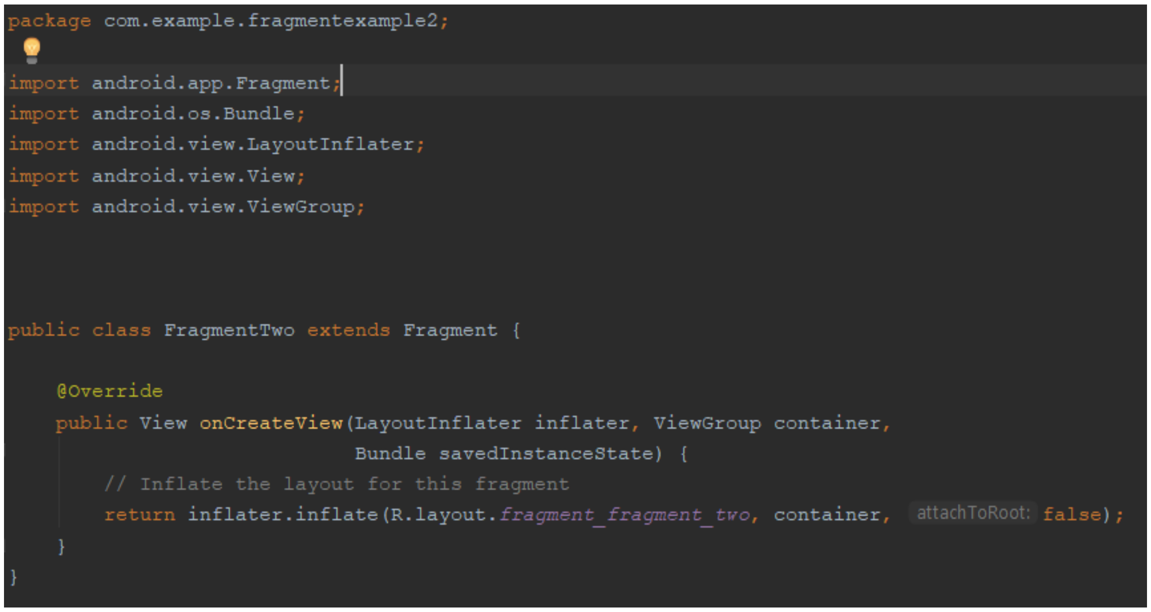
- Then create the 2 fragment classes.
- These files contain only the **onCreateView()** method to inflate the UI of the fragment and returns the root of the fragment layout. If the fragment does not have any UI, it will return null.

1. **First Fragment class:**



```
package com.example.fragmentexample2;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;



public class FragmentOne extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_fragment_one, container,  attachToRoot: false);
    }

}
```

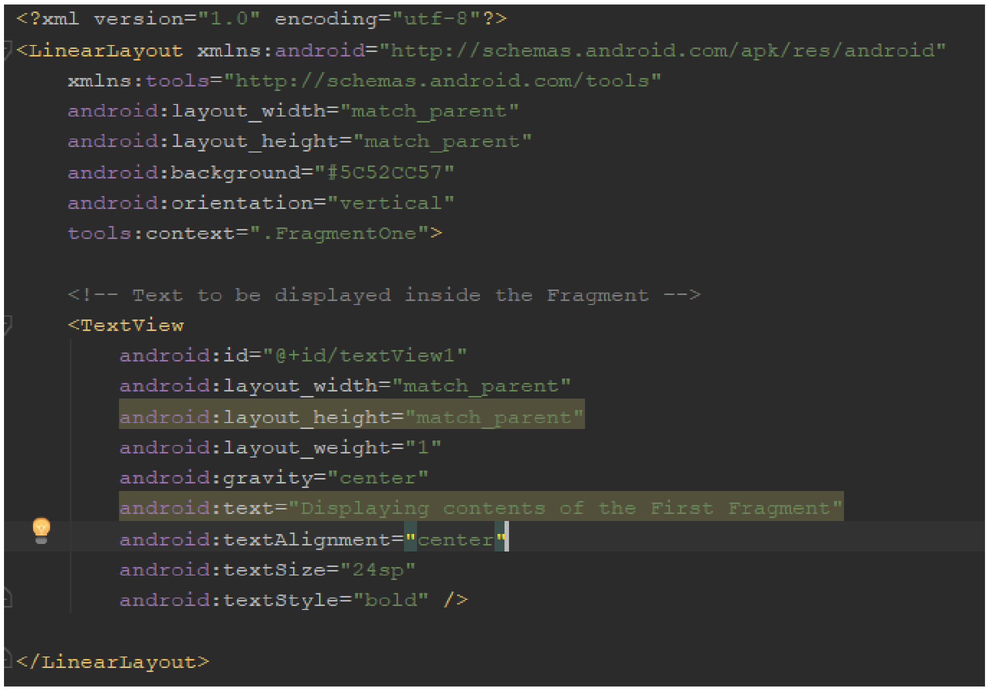
2. **Second Fragment class:**



```
package com.example.fragmentexample2;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;



public class FragmentTwo extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_fragment_two, container,  attachToRoot: false);
    }
}
```

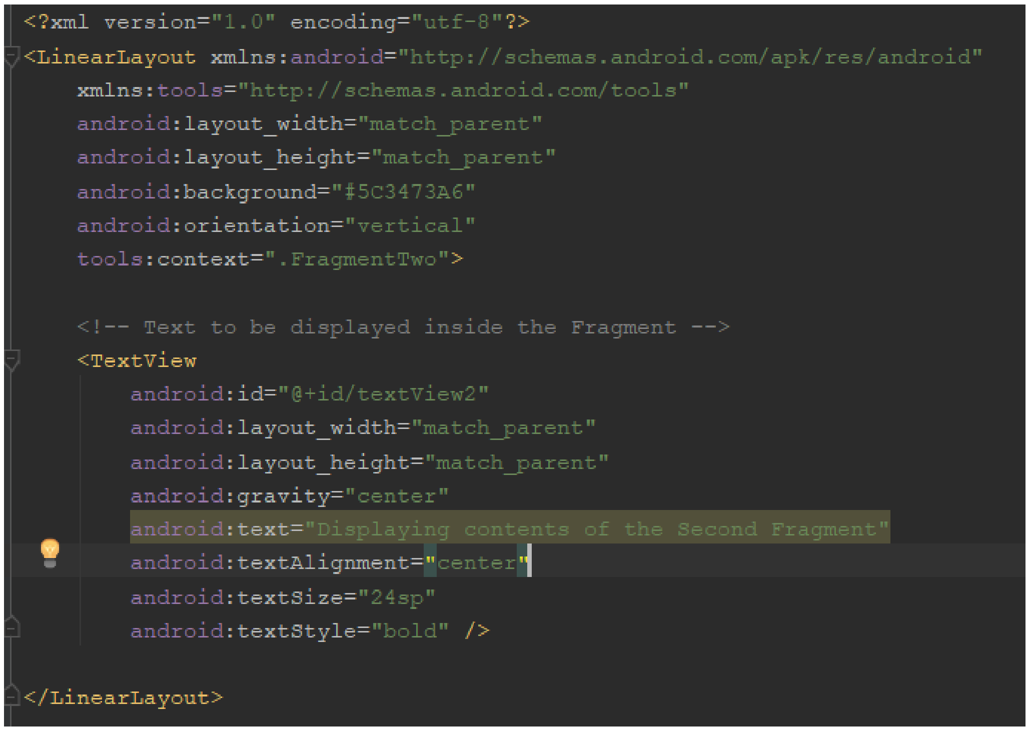
- Create two **Layout Resource Files** for both the fragments. Fragment displays a text on the screen and have a background color to differentiate their area in the Activity layout.

✓ **fragment_fragment_one.xml file:**



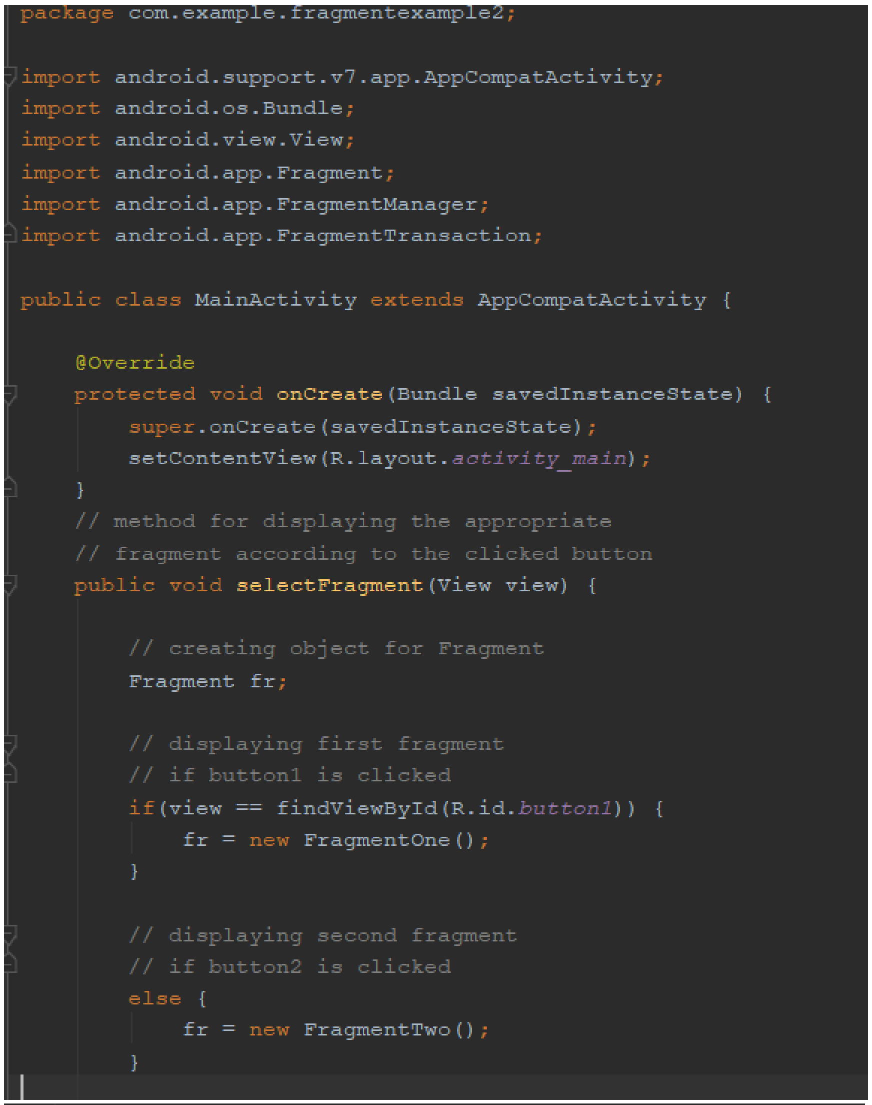
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#5C52CC57"
    android:orientation="vertical"
    tools:context=".FragmentOne">

    <!-- Text to be displayed inside the Fragment -->
    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:gravity="center"
        android:text="Displaying contents of the First Fragment"
        android:textAlignment="center"
        android:textSize="24sp"
        android:textStyle="bold" />

</LinearLayout>
```

✓ **fragment_fragment_two.xml file:**

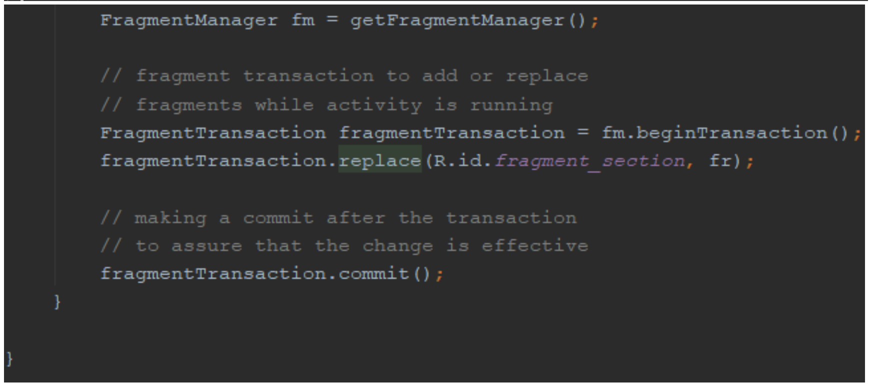


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#5C3473A6"
    android:orientation="vertical"
    tools:context=".FragmentTwo">

    <!-- Text to be displayed inside the Fragment -->
    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="Displaying contents of the Second Fragment"
        android:textAlignment="center"
        android:textSize="24sp"
        android:textStyle="bold" />

</LinearLayout>
```

- Now, the functionality of the button to perform operations on clicking will be defined in the **MainActivity** class. Moreover, the code for the replacement of fragments during run time is also mentioned in this file.

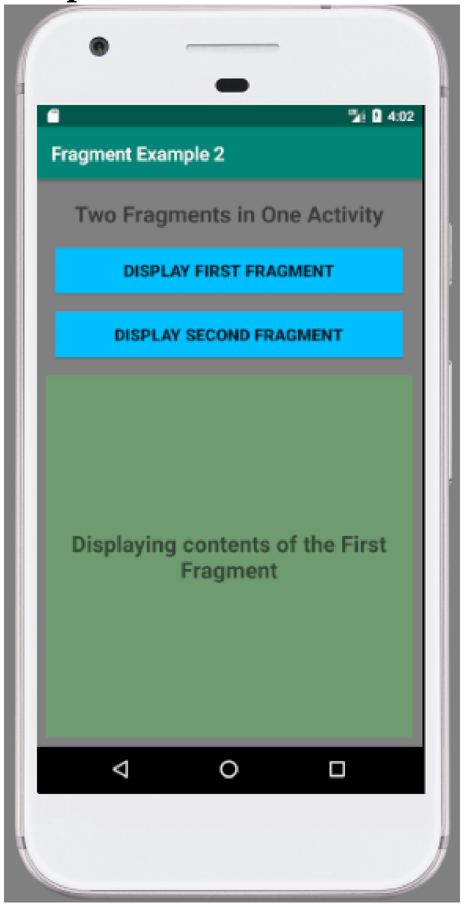


```
package com.example.fragmentexample2;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    // method for displaying the appropriate
    // fragment according to the clicked button
    public void selectFragment(View view) {

        // creating object for Fragment
        Fragment fr;

        // displaying first fragment
        // if button1 is clicked
        if(view == findViewById(R.id.button1)) {
            fr = new FragmentOne();
        }

        // displaying second fragment
        // if button2 is clicked
        else {
            fr = new FragmentTwo();
        }
```



```
        FragmentManager fm = getFragmentManager();

        // fragment transaction to add or replace
        // fragments while activity is running
        FragmentTransaction fragmentTransaction = fm.beginTransaction();
        fragmentTransaction.replace(R.id.fragment_section, fr);

        // making a commit after the transaction
        // to assure that the change is effective
        fragmentTransaction.commit();
    }

}
```

Output:

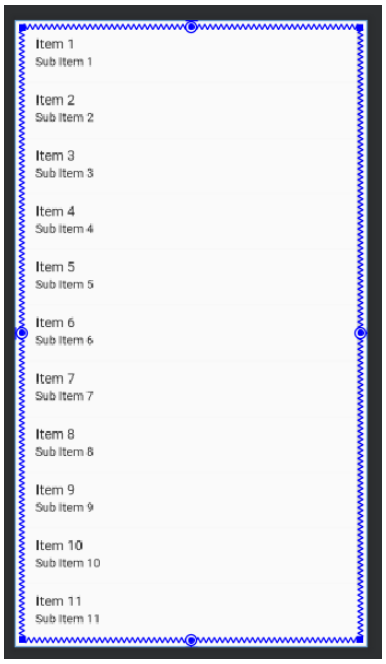

4:02
Fragment Example 2
Two Fragments in One Activity
DISPLAY FIRST FRAGMENT
DISPLAY SECOND FRAGMENT
Displaying contents of the First Fragment

<u>Adapter and Adapter View</u>

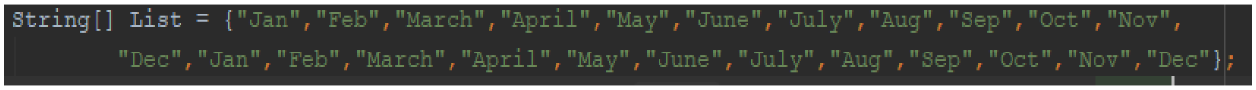
<u>Example 03 – List View</u>

- Create a new project.
- Open *activity_main.xml* and add a **ListView** and set an id for it.
- You can use either the Designer view or XML to add the list View.

  Design
  Legacy → ListView



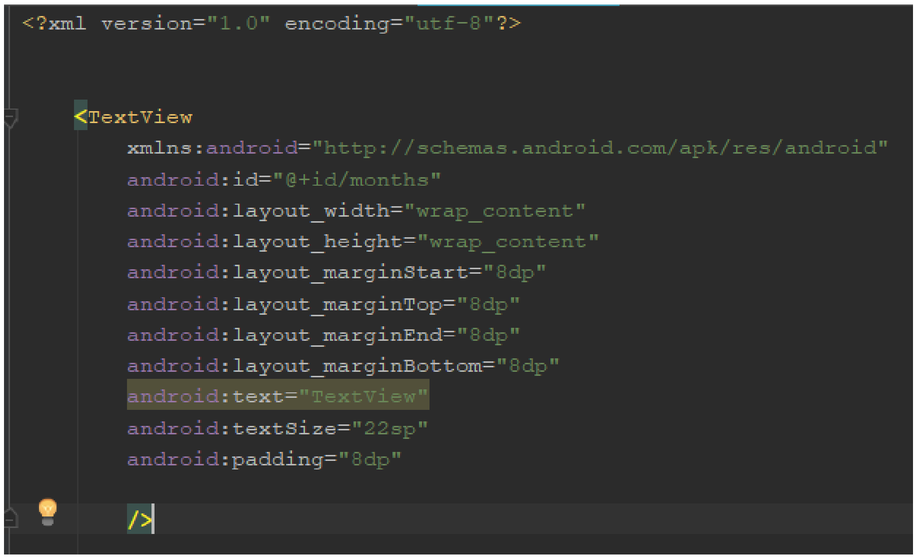
- So we need a **dataset** and a **View** into which the dataset will be converted by the Adapter. Here we have a simple Array with month names in it.



```
String[] List = {"Jan","Feb","March","April","May","June","July","Aug","Sep","Oct","Nov",
        "Dec","Jan","Feb","March","April","May","June","July","Aug","Sep","Oct","Nov","Dec"};
```

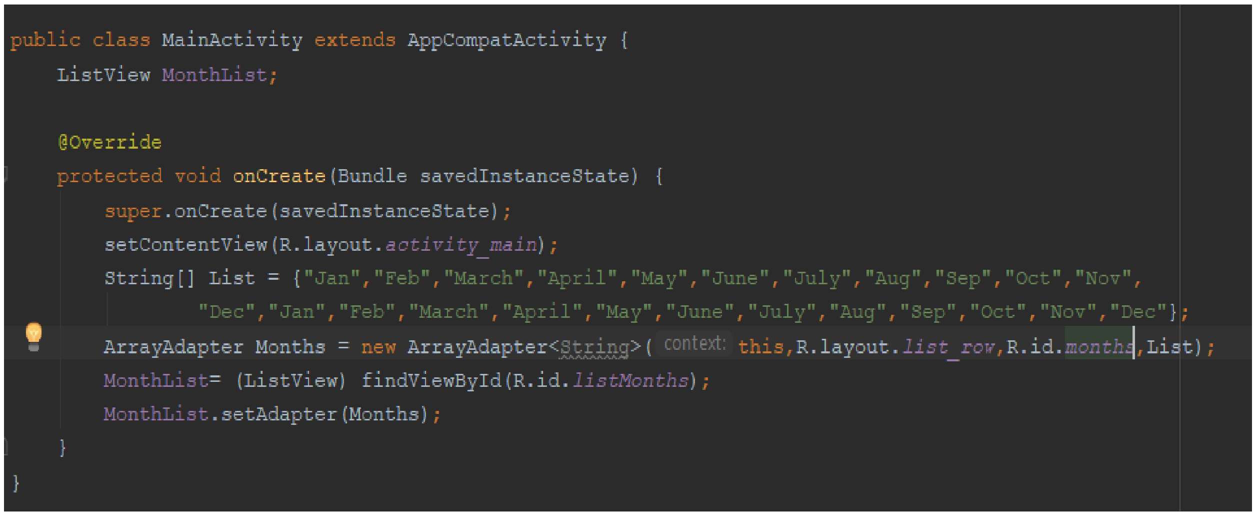
- As our data set has simple text values, so we can define a simple **TextView** to hold these values and populate the **ListView**.

- So now we will create a new XML file, with name **list_row.xml** in the layout folder, and add a **TextView** in it as follows,



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


    <TextView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/months"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="TextView"
        android:textSize="22sp"
        android:padding="8dp"

        />
```

- Then open MainActivity.java class, and add an **ArrayAdapter** to create text views from the data in the array, and create a list by supplying those view objects to the **ListView**.

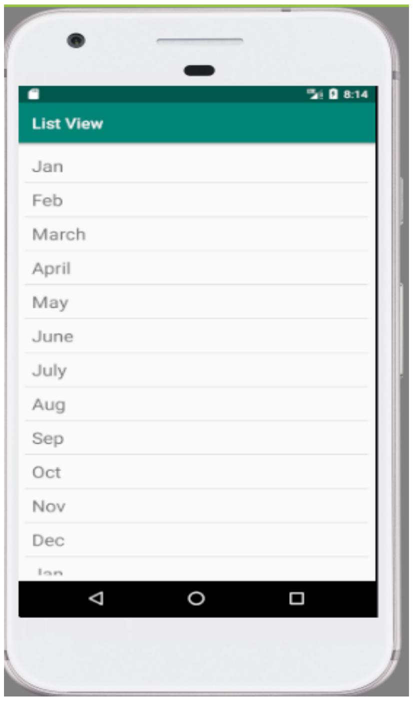


```java
public class MainActivity extends AppCompatActivity {
    ListView MonthList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        String[] List = {"Jan","Feb","March","April","May","June","July","Aug","Sep","Oct","Nov",
                "Dec","Jan","Feb","March","April","May","June","July","Aug","Sep","Oct","Nov","Dec"};
        ArrayAdapter Months = new ArrayAdapter<String>( context: this,R.layout.list_row,R.id.months,List);
        MonthList= (ListView) findViewById(R.id.listMonths);
        MonthList.setAdapter(Months);
    }
}
```

For the ArrayAdapter object you have to pass 4 parameters.

- ✓ The application context
- ✓ The layout file that you have created
- ✓ The TextView that you have created in the layout file
- ✓ The array that you have created

Output:

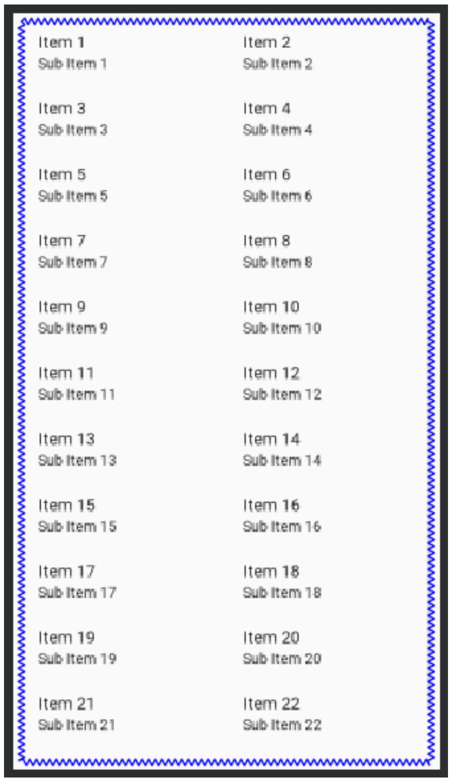


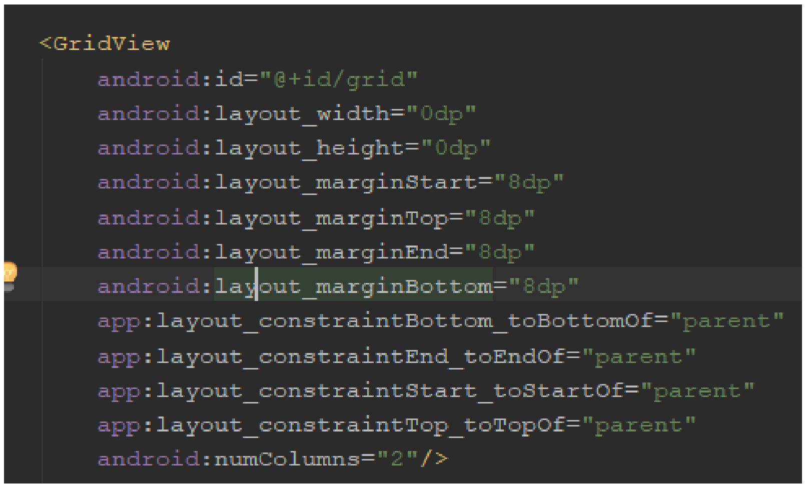
```
8:14
List View
Jan
Feb
March
April
May
June
July
Aug
Sep
Oct
Nov
Dec
Jan
```

Example 04 – Grid View

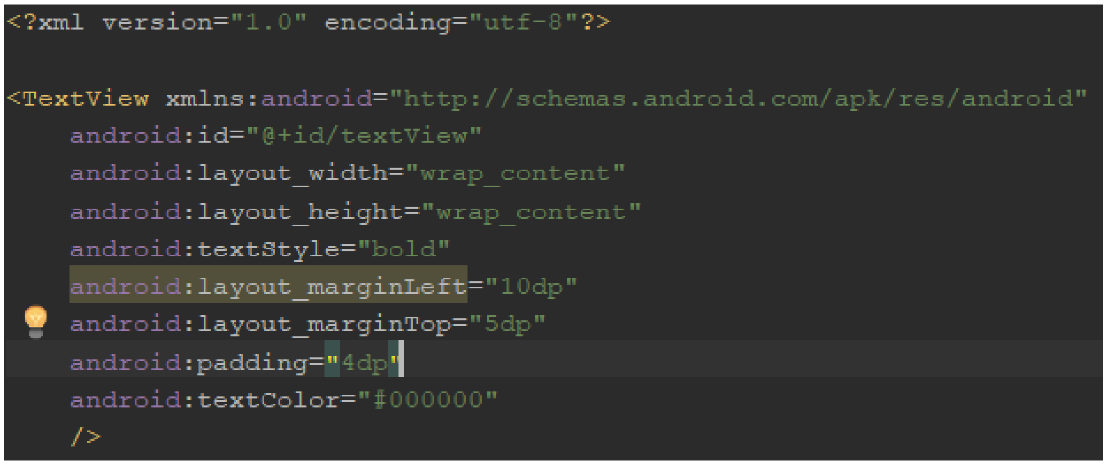
- Create a new project.
- Open *activity_main.xml* and add a **GridView** and set an id for it.
- You can use either the Designer view or XML to add the list View.

Design
Legacy → GridView



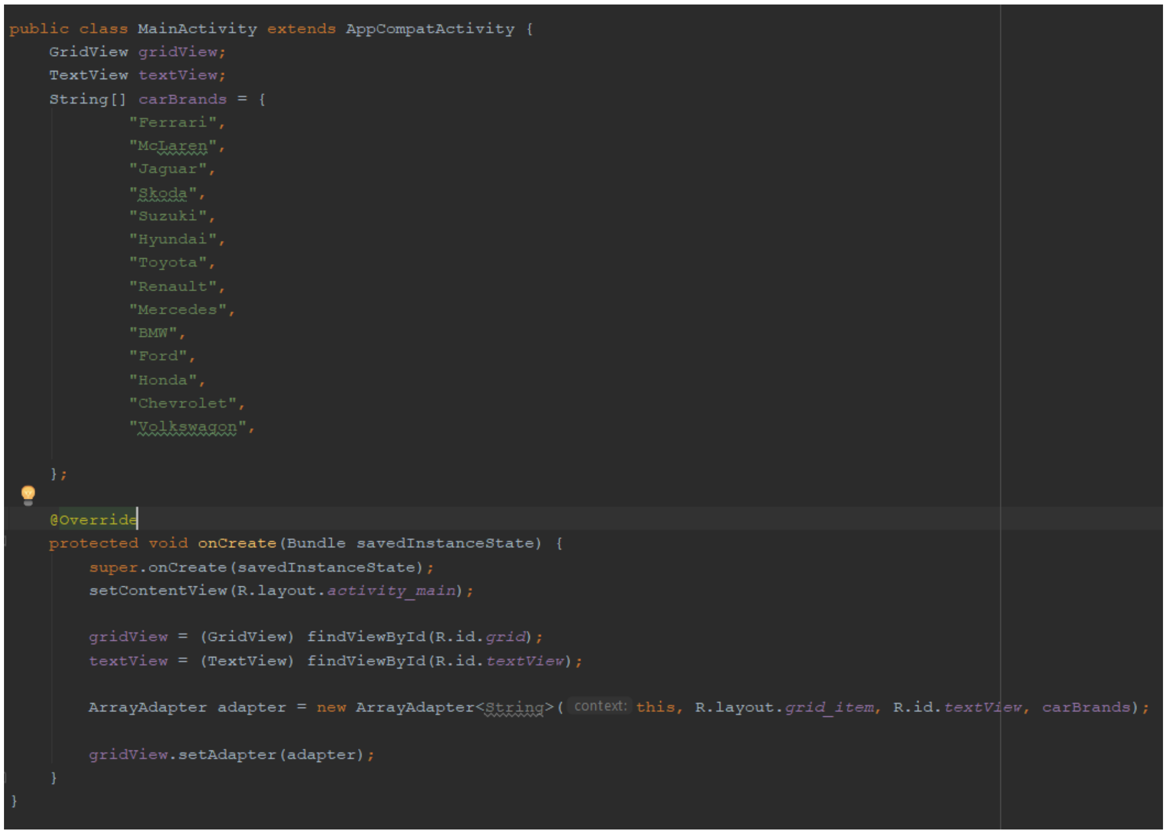

```xml
<GridView
    android:id="@+id/grid"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:numColumns="2"/>
```

- You have to define the number of columns using **numColumns** attribute. In here we have defined the number of columns as **2**.
- As our Grid will have only text values, hence we must define a TextView.
- So now we will create a new XML file, with name **grid_item.xml** in the layout folder, and add a TextView as follows:



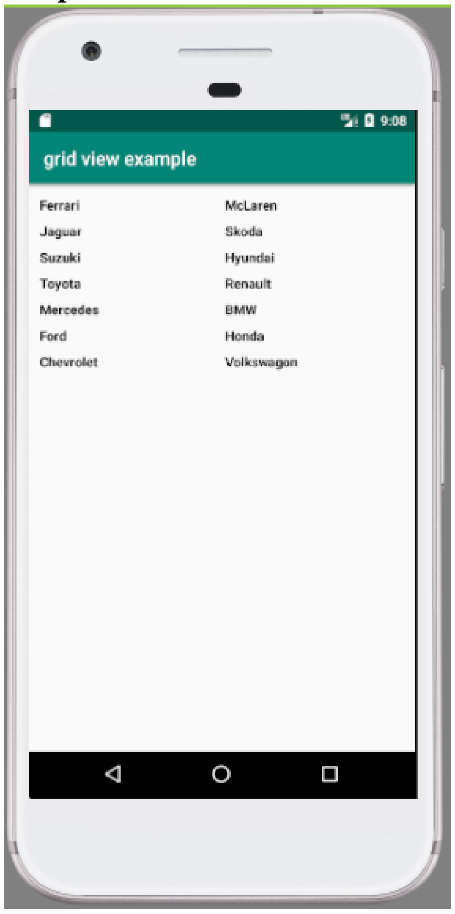
```
<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"
    android:padding="4dp"
    android:textColor="#000000"
    />
```

- Then open MainActivity.java class, and add an **ArrayAdapter** to create text views from the data in the array, and create a list by supplying those view objects to the **GridView**.



```
public class MainActivity extends AppCompatActivity {
    GridView gridView;
    TextView textView;
    String[] carBrands = {
            "Ferrari",
            "McLaren",
            "Jaguar",
            "Skoda",
            "Suzuki",
            "Hyundai",
            "Toyota",
            "Renault",
            "Mercedes",
            "BMW",
            "Ford",
            "Honda",
            "Chevrolet",
            "Volkswagon",

    };

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

        gridView = (GridView) findViewById(R.id.grid);
        textView = (TextView) findViewById(R.id.textView);

        ArrayAdapter adapter = new ArrayAdapter<String>( context: this, R.layout.grid_item, R.id.textView, carBrands);

        gridView.setAdapter(adapter);
    }
}
```

Output:



Ferrari	McLaren
Jaguar	Skoda
Suzuki	Hyundai
Toyota	Renault
Mercedes	BMW
Ford	Honda
Chevrolet	Volkswagon

Exercise 01- Static Fragments

Create an app that has two static fragments. They should be arranged in horizontal orientation. One should display an image and the other should display a text. (You can use any image and text as you wish)

Exercise 02 – Dynamic Fragments

Create an app that has 4 buttons as Name, Address, Phone and Email.

- When the user clicks Name Button your name should be displayed in a fragment of the same activity.
- When the user clicks Address Button your address should be displayed in a fragment.
- When the user clicks Phone Button your phone number should be displayed in a fragment.
- When the user clicks Email Button your email address should be displayed in a fragment.

Exercise 03 – List Views

Create a list of items which has a list of names (Eg: list of animals, list of vehicles… you can add the list as you wish) and in front each item an image should be repeated (repeat the same image.).

Exercise 04 – Grid Views

Create a grid of items (You can add the items as you wish) with **four** columns. When the user clicks a particular item from the list, the name of the item should be displayed in a Toast.