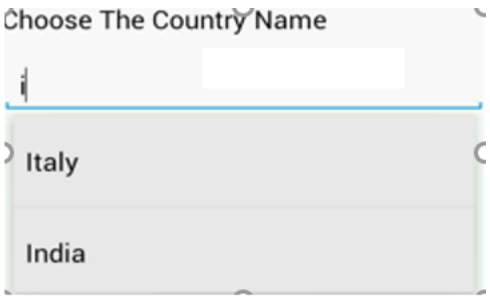
Android AutoCompleteTextView with Examples

AutoCompleteTextView is an editable text view which is used to show the list of suggestions based on the user typing text. The list of suggestions will be shown as a dropdown menu from which the user can choose an item to replace the content of the textbox.

The **AutoCompleteTextView** is a subclass of EditText class so we can inherit all the properties of EditText in AutoCompleteTextView based on our requirements.

Following is the pictorial representation of using AutoCompleteTextView in android applications.



Generally, the dropdown list of suggestions can be obtained from the data adaptor and those suggestions will be appeared only after giving the number characters defined in the **Threshold** limit.

The **Threshold** property of AutoCompleteTextView is used to define the minimum number of characters the user must type to see the list of suggestions.

The dropdown list of suggestions can be closed at any time in case if no item is selected from the list or by pressing the **back** or **enter** key.

In android, we can create an **AutoCompleteTextView** control in two ways either in the XML layout file or create it in the Activity file programmatically.

Create AutoCompleteTextView in Layout File

Following is the sample way to define **AutoCompleteTextView** control in the XML layout file in the android application.

```
<?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" >
    <AutoCompleteTextView
```

```
        android:id="@+id/autoComplete_Country"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

If you observe above code snippet, here we defined **AutoCompleteTextView** control in xml layout file.

Create AutoCompleteTextView Control in Activity File

In android, we can create **AutoCompleteTextView** control programmatically in an activity file to show the list of suggestions based on the user entered text.

Following is the example of creating **AutoCompleteTextView** control dynamically in activity file.

```
LinearLayout l_layout =  (LinearLayout) findViewById(R.id.linear_Layout);
AutoCompleteTextView actv = new AutoCompleteTextView(this);
  l_layout.addView(actv);
```

Set the Text of Android AutoCompleteTextView

In android, we can set the text of **AutoCompleteTextView** control by using **setAdapter()** method in Activity file.

Following is example of binding data AutoCompleteTextView in activity file using **setAdapter()** method.

```
String[] Countries = { "India", "USA", "Australia", "UK", "Italy", "Irel
and", "Africa" };
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.
layout.simple_dropdown_item_line, Countries);
AutoCompleteTextView actv = (AutoCompleteTextView)findViewById(R.id.auto
Complete_Country);
actv.setAdapter(adapter);
```

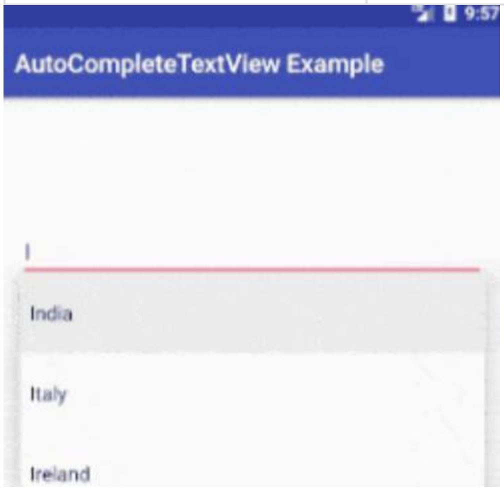
If you observe above code snippet, we are finding the **AutoCompleteTextView** control which we defined in XML layout file using id property and binding the data using **setAdapter()** method.

Android AutoCompleteTextView Attributes

Following are the some of commonly used attributes related to AutoCompleteTextView control in android applications.

Attribute	Description
android:id	It is used to uniquely identify the control
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text.
android:hint	It is used to display the hint text when text is empty
android:textColor	It is used to change the color of the text.
android:textColorHint	It is used to change the text color of hint text.
android:textSize	It is used to specify the size of text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for autocomplete textview control
android:ems	It is used to make the textview be exactly this many ems wide.
android:width	It makes the TextView be exactly this many pixels wide.
android:height	It makes the TextView be exactly this many pixels tall.

Attribute	Description
android:textColorHighlight	It is used to change the color of the text selection highlight.
android:fontFamily	It is used to specify the fontFamily for the text.



```
<?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:paddingLeft="20dp"
    android:paddingRight="20dp"
    android:orientation="vertical"
    android:id="@+id/linear_Layout">
    <AutoCompleteTextView
        android:id="@+id/ac_Country"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:hint="Enter Country Name"/>
</LinearLayout>
```

```
public class MainActivity extends AppCompatActivity {
    String[] Countries =
{ "India", "USA", "Australia", "UK", "Italy", "Ireland", "Africa" };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line, Countries);
        AutoCompleteTextView actv =
(AutoCompleteTextView)findViewById(R.id.ac_Country);
        actv.setThreshold(1);
        actv.setAdapter(adapter);
        actv.setOnItemClickListener(new AdapterView.OnItemClickListener(
) {
            @Override
            public void onItemClick(AdapterView<?> parent, View
view, int position, long id) {
                Toast.makeText(getApplicationContext(), "Selected Item:
" + parent.getSelectedItem(), Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Android ListView

ListView is a **ViewGroup** that is used to display the list of scrollable of items in multiple rows and the list items are automatically inserted to the list using an **adapter**.

Generally, the adapter pulls data from sources such as an array or database and converts each item into a result view and that's placed into the list.

Android Adapter

In android, **Adapter** will act as an intermediate between the data sources and adapter views such as ListView, Gridview to fill the data into adapter views. The adapter will hold the data and iterates through an items in data set and generate the views for each item in the list.

Generally, in android we have a different types of adapters available to fetch the data from different data sources to fill the data into adapter views, those are

Adapter	Description
ArrayAdapter	It will expects an Array or List as input.
CurosrAdapter	It will accepts an instance of cursor as an input.
SimpleAdapter	It will accepts a static data defined in the resources.
BaseAdapter	It is a generic implementation for all three adapter types and it can be used for ListView, Gridview or Spinners based on our requirements

Android ListView Example

Following is the example of creating a **ListView** using **arrayadapter** in android application.

Create a new android application using android studio and give names as **ListView.**
Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

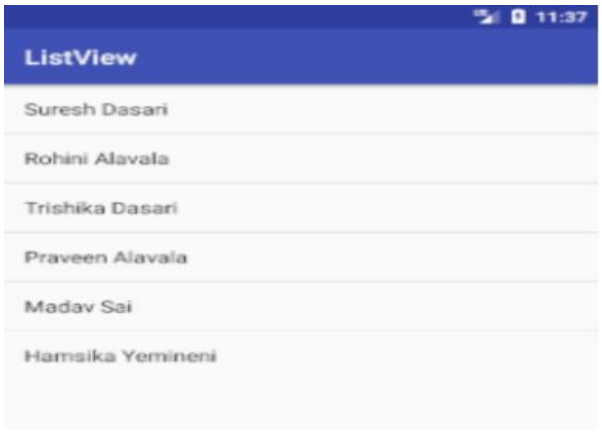
activity_main.xml

```
<?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:orientation="vertical">
    <ListView
        android:id="@+id/userlist"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>
</LinearLayout>
public class MainActivity extends AppCompatActivity {
    private ListView mListView;
    private ArrayAdapter aAdapter;
    private String[] users = { "Suresh Dasari", "Rohini Alavala", "Trish
ika Dasari", "Praveen Alavala", "Madav Sai", "Hamsika Yemineni"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mListView = (ListView) findViewById(R.id.userlist);
        aAdapter = new ArrayAdapter(this, android.R.layout.simple_list_i
tem_1, users);
        mListView.setAdapter(aAdapter);
    }
}
```

If you observe above code, we are binding static array (**users**) details to **ListView** using **ArrayAdapter** and calling our layout using **setContentView** method in the form of **R.layout.layout_file_name**. Here our xml file name is **activity_main.xml** so we used file name **activity_main**.

Output of Android ListView Example

When we run above example using android virtual device (AVD) we will get a result like as shown below.



ListView

Suresh Dasari

Rohini Alavala

Trishika Dasari

Praveen Alavala

Madav Sai

Hamsika Yemineni

Android Custom ListView (Adding Images, sub-title)

After creating simple ListView, android also provides facilities to customize our ListView.

As the simple ListView, custom ListView also uses Adapter classes which added the content from data source (such as string array, array, database etc). Adapter bridges data between an AdapterViews and other Views

Example of Custom ListView

In this custom listview example, we are adding image, text with title and its sub-title.

Structure of custom listview project

activity_main.xml

Create an activity_main.xml file in layout folder.

File: 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"
    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="com.example.test.listviewwithimage.MainActivity">

        <ListView
            android:id="@+id/list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="50dp">
        </ListView>
    </RelativeLayout>
```

Create an additional mylist.xml file in layout folder which contains view components displayed in listview.

mylist.xml

File: mylist.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="horizontal" >

    <ImageView
        android:id="@+id/icon"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:padding="5dp" />

    <LinearLayout android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">


<TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Medium Text"
        android:textStyle="bold"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:padding="2dp"
        android:textColor="#4d4d4d" />
    <TextView
        android:id="@+id/subtitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:layout_marginLeft="10dp"/>
    </LinearLayout>
</LinearLayout>
```

Place the all required images in drawable folder.

Activity class

File: MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    ListView list;
      String[] maintitle ={
            "Title 1","Title 2",
            "Title 3","Title 4",
            "Title 5",
    };
    String[] subtitle ={
            "Sub Title 1","Sub Title 2",
            "Sub Title 3","Sub Title 4",
            "Sub Title 5",
    };


    Integer[] imgid={
            R.drawable.download_1,R.drawable.download_2,
            R.drawable.download_3,R.drawable.download_4,
           R.drawable.download_5,
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       MyListAdapter adapter=new MyListAdapter(this, maintitle, subtitle,imgid);
      list=(ListView)findViewById(R.id.list);
       list.setAdapter(adapter);
list.setOnItemClickListener(new AdapterView.OnItemClickListener() {

           @Override
      public void onItemClick(AdapterView<?> parent, View view,int position, long id) {
        // TODO Auto-generated method stub
              if(position == 0) {
                //code specific to first list item
                  Toast.makeText(getApplicationContext(),"Place Your First Option Code",To
ast.LENGTH_SHORT).show();
             }
            else if(position == 1) {
                  //code specific to 2nd list item
                Toast.makeText(getApplicationContext(),"Place Your Second Option Code",Toast.LE
NGTH_SHORT).show();
              }
             else if(position == 2) {

                  Toast.makeText(getApplicationContext(),"Place Your Third Option Code",T
oast.LENGTH_SHORT).show();
             }
              else if(position == 3) {

                  Toast.makeText(getApplicationContext(),"Place Your Forth Option Code",T
oast.LENGTH_SHORT).show();
             }
              else if(position == 4) {
                    Toast.makeText(getApplicationContext(),"Place Your Fifth Option Code",To
ast.LENGTH_SHORT).show();
                }       }
            });
    }
}
```

Customize Our ListView

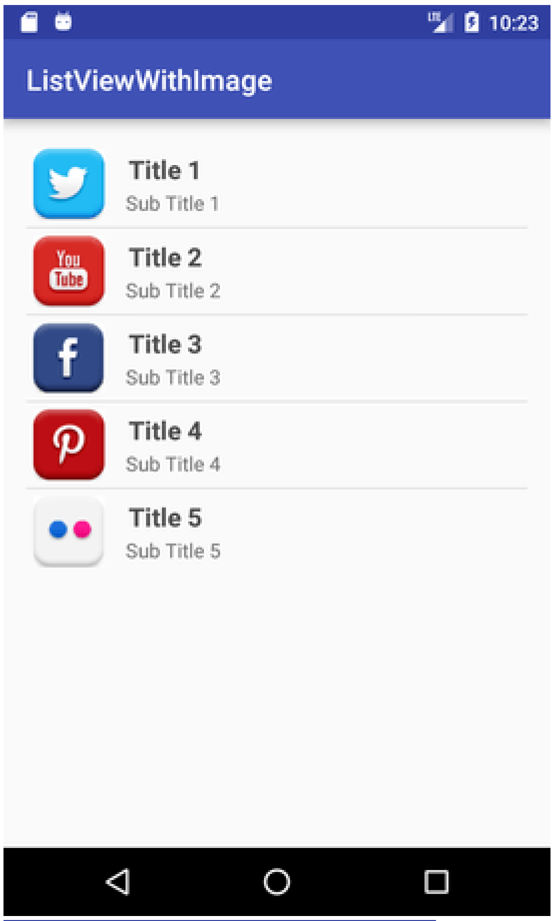
Create another java class MyListView.java which extends ArrayAdapter class. This class customizes our listview.

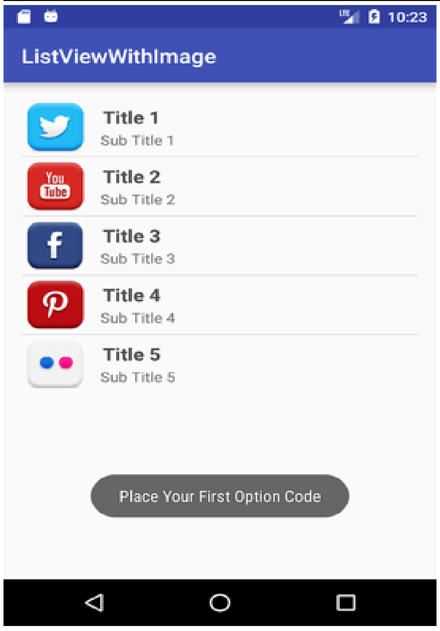
MyListView.java

```
public class MyListAdapter extends ArrayAdapter<String> {
  private final Activity context;
   private final String[] maintitle;
   private final String[] subtitle;
   private final Integer[] imgid;

   public MyListAdapter(Activity context, String[] maintitle,String[] subtitle, Integer[] im
gid) {
       super(context, R.layout.mylist, maintitle);
        // TODO Auto-generated constructor stub

        this.context=context;
       this.maintitle=maintitle;
        this.subtitle=subtitle;
       this.imgid=imgid;

   }

   public View getView(int position,View view,ViewGroup parent) {
       LayoutInflater inflater=context.getLayoutInflater();
       View rowView=inflater.inflate(R.layout.mylist, null,true);

       TextView titleText = (TextView) rowView.findViewById(R.id.title);
```

```
        ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);
        TextView subtitleText = (TextView) rowView.findViewById(R.id.subtitle);

        titleText.setText(maintitle[position]);
        imageView.setImageResource(imgid[position]);
        subtitleText.setText(subtitle[position]);

        return rowView;

    };
}
```

Output


10:23
ListViewWithImage
Title 1
Sub Title 1
Title 2
Sub Title 2
Title 3
Sub Title 3
Title 4
Sub Title 4
Title 5
Sub Title 5

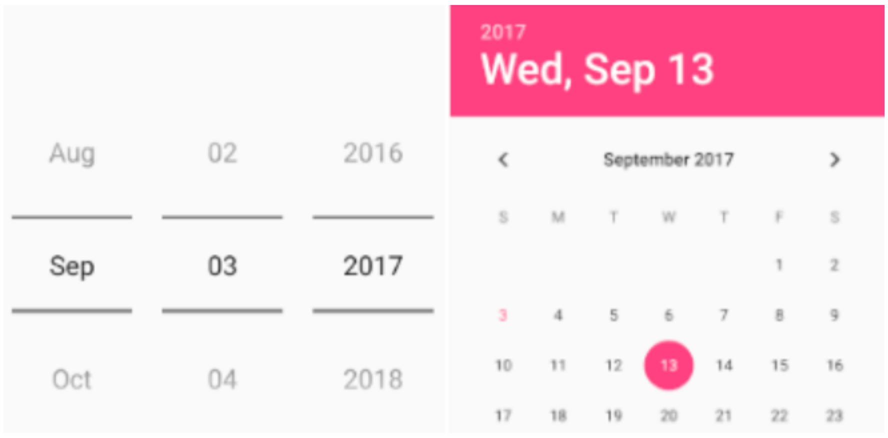

10:23
ListViewWithImage
Title 1
Sub Title 1
Title 2
Sub Title 2
Title 3
Sub Title 3
Title 4
Sub Title 4
Title 5
Sub Title 5
Place Your First Option Code

Android DatePicker Example

DatePicker is a control that will allow users to select the date by a day, month and year in our application user interface.

If we use **DatePicker** in our application, it will ensure that the users will select a valid date.

Following is the pictorial representation of using a datepicker control in android applications.



Generally, in android DatePicker available in two modes, one is to show the complete calendar and another one is to show the dates in spinner view.

Create Android DatePicker in XML Layout File

In android, we can create a DatePicker in XML layout file using **<DatePicker>** element with different attributes like as shown below

```
<DatePicker android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

In anroid, the DatePicker supports a two types of modes, those are **Calendar** and Spinner to show the date details in our application.

Android DatePicker with Calendar Mode

We can define android DatePicker to show only a calendar view by using DatePicker android:datePickerMode attribute.

Following is the example of showing the DatePicker in **Calendar** mode.

```
<DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:datePickerMode="calendar"/>
```

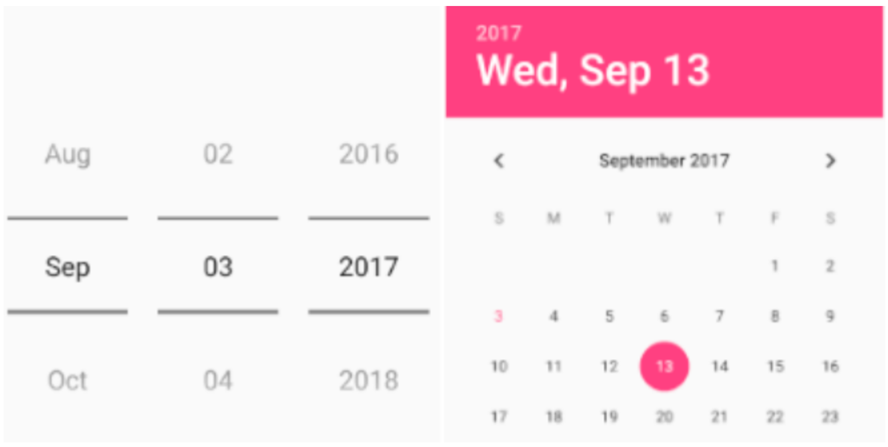
Android DatePicker with Spinner Mode

If we want to show the DatePicker in spinner format like showing day, month and year separately to select the date, then by using DatePicker android:datePickerMode attribute we can achieve this.

Following is the example of showing the DatePicker in Spinner mode.

```
<DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:datePickerMode="spinner"/>
```

The above code snippet will return the DatePicker in android like as shown below

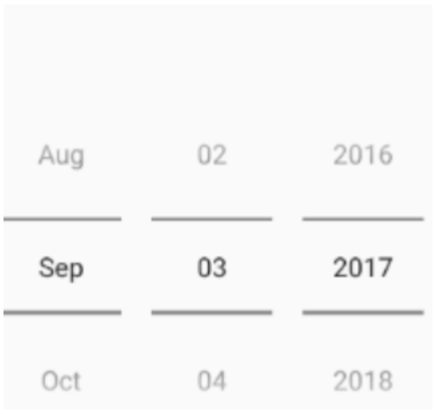


If you observe the above result we got the DatePicker in both **Spinner and Calendar** modes to select the date.

To get only spinner mode date selection, then we need to set android:calendarViewShown="false" attribute in DatePicker control like as shown below.

```
<DatePicker
    android:id="@+id/datePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:datePickerMode="spinner"
    android:calendarViewShown="false"/>
```

The above code will return the DatePicker like as shown below



If you observe the above result we got the DatePicker in spinner mode to select the date separately by day, month and year.

This is how we can use DatePicker in different modes based on our requirements in android applications.

Android DatePicker Control Attributes

The following are some of the commonly used attributes related to **DatePicker** control in android applications.

Attribute	Description
android:id	It is used to uniquely identify the control
android:datePickerMode	It is used to specify datepicker mode either spinner or calendar
android:background	It is used to set the background color for the date picker.
android:padding	It is used to set the padding for left, right, top or bottom of the date picker.

Android DatePicker Example

Following is the example of defining one **DatePicker** control, one TextView control and one Button control in RelativeLayout to show the selected date on Button click in the android application.

Create a new android application using android studio and give names as **DatePickerExample**. Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/androi
d"
    android:layout_width="match_parent" android:layout_height="match_par
ent">
    <DatePicker
        android:id="@+id/datePicker1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp" />
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/datePicker1"
        android:layout_marginLeft="100dp"
        android:text="Get Date" />
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button1"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="10dp"
        android:textStyle="bold"
        android:textSize="18dp"/>
</RelativeLayout>
```

If you observe above code we created a one DatePicker control, one TextView control and one Button control in XML Layout file.

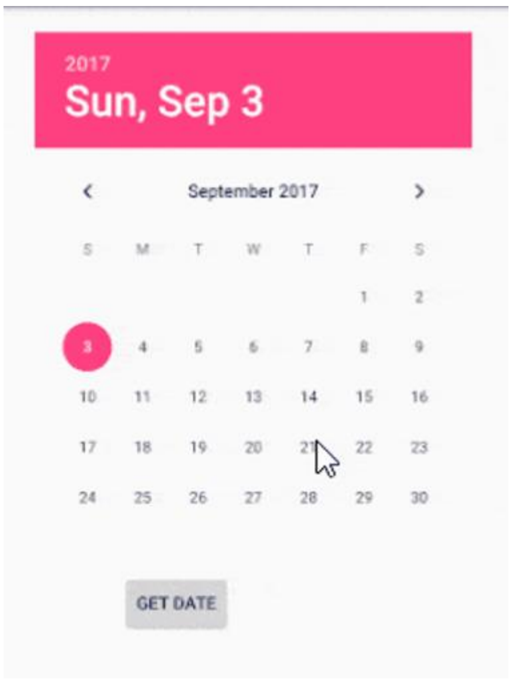
MainActivity.java

```

public class MainActivity extends AppCompatActivity {
    DatePicker picker;
    Button btnGet;
    TextView tvw;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvw=(TextView)findViewById(R.id.textView1);
        picker=(DatePicker)findViewById(R.id.datePicker1);
        btnGet=(Button)findViewById(R.id.button1);
        btnGet.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                tvw.setText("Selected Date: "+ picker.getDayOfMonth()+"/
"+ (picker.getMonth() + 1)+"/"+picker.getYear());
            }
        });
    }
}
```
If you observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name** in our activity file. Here our xml file name is **activity_main.xml** so we used file name **activity_main** and we are trying to show the selected date of DatePicker on Button click.

Generally, during the launch of our activity, the **onCreate()** callback method will be called by the android framework to get the required layout for an activity.

Output of Android DatePicker Example

When we run the above example using an android virtual device (AVD) we will get a result like as shown below.

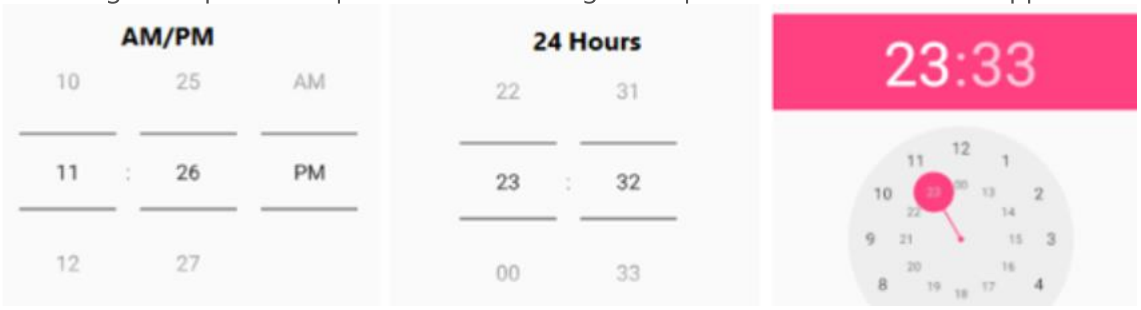


Android TimePicker Example

TimePicker is a widget for selecting the time of day, in either 24-hour or AM/PM mode.

If we use **TimePicker** in our application, it will ensure that the users will select a valid time for the day.

Following is the pictorial representation of using a timepicker control in android applications



Generally, in android TimePicker available in two modes, one is to show the time in clock mode and another one is to show the time in spinner mode.

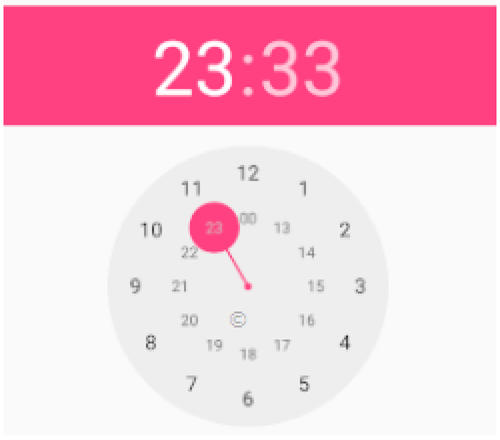
Android TimePicker with Clock Mode

We can define android TimePicker to show time in clock format by using TimePicker android:timePickerMode attribute.

Following is the example of showing the TimePicker in **Clock** mode.

```
<TimePicker android:id="@+id/timePicker1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:timePickerMode="clock" />
```

The above code will return the TimePicker like as shown below.



If you observe the above result we got the **TimePicker** in **clock** mode to select a time in Hours and Minutes based on our requirements.

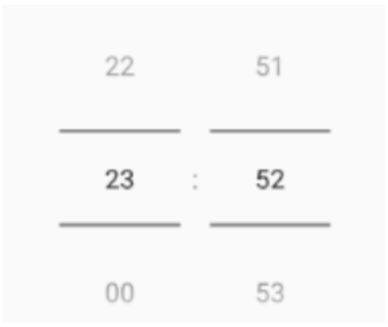
Android TimePicker with Spinner Mode

If we want to show the TimePicker in spinner format like showing hours and minutes separately to select the time, then by using TimePicker android:timePickerMode attribute we can achieve this.

Following is the example of showing the TimePicker in spinner mode.

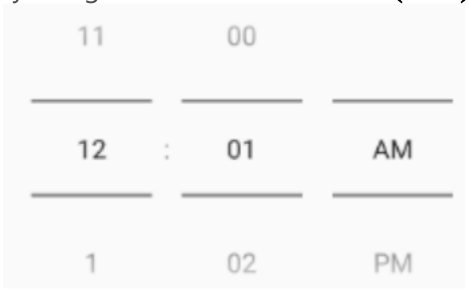
```
<TimePicker
   android:id="@+id/datePicker1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:timePickerMode="spinner"/>
```

The above code will return the TimePicker like as shown below



If you observe the above result we got the TimePicker in spinner mode to select the time in Hours and Minutes.

We can change the TimePicker in spinner mode to **AM** / **PM** format instead of **24 Hours** format by using the **setIs24HourView(true)** method in Activity file like as shown below.



```
TimePicker picker=(TimePicker)findViewById(R.id.timePicker1);
picker.setIs24HourView(true);
```

Android TimePicker Control Attributes

The following are some of the commonly used attributes related to **TimePicker** control in android applications.

Attribute	Description
android:id	It is used to uniquely identify the control
android:timePickerMode	It is used to specify timepicker mode, either spinner or clock
android:background	It is used to set the background color for the date picker.
android:padding	It is used to set the padding for left, right, top or bottom of the date picker.

Android TimePicker Example

Let's see a simple example of android time picker.

activity_main.xml

File: activity_main.xml

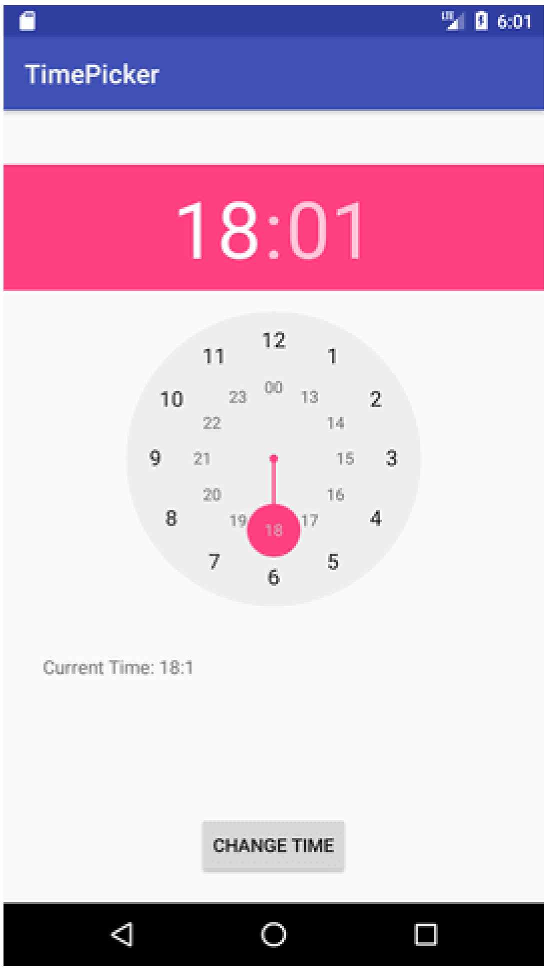
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button1"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginBottom="102dp"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:text="" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp"
        android:text="Change Time" />

    <TimePicker
        android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="36dp" />
</RelativeLayout>
```

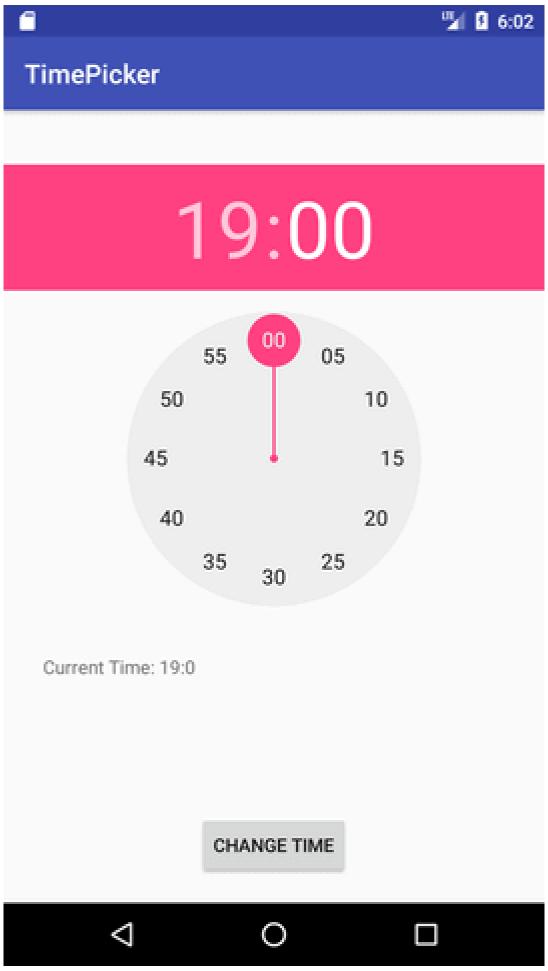
Activity class

File: MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    TextView textview1;
    TimePicker timepicker;
    Button changetime;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textview1=(TextView)findViewById(R.id.textView1);
        timepicker=(TimePicker)findViewById(R.id.timePicker);
         //Uncomment the below line of code for 24 hour view
        timepicker.setIs24HourView(true);
         changetime=(Button)findViewById(R.id.button1);

         textview1.setText(getCurrentTime());

         changetime.setOnClickListener(new View.OnClickListener(){
            @Override
             public void onClick(View view) {
                textview1.setText(getCurrentTime());
             }
        });

    }

    public String getCurrentTime(){
         String currentTime="Current Time: "+timepicker.getCurrentHour()+":"+timepicker
.getCurrentMinute();
        return currentTime;
    }

}
```

Output:





Android ScrollView (Vertical and Horizontal)

ScrollView is a kind of layout that is useful to add vertical or horizontal scroll bars to the content which is larger than the actual size of layouts such as linearlayout, relativelayout, framelayout, etc.

Generally, the android **ScrollView** is useful when we have content that doesn't fit our android app layout screen. The ScrollView will enable a scroll to the content which is exceeding the screen layout and allow users to see the complete content by scrolling.

The android **ScrollView** can hold only one direct child. In case, if we want to add multiple views within the scroll view, then we need to include them in another standard layout like linearlayout, relativelayout, framelayout, etc.

To enable scrolling for our android applications, **ScrollView** is the best option but we should not use ScrollView along with ListView or Gridview because they both will take care of their own vertical scrolling.

In android, **ScrollView** supports only **vertical** scrolling. In case, if we want to implement **horizontal** scrolling, then we need to use a **HorizontalScrollView** component.

The android ScrollView is having a property called android:fillViewport, which is used to define whether the ScrollView should stretch it's content to fill the viewport or not.

Now we will see how to use **ScrollView** with linearlayout to enable scroll view to the content which is larger than screen layout in android application with examples.

Android ScrollView Example

Following is the example of enabling vertical scrolling to the content which is larger than the layout screen using an android **ScrollView** object.

Create a new android application using android studio and give names as **ScrollViewExample**. Once we create an application, open **activity_main.xml** file from **\res\layout** folder path and write the code like as shown below.

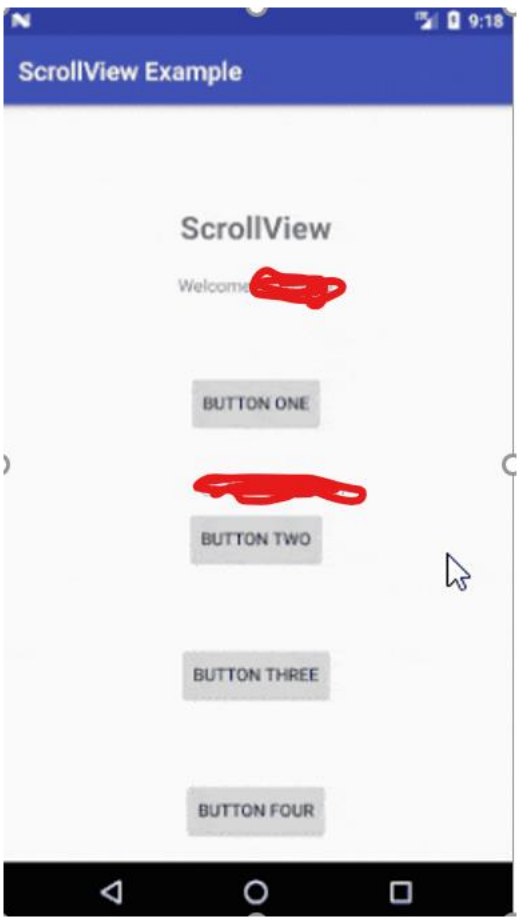
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fillViewport="false">
<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/loginscrn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="80dp"
        android:text="ScrollView"
        android:textSize="25dp"
        android:textStyle="bold"
        android:layout_gravity="center"/>
    <TextView android:id="@+id/fstTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Welcome "
        android:layout_gravity="center"/>
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button One" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button Two" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button Three" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button Four" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button Five" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button Six" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button Seven" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button Eight" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="60dp"
        android:text="Button Nine" />
</LinearLayout>
</ScrollView>
```

If you observe above code, we used a **ScrollView** to enable the scrolling for linearlayout whenever the content exceeds layout screen.

Output of Android ScrollView Example

When we run the above example in android emulator we will get a result as shown below.



If you observe the above result, **ScrollView** provided a vertical scrolling for linearlayout whenever the content exceeds the layout screen.

As we discussed, **ScrollView** can provide only vertical scrolling for the layout. In case, if we want to enable horizontal scrolling, then we need to use **HorizontalScrollView** in our application.

We will see how to enable horizontal scrolling for the content which is exceeding the layout screen in the android application.

Android HorizontalScrollView

Example

Now open **activity_main.xml** file in your android application and write the code like as shown below.

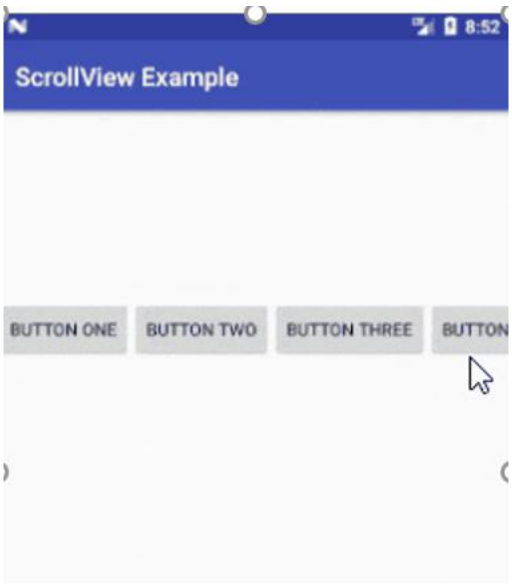
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<HorizontalScrollView xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:fillViewport="true">
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="150dp">
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button One" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button Two" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button Three" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button Four" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button Five" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button Six" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button Seven" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button Eight" />
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button Nine" />
</LinearLayout>
</HorizontalScrollView>
```

If you observe above code, we used a **HorizontalScrollView** to enable horizontal scrolling for linearlayout whenever the content exceeds layout screen.

Output of Android HorizontalScrollView Example

When we run the above example in the android emulator we will get a result like as shown below.



If you observe above result, **HorizontalScrollView** provided a horizontal scrolling for linearlayout whenever the content exceeds the layout screen.

This is how we can enable scrolling for the content which exceeds layout screen using **ScrollView** and **HorizontalScrollView** object based on our requirements.

Android Image Switcher

ImageSwitcher is a specialized view switcher that will provide a smooth transition animation effect to the images while switching from one image to another.

To use **ImageSwitcher** in our application, we need to define an XML component .xml like as shown below.

```
<ImageSwitcher android:id="@+id/imgSw"
    android:layout_width="match_parent"
    android:layout_height="250dp">
</ImageSwitcher>
```

After that we need to create an instance of **ImageSwitcher** and use **setFactory()** method to implement the **ViewFactory** interface to return the ImageView. We need to add the required animation effects to ImageSwitcher based on our requirements.

Following is the syntax of using **ImageSwitcher** to add a smooth transition animation effect to images while switching from one image to another.

```
ImageSwitcher imgsw = (ImageSwitcher) findViewById(R.id.imgSw);
imgsw.setFactory(new ViewSwitcher.ViewFactory() {
    @Override
    public View makeView() {
        ImageView imgVw= new ImageView(MainActivity.this);
        imgVw.setImageResource(images[position]);
        return imgVw;
    }
});
imgsw.setInAnimation(this, android.R.anim.slide_in_left);
imgsw.setOutAnimation(this, android.R.anim.slide_out_right);
```

If you observe above code snippet, we created an instance of **ImageSwitcher**, used **setFactory()** method to implement **ViewFactory** interface and added required animation effects.

Apart from the above methods, the **ImageSwitcher** class is having different methods available, those are

Method	Description
setImageDrawable	This method is used to set a new drawable on the next ImageView in the switcher.
setImageResource	This method is used to set a new image on the ImageSwitcher with the given resource id.
setImageURI	This method is used to set a new image on the ImageSwitcher with the given Uri.

Now we will see how to use ImageSwitcher to add smooth transition animation effects while switching between the images in android application with examples.

Android ImageSwitcher Example

Following is the example of switching between the images with a smooth animation transition effect using **imageswitcher** in the android application.

Create a new android application using android studio and give names as **ImageSwitcherExample**.
Now open **activity_main.xml** file from **\res\layout** folder path and write the code like as shown below.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/androi
d"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">
    <ImageSwitcher android:id="@+id/imgSw"
        android:layout_width="match_parent"
        android:layout_height="250dp"/>
    <Button
        android:id="@+id/btnPrevious"
        android:layout_below="@+id/imgSw"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Previous" android:layout_marginLeft="100dp" />
    <Button
        android:id="@+id/btnNext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btnPrevious"
        android:layout_toRightOf="@+id/btnPrevious"
        android:text="Next" />
</RelativeLayout>
```

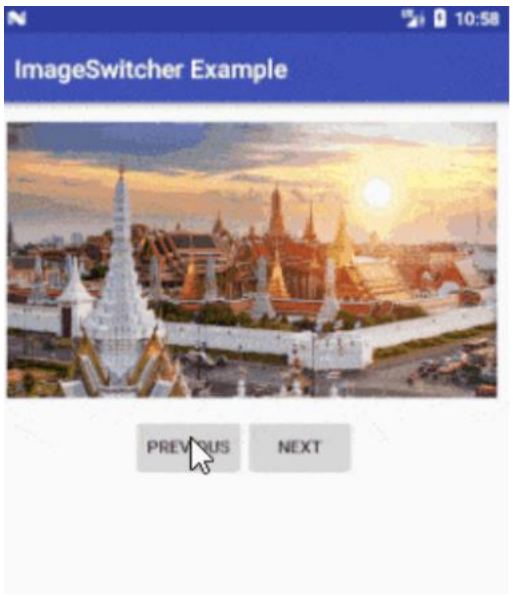
Now open your main activity file **MainActivity.java** from **\java\com.tutlane.imageswitcherexample** path and write the code like as shown below

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    private Button previousbtn, nextbtn;
    private ImageSwitcher imgsw;
    private int[] images = {R.drawable.bangkok,R.drawable.bangkok2};
    private int position = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        previousbtn = (Button)findViewById(R.id.btnPrevious);
        nextbtn = (Button)findViewById(R.id.btnNext);
        imgsw = (ImageSwitcher) findViewById(R.id.imgSw);
        imgsw.setFactory(new ViewSwitcher.ViewFactory() {
            @Override
            public View makeView() {
                ImageView imgVw= new ImageView(MainActivity.this);
                imgVw.setImageResource(images[position]);
                return imgVw;
            }
        });
        imgsw.setInAnimation(this, android.R.anim.slide_in_left);
        imgsw.setOutAnimation(this, android.R.anim.slide_out_right);
        previousbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(position>0)
                position--;
                else if(position<0)
                    position = 0;
                imgsw.setImageResource(images[position]);
            }
        });
        nextbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(position<images.length)
                    position++;
                if(position>=images.length)
                    position = images.length-1;
                imgsw.setImageResource(images[position]);
            }
        });
    }
}
```
If you observe above code, we used a set of images (**bangkok**, **bangkok2**) from **drawable** folder, you need to add your images in **drawable** folder and by using imageswitcher **setFactory()** and **setImageResource()** methods we are switching the images with slide in / slide out animation effect.

Output of Android ImageSwitcher Example

When we run the above program in the android studio we will get the result as shown below.



Android Image Slider

Android *image slider* slides one entire screen to another screen. Image slider is created by **ViewPager** which is provided by support library. To implement image slider, you need to inherit ViewPager class which extends PagerAdapter.

Example of Image Slider

Let's see an example of android image slider.

activity_main.xml

In activity_main.xml file, we have wrapped ViewPager inside RelativeLayout.

File: 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="fill_parent"
   android:layout_height="fill_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"
```

```
  tools:context="com.example.test.imageslider.MainActivity">


  <android.support.v4.view.ViewPager
    android:id="@+id/viewPage"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />

</RelativeLayout>
```

Activity class

File: MainActivity.java

```
public class MainActivity extends AppCompatActivity {

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

    ViewPager mViewPager = (ViewPager) findViewById(R.id.viewPage);
    ImageAdapter adapterView = new ImageAdapter(this);
    mViewPager.setAdapter(adapterView);
  }
}
```

ImageAdapter class

Now create ImageAdapter class which extends PagerAdapter for android image slider.

Place some images in drawable folder which are to be slid.

File: ImageAdapter.java

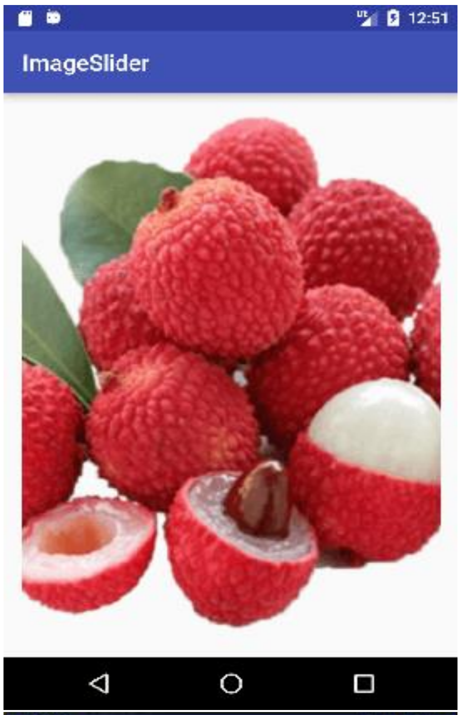
```
public class ImageAdapter extends PagerAdapter{
  Context mContext;

  ImageAdapter(Context context) {
    this.mContext = context;
  }
```

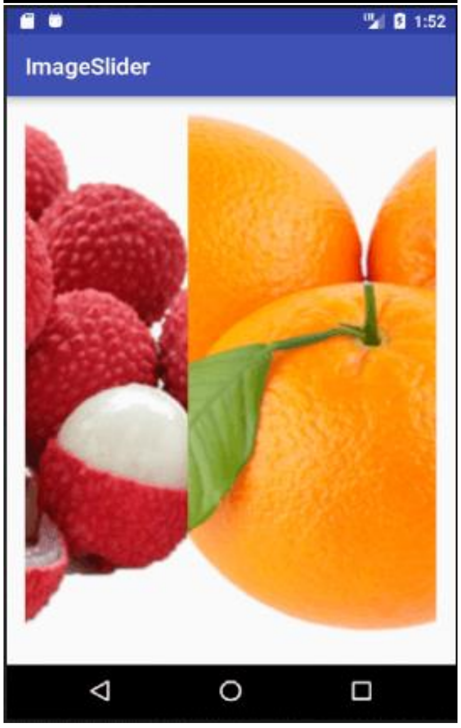
```
     @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == ((ImageView) object);
    }
    private int[] sliderImageId = new int[]{
            R.drawable.image1, R.drawable.image2, R.drawable.image3,R.drawable.i
mage4, R.drawable.image5,
    };

    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        ImageView imageView = new ImageView(mContext);
        imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
        imageView.setImageResource(sliderImageId[position]);
        ((ViewPager) container).addView(imageView, 0);
        return imageView;
    }
     @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        ((ViewPager) container).removeView((ImageView) object);
    }

    @Override
    public int getCount() {
        return sliderImageId.length;
    }
}
```

We need to override following methods of PagerAdapter class.

1. **isViewFromObject(View, Object):** This method checks the view whether it is associated with key and returned by instantiateItem().
2. **instantiateItem(ViewGroup, int):** This method creates the page position passed as an argument.
3. **destroyItem(ViewGroup, int, Object):** It removes the page from its current position from container. In this example we simply removed object using removeView().
4. **getCount():** It returns the number of available views in ViewPager.

Output





Android TabLayout

TabLayout is used to implement horizontal tabs. TabLayout is released by Android after the deprecation of *ActionBar.TabListener (API level 21)*.

TabLayout is introduced in design support library to implement tabs.

Tabs are created using *newTab()* method of TabLayout class. The title and icon of Tabs are set through *setText(int)* and *setIcon(int)* methods of TabListener interface respectively. Tabs of layout are attached over TabLayout using the method addTab(Tab) method.

```
TabLayout tabLayout = (TabLayout)findViewById(R.id.tabLayout);
tabLayout.addTab(tabLayout.newTab().setText("Tab 1"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 2"));
tabLayout.addTab(tabLayout.newTab().setText("Tab 3"));
```

We can also add tab item to TabLayout using TabItem of android design widget.

```
<android.support.design.widget.TabItem
            android:text="@string/tab_text"/>
```

Example of TabLayout using ViewPager

Let's create an example of TabLayout using ViewPager and Fragment.

File: activity.xml

Create an activity.xml file with TabLayout and ViewPager view components.

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

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/ap
k/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="tablayout.example.com.tablayout.MainActivity">
      <android.support.design.widget.TabLayout
        android:id="@+id/tabLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#1db995">
    </android.support.design.widget.TabLayout>
      <android.support.v4.view.ViewPager
```

```
 android:id="@+id/viewPager"
 android:layout_width="355dp"
 android:layout_height="455dp"
 app:layout_constraintTop_toBottomOf="@+id/tabLayout"
 tools:layout_editor_absoluteX="8dp" />
</android.support.constraint.ConstraintLayout>
```

File: build.gradle

Now gave the dependency library of TabLayout in build.gradle file.

1. implementation 'com.android.support:design:26.1.0'

File: MainActivity.java

In this file, we implement two additional listener *addOnPageChangeListener(listener)* of ViewPager which makes slides the different fragments of tabs and *addOnTabSelectedListener(listener)* of TabLayout which select the current tab on tab selection.

```
public class MainActivity extends AppCompatActivity {
    TabLayout tabLayout;
    ViewPager viewPager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tabLayout=(TabLayout)findViewById(R.id.tabLayout);
        viewPager=(ViewPager)findViewById(R.id.viewPager);

        tabLayout.addTab(tabLayout.newTab().setText("Home"));
        tabLayout.addTab(tabLayout.newTab().setText("Sport"));
        tabLayout.addTab(tabLayout.newTab().setText("Movie"));
        tabLayout.setTabGravity(TabLayout.GRAVITY_FILL);

        final MyAdapter adapter = new MyAdapter(this,getSupportFragmentManager(), tabLayo
ut.getTabCount());
        viewPager.setAdapter(adapter);
        viewPager.addOnPageChangeListener(new TabLayout.TabLayoutOnPageChangeL
istener(tabLayout));

        tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab) {
                viewPager.setCurrentItem(tab.getPosition());
            }

            @Override
            public void onTabUnselected(TabLayout.Tab tab) {

            }

            @Override
            public void onTabReselected(TabLayout.Tab tab) {

            }
        });

    }
}
```

File: MyAdapter.java

```
public class MyAdapter extends FragmentPagerAdapter {

    private Context myContext;
    int totalTabs;

    public MyAdapter(Context context, FragmentManager fm, int totalTabs) {
        super(fm);
        myContext = context;
        this.totalTabs = totalTabs;
    }
```

```java
    // this is for fragment tabs
    @Override
    public Fragment getItem(int position) {
      switch (position) {
          case 0:
            HomeFragment homeFragment = new HomeFragment();
             return homeFragment;
         case 1:
             SportFragment sportFragment = new SportFragment();
            return sportFragment;
          case 2:
            MovieFragment movieFragment = new MovieFragment();
             return movieFragment;
         default:
             return null;
      }
    }
 // this counts total number of tabs
    @Override
    public int getCount() {
       return totalTabs;
    }
 }
```

Now create different fragment files for all different tabs.

File: HomeFragment.java

```java
public class HomeFragment extends Fragment {
   public HomeFragment() {
      // Required empty public constructor
   }

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

```
    }

}
```

File: fragment_home.xml

```
<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="tablayout.example.com.tablayout.HomeFragment">

    <!-- TODO: Update blank fragment layout -->

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/home_fragment" />

</FrameLayout>
```

File: SportFragment.java

```
package tablayout.example.com.tablayout;

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

public class SportFragment extends Fragment {


    public SportFragment() {
        // Required empty public constructor
    }
```

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

}
```

File: fragment_sport.xml

```xml
<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="tablayout.example.com.tablayout.SportFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/sport_fragment" />

</FrameLayout>
```

File: MovieFragment.java

```java
public class MovieFragment extends Fragment {


    public MovieFragment() {
        // Required empty public constructor
    }

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

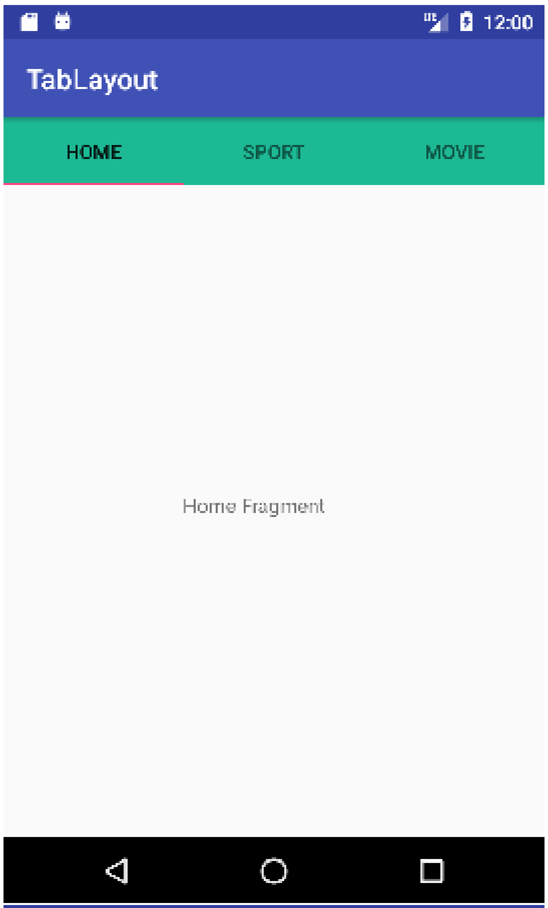
File: fragment_movie.xml

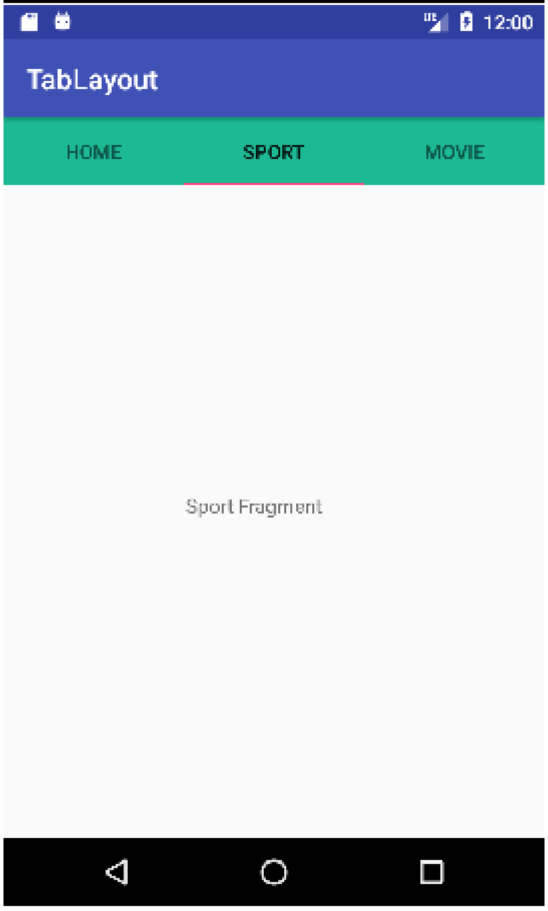
```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="tablayout.example.com.tablayout.MovieFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/movie_fragment" />

</FrameLayout>
```

File: strings.xml

```
        <resources>
            <string name="app_name">TabLayout</string>

            <!-- TODO: Remove or change this placeholder text -->
            <string name="home_fragment">Home Fragment</string>
            <string name="sport_fragment">Sport Fragment</string>
            <string name="movie_fragment">Movie Fragment</string>

        </resources>
```

Output


12:00
TabLayout
HOME
SPORT
MOVIE
Home Fragment


12:00
TabLayout
HOME
SPORT
MOVIE
Sport Fragment

Android TabLayout with FrameLayout

In the previous page, we created a sliding tabs using TabLayout and ViewPager. Here, we are going to create non sliding tabs using *TabLayout* and *FrameLayout*.

Items of TabLayout are implemented by adding *TabItem* of android support design widget.

Example of TabLayout using FrameLayout

Let's create an example of TabLayout using FrameLayout and Fragment.

File: activity.xml

Create an activity.xml file with TabLayout and FrameLayout view components.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout 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="tablayout.example.com.tablayoutwithframelayout.MainActivity">


    <android.support.design.widget.TabLayout
        android:id="@+id/tabLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#7367">

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Home" />

        <android.support.design.widget.TabItem
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Java" />

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Android" />

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Php" />
    </android.support.design.widget.TabLayout>

    <FrameLayout
        android:id="@+id/frameLayout"
        android:layout_width="match_parent"
        android:layout_height="455dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tabLayout">

    </FrameLayout>
</android.support.constraint.ConstraintLayout>
```

File: build.gradle

Now gave the dependency library of TabLayout in build.gradle file.

1. implementation 'com.android.support:design:26.1.0'

File: MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    TabLayout tabLayout;
    FrameLayout frameLayout;
    Fragment fragment = null;
    FragmentManager fragmentManager;
    FragmentTransaction fragmentTransaction;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tabLayout=(TabLayout)findViewById(R.id.tabLayout);
        frameLayout=(FrameLayout)findViewById(R.id.frameLayout);

        fragment = new HomeFragment();
        fragmentManager = getSupportFragmentManager();
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.frameLayout, fragment);
        fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OP
EN);
        fragmentTransaction.commit();

        tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab) {
                // Fragment fragment = null;
                switch (tab.getPosition()) {
                    case 0:
                        fragment = new HomeFragment();
                        break;
                    case 1:
                        fragment = new JavaFragment();
                        break;
                    case 2:
                        fragment = new AndroidFragment();
                        break;
                    case 3:
                        fragment = new PhpFragment();
                        break;
                }
                FragmentManager fm = getSupportFragmentManager();
                FragmentTransaction ft = fm.beginTransaction();
               ft.replace(R.id.frameLayout, fragment);
                ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
               ft.commit();
            }

            @Override
           public void onTabUnselected(TabLayout.Tab tab) {

           }

           @Override
            public void onTabReselected(TabLayout.Tab tab) {

            }
        });

    }
}
```

Now create different fragment files for all different tabs.

File: HomeFragment.java

```
public class HomeFragment extends Fragment {

    public HomeFragment() {
       // Required empty public constructor
    }

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

File: fragment_home.xml

```
<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="tablayout.example.com.tablayoutwithframelayout.HomeFragment">
      <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/home_fragment" />
  </FrameLayout>
```

File: JavaFragment.java

```
public class JavaFragment extends Fragment {
    public JavaFragment() {
      // Required empty public constructor
   }

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

File: fragment_java.xml

```
<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="tablayout.example.com.tablayoutwithframelayout.JavaFragment">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/java_fragment" />
</FrameLayout>
```

File: AndroidFragment.java

```
public class AndroidFragment extends Fragment {

    public AndroidFragment() {
        // Required empty public constructor
    }

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

}
```

File: fragment_android.xml

```
<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="tablayout.example.com.tablayoutwithframelayout.AndroidFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
```

```
      android:text="@string/android_fragment" />
  </FrameLayout>
```

File: PhpFragment.java

```
public class PhpFragment extends Fragment {

    public PhpFragment() {
        // Required empty public constructor
    }

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

}
```

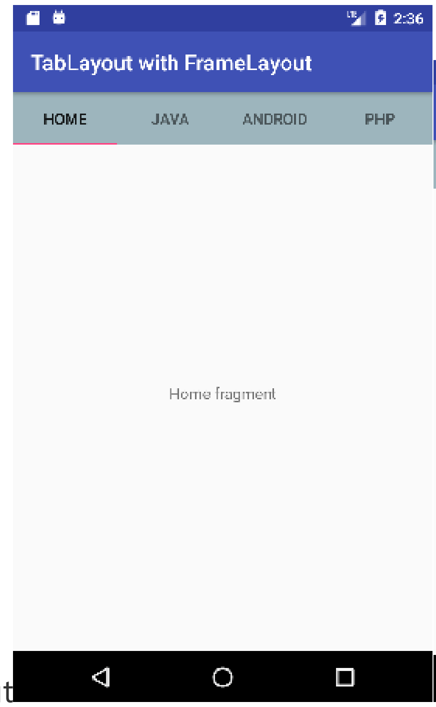
File: fragment_php.xml

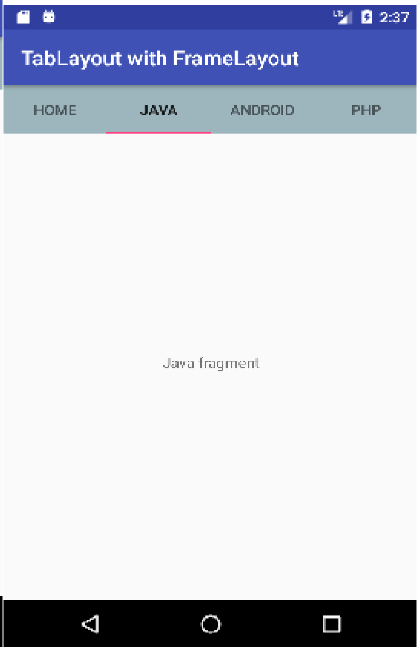
```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="tablayout.example.com.tablayoutwithframelayout.PhpFragment">
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/php_fragment" />
</FrameLayout>
```

File: strings.xml

```xml
<resources>
    <string name="app_name">TabLayout with FrameLayout</string>

    <!-- TODO: Remove or change this placeholder text -->
    <string name="home_fragment">Home fragment</string>
    <string name="java_fragment">Java fragment</string>
    <string name="android_fragment">Android fragment</string>
    <string name="php_fragment">Php fragment</string>
</resources>
```

Output





Android SearchView

Android **SearchView** provides user interface to search query submitted over search provider. SearchView widget can be implemented over ToolBar/ActionBar or inside a layout.

SearchView is by default collapsible and set to be iconified using *setIconifiedByDefault(true)* method of SearchView class. For making search field visible, SearchView uses setIconifiedByDefault(false) method.

Methods of SearchView

1. **public boolean onQueryTextSubmit(String query):** It searches the query on the submission of content over SearchView editor. It is case dependent.
2. **public boolean onQueryTextChange(String newText):** It searches the query at the time of text change over SearchView editor.

Example of SearchView

Let's see the example of SearchView over layout, searching data in a ListView.

activity_main.xml

Create an activity_main.xml file in layout folder containing ScrollView and ListView.

File: activity_main.xml

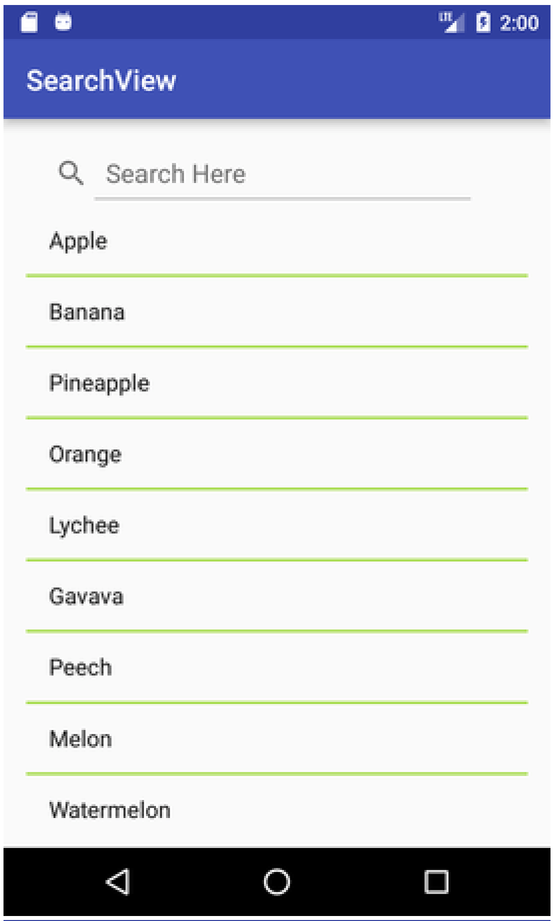
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.test.searchview.MainActivity">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/lv1"
        android:divider="#ad5"
        android:dividerHeight="2dp"
        android:layout_below="@+id/searchView"/>

    <SearchView
        android:id="@+id/searchView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:queryHint="Search Here"
        android:iconifiedByDefault="false"
        android:layout_alignParentTop="true"
    />
  </RelativeLayout>
```

Activity class

File: MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    SearchView searchView;
    ListView listView;
    ArrayList<String> list;
    ArrayAdapter<String > adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        searchView = (SearchView) findViewById(R.id.searchView);
        listView = (ListView) findViewById(R.id.lv1);

        list = new ArrayList<>();
        list.add("Apple");
        list.add("Banana");
        list.add("Pineapple");
        list.add("Orange");
        list.add("Lychee");
        list.add("Gavava");
        list.add("Peech");
        list.add("Melon");
        list.add("Watermelon");
        list.add("Papaya");

        adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,list);
        listView.setAdapter(adapter);

        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {

                if(list.contains(query)){
                    adapter.getFilter().filter(query);
                }else{
                    Toast.makeText(MainActivity.this, "No Match found",Toast.LENGTH_LONG).show();

                }
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
            //    adapter.getFilter().filter(newText);
                return false;
            }
        });
    }
}
```

Output



2:00

SearchView

Search Here

Apple

Banana

Pineapple

Orange

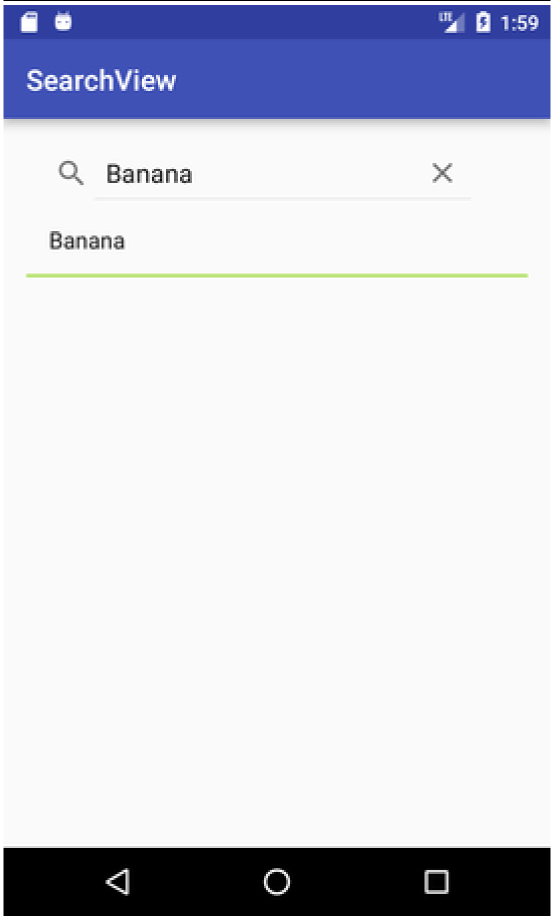
Lychee

Gavava

Peech

Melon

Watermelon

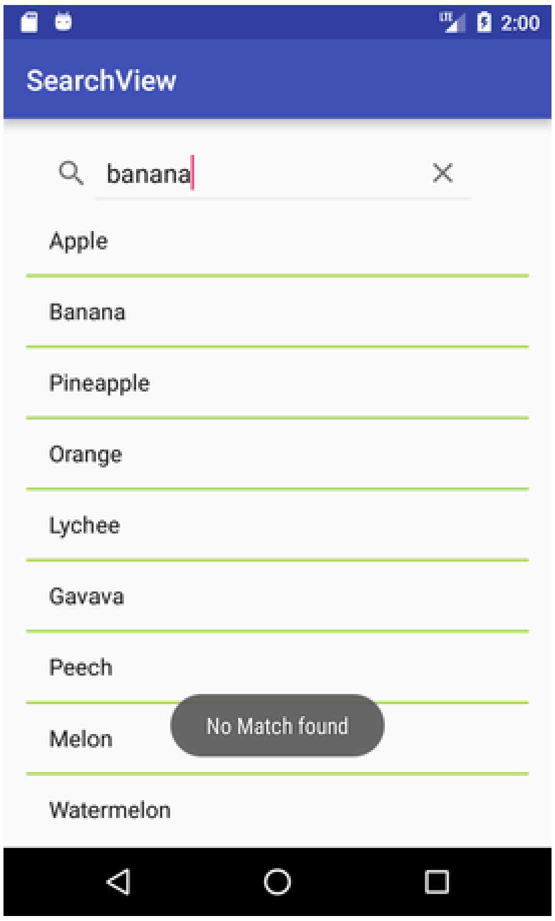


1:59

SearchView

Banana

Banana



2:00

SearchView

banana

Apple

Banana

Pineapple

Orange

Lychee

Gavava

Peech

Melon

No Match found

Watermelon

Android EditText with TextWatcher (Searching data from ListView)

Android **EditText** is a subclass of *TextView*. EditText is used for entering and modifying text. While using EditText width, we must specify its input type in *inputType* property of EditText which configures the keyboard according to input.

EditText uses **TextWatcher** interface to watch change made over EditText. For doing this, EditText calls the *addTextChangedListener()* method.

Methods of TextWatcher

1. **beforeTextChanged(CharSequence arg0, int arg1, int arg2, int arg3):** It is executed before making any change over EditText.
2. **onTextChanged(CharSequence cs, int arg1, int arg2, int arg3):** It is executed while making any change over EditText.
3. **afterTextChanged(Editable arg0):** It is executed after change made over EditText.

Example of EditText with TextWatcher()

In this example, we will implement EditText with TextWatcher to search data from ListView.

activity_main.xml

Create an activity_main.xml file in layout folder containing EditText and ListView.

[redacted]

File: 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"
    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="com.example.test.searchfromlistview.MainActivity">
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:inputType="text"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_below="@+id/editText"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

Create another file list_item.xml in layout folder which contains data of ListView.

list_item.xm

File: list_item.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/product_name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="10dip"
```

```
   android:textSize="16dip"
   android:textStyle="bold"/>
</LinearLayout>
```

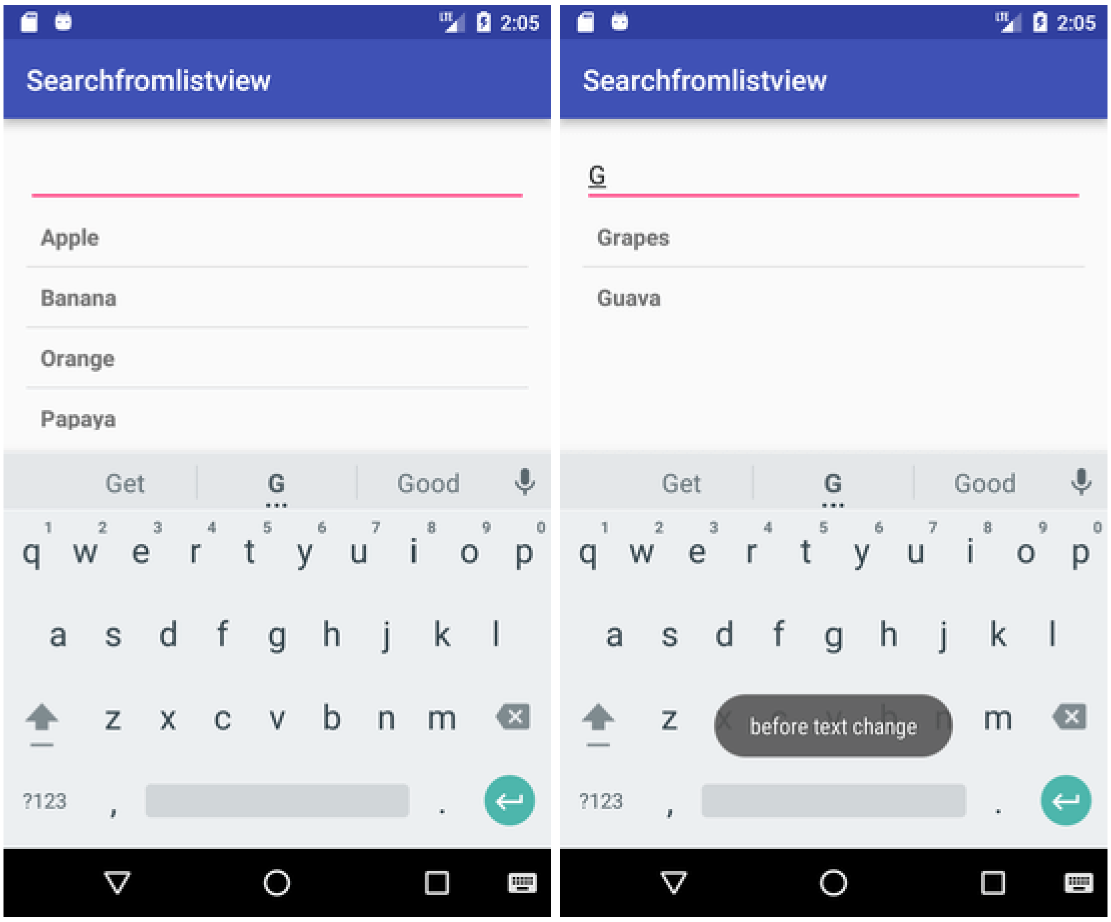
Activity class

Activity class

```
public class MainActivity extends AppCompatActivity {

   private ListView lv;
   private EditText editText;
   private ArrayAdapter<String> adapter;

   private String products[] = {"Apple", "Banana","Pinapple", "Orange", "Papaya", "Melon",
         "Grapes", "Water Melon","Lychee", "Guava", "Mango", "Kivi"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      lv = (ListView) findViewById(R.id.listView);
      editText = (EditText) findViewById(R.id.editText);
      adapter = new ArrayAdapter<String>(this, R.layout.list_item, R.id.product_name,
products);
      lv.setAdapter(adapter);

      editText.addTextChangedListener(new TextWatcher() {

         @Override
         public void onTextChanged(CharSequence cs, int arg1, int arg2, int arg3) {
            adapter.getFilter().filter(cs);
         }

         @Override
         public void beforeTextChanged(CharSequence arg0, int arg1, int arg2, int arg3) {
            Toast.makeText(getApplicationContext(),"before text change",Toast.LENGTH_
LONG).show();
```

```
          }

          @Override
           public void afterTextChanged(Editable arg0) {
             Toast.makeText(getApplicationContext(),"after text change",Toast.LENGTH_LONG).sh
ow();
           }
      });
   }
}
```

Output



Searchfromlistview

Apple
Banana
Orange
Papaya

Searchfromlistview

G

Grapes
Guava

before text change

Android TabLayout with FrameLayout

In the previous page, we created a sliding tabs using TabLayout and ViewPager. Here, we are going to create non sliding tabs using *TabLayout* and *FrameLayout*.

Items of TabLayout are implemented by adding *TabItem* of android support design widget.

Example of TabLayout using FrameLayout

Let's create an example of TabLayout using FrameLayout and Fragment.

File: activity.xml

Create an activity.xml file with TabLayout and FrameLayout view components.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout 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="tablayout.example.com.tablayoutwithframelayout.MainActivity">


    <android.support.design.widget.TabLayout
        android:id="@+id/tabLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#7367">

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Home" />

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Java" />

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Android" />

        <android.support.design.widget.TabItem
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Php" />
    </android.support.design.widget.TabLayout>

    <FrameLayout
        android:id="@+id/frameLayout"
        android:layout_width="match_parent"
        android:layout_height="455dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tabLayout">

    </FrameLayout>
</android.support.constraint.ConstraintLayout>
```

File: build.gradle

Now gave the dependency library of TabLayout in build.gradle file.

1. implementation 'com.android.support:design:26.1.0'

File: MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    TabLayout tabLayout;
    FrameLayout frameLayout;
    Fragment fragment = null;
    FragmentManager fragmentManager;
    FragmentTransaction fragmentTransaction;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tabLayout=(TabLayout)findViewById(R.id.tabLayout);
        frameLayout=(FrameLayout)findViewById(R.id.frameLayout);

        fragment = new HomeFragment();
        fragmentManager = getSupportFragmentManager();
        fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.frameLayout, fragment);
        fragmentTransaction.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OP
EN);
        fragmentTransaction.commit();

        tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
            @Override
            public void onTabSelected(TabLayout.Tab tab) {
                // Fragment fragment = null;
                switch (tab.getPosition()) {
                    case 0:
                        fragment = new HomeFragment();
                        break;
                    case 1:
                        fragment = new JavaFragment();
                        break;
                    case 2:
                        fragment = new AndroidFragment();
                        break;
                    case 3:
                        fragment = new PhpFragment();
                        break;
                }
                FragmentManager fm = getSupportFragmentManager();
                FragmentTransaction ft = fm.beginTransaction();
                ft.replace(R.id.frameLayout, fragment);
                 ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
                ft.commit();
            }

            @Override
            public void onTabUnselected(TabLayout.Tab tab) {

            }

            @Override
             public void onTabReselected(TabLayout.Tab tab) {

             }
        });

    }
}
```

Now create different fragment files for all different tabs.

File: HomeFragment.java

```
public class HomeFragment extends Fragment {

    public HomeFragment() {
        // Required empty public constructor
    }

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

}
```

File: fragment_home.xml

```
<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="tablayout.example.com.tablayoutwithframelayout.HomeFragment">


    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/home_fragment" />

</FrameLayout>
```

File: JavaFragment.java

```
public class JavaFragment extends Fragment {

    public JavaFragment() {
        // Required empty public constructor
    }

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

File: fragment_java.xml

```
<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="tablayout.example.com.tablayoutwithframelayout.JavaFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/java_fragment" />

</FrameLayout>
```

File: AndroidFragment.java

```
public class AndroidFragment extends Fragment {

    public AndroidFragment() {
        // Required empty public constructor
    }

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

}
```

File: fragment_android.xml

```
<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="tablayout.example.com.tablayoutwithframelayout.AndroidFragment"
>
```

```
    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/android_fragment" />
</FrameLayout>
```

File: PhpFragment.java

```
public class PhpFragment extends Fragment {

    public PhpFragment() {
        // Required empty public constructor
    }

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

}
```

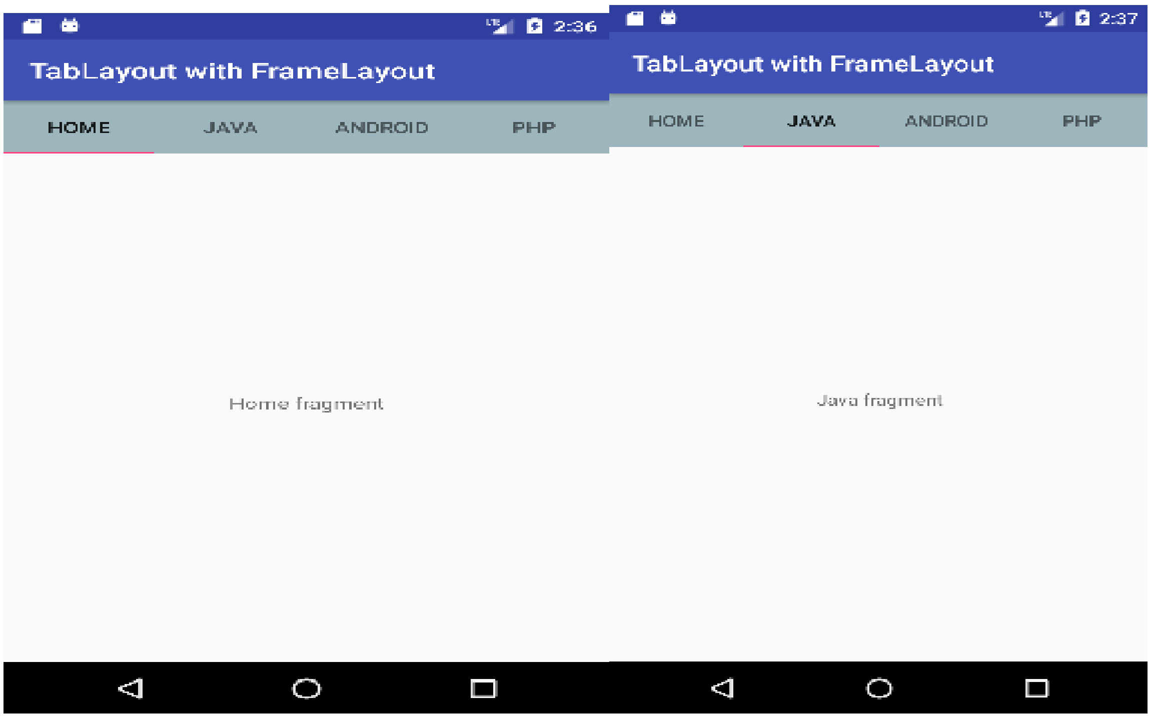
File: fragment_php.xml

```
<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="tablayout.example.com.tablayoutwithframelayout.PhpFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="@string/php_fragment" />

</FrameLayout>
```

File: strings.xml

```
<resources>
    <string name="app_name">TabLayout with FrameLayout</string>

    <!-- TODO: Remove or change this placeholder text -->
    <string name="home_fragment">Home fragment</string>
    <string name="java_fragment">Java fragment</string>
    <string name="android_fragment">Android fragment</string>
    <string name="php_fragment">Php fragment</string>
</resources>
```

Output


2:36
TabLayout with FrameLayout
HOME
JAVA
ANDROID
PHP
Home fragment
2:37
TabLayout with FrameLayout
HOME
JAVA
ANDROID
PHP
Java fragment