Course: 405-02: Mobile Application Development – 2

Unit-1: Project structure of Mobile Application:

1.1 Internal details of Android Application:

1.1.1 Dalvik VM, Screen Orientation

Dalvik Virtual Machine | DVM

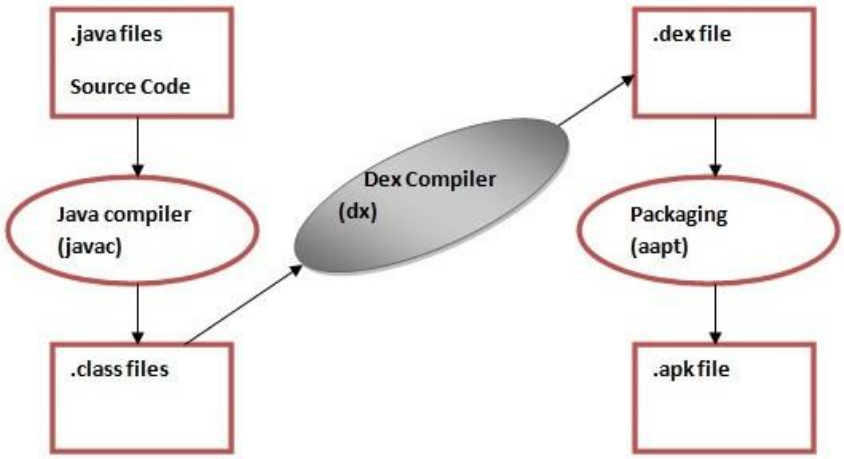
As we know the modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.

The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for *memory*, *battery life* and *performance*.

Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.

The **Dex compiler** converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.

Let's see the compiling and packaging process from the source file:



The **javac tool** compiles the java source file into the class file.

The **dx tool** takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.

The **Android Assets Packaging Tool (aapt)** handles the packaging process.

Screen Orientation

The **screenOrientation** is the attribute of activity element. The orientation of android activity can be portrait, landscape, sensor, unspecified etc. You need to define it in the AndroidManifest.xml file.

Syntax:

```
<activity android:name="package_name.Your_ActivityName"
    android:screenOrientation="orirntation_type">
</activity>
```

Example:

```
<activity android:name=" example.javatpoint.com.screenorientation.MainActivity"
    android:screenOrientation="portrait">
</activity>
```

```
<activity android:name=".SecondActivity"
    android:screenOrientation="landscape">
</activity>
```

The common values for screenOrientation attribute are as follows:

Value	Description
unspecified	It is the default value. In such case, system chooses the orientation.
portrait	taller not wider
landscape	wider not taller
sensor	orientation is determined by the device orientation sensor.

Android Portrait and Landscape mode screen orientation example

In this example, we will create two activities of different screen orientation. The first activity (MainActivity) will be as "portrait" orientation and second activity (SecondActivity) as "landscape" orientation type.

activity_main.xml

File: activity_main.xml

```
<?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="example.com.screenorientation.MainActivity">



    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginTop="112dp"
        android:onClick="onClick"
        android:text="Launch next activity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.612"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editText1"
        app:layout_constraintVertical_bias="0.613" />


    <TextView
```

```
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="124dp"
        android:ems="10"
        android:textSize="22dp"
        android:text="This activity is portrait orientation"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Activity class

File: MainActivity.java

```
package example.com.screenorientation;

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

public class MainActivity extends AppCompatActivity {
```

```
    Button button1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button1=(Button)findViewById(R.id.button1);
    }
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this,SecondActivity.class);
        startActivity(intent);
    }
}
```

activity_second.xml

File: activity_second.xml

```
<?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="example.com.screenorientation.SecondActivity">

    <TextView
        android:id="@+id/textView"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="180dp"
        android:text="this is landscape orientation"
        android:textSize="22dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

SecondActivity class

File: SecondActivity.java

```
package example.com.screenorientation;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class SecondActivity extends AppCompatActivity {

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

```
    }
}
```

AndroidManifest.xml

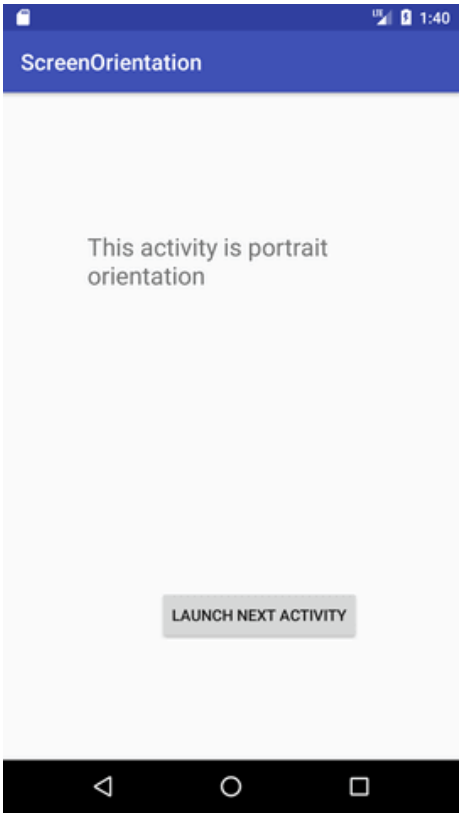
File: AndroidManifest.xml

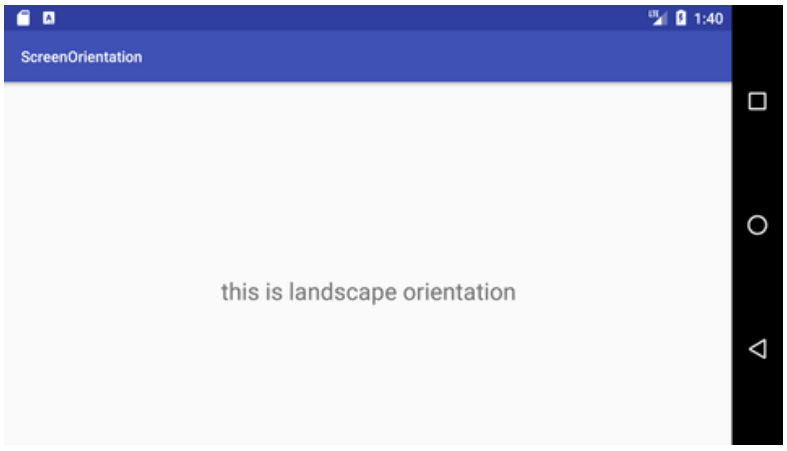
In AndroidManifest.xml file add the screenOrientation attribute in activity and provides its orientation. In this example, we provide "portrait" orientation for MainActivity and "landscape" for SecondActivity.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name="example.com.screenorientation.MainActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

```
        <activity android:name=".SecondActivity"
            android:screenOrientation="landscape">
        </activity>
    </application>

</manifest>
```

Output:





1.1.2 AndroidManifest, R.java

AndroidManifest File

The **AndroidManifest.xml file** *contains information of your package*, including components of the application such as activities, services, broadcast receivers, content providers etc.

It performs some other tasks also:

- It is **responsible to protect the application** to access any protected parts by providing the permissions.
- It also **declares the android api** that the application is going to use.
- It **lists the instrumentation classes**. The instrumentation classes provides profiling and other information. These information are removed just before the application is published etc.

This is the required xml file for all the android application and located inside the root directory.

A simple AndroidManifest.xml file looks like this:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.hello"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

```
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Elements of the AndroidManifest.xml file

The elements used in the above xml file are described below.

<manifest>

manifest is the root element of the AndroidManifest.xml file. It has **package** attribute that describes the package name of the activity class.

<application>

application is the subelement of the manifest. It includes the namespace declaration. This element contains several subelements that declares the application component such as activity etc.

The commonly used attributes are of this element are **icon**, **label**, **theme** etc.

android:icon represents the icon for all the android application components.

android:label works as the default label for all the application components.

android:theme represents a common theme for all the android activities.

<activity>

activity is the subelement of application and represents an activity that must be defined in the AndroidManifest.xml file. It has many attributes such as label, name, theme, launchMode etc.

android:label represents a label i.e. displayed on the screen.

android:name represents a name for the activity class. It is required attribute.

<intent-filter>

intent-filter is the sub-element of activity that describes the type of intent to which activity, service or broadcast receiver can respond to.

<action>

It adds an action for the intent-filter. The intent-filter must have at least one action element.

<category>

It adds a category name to an intent-filter.

Android R.java file

Android R.java is *an auto-generated file by aapt* (Android Asset Packaging Tool) that contains resource IDs for all the resources of res/ directory.

If you create any component in the activity_main.xml file, id for the corresponding component is automatically created in this file. This id can be used in the activity source file to perform any action on the component.

Note: If you delete R.jar file, android creates it automatically.

Let's see the android R.java file. It includes a lot of static nested classes such as menu, id, layout, attr, drawable, string etc.

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package com.example.helloandroid;
```

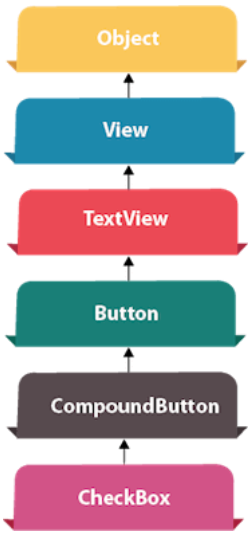
```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int menu_settings=0x7f070000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int activity_main=0x7f060000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int hello_world=0x7f040001;
        public static final int menu_settings=0x7f040002;
    }
    public static final class style {
        /**
        Base application theme, dependent on API level. This theme is replaced
        by AppBaseTheme from res/values-vXX/styles.xml on newer devices.

            Theme customizations available in newer API levels can go in
            res/values-vXX/styles.xml, while customizations related to
            backward-compatibility can go here.

        Base application theme for API 11+. This theme completely replaces
        AppBaseTheme from res/values/styles.xml on API 11+ devices.
```

```
API 11 theme customizations can go here.

        Base application theme for API 14+. This theme completely replaces
        AppBaseTheme from BOTH res/values/styles.xml and
        res/values-v11/styles.xml on API 14+ devices.

    API 14 theme customizations can go here.
         */
        public static final int AppBaseTheme=0x7f050000;
        /**  Application theme.
    All customizations that are NOT specific to a particular API-level can go here.
         */
        public static final int AppTheme=0x7f050001;
    }
}
```

1.2 Android Widgets (UI)

1.2.1 Default and Custom Checkbox

Android CheckBox is a type of two state button either checked or unchecked.



There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

Android CheckBox class is the subclass of CompoundButton class.

Android CheckBox class

The android.widget.CheckBox class provides the facility of creating the CheckBoxes.

Methods of CheckBox class

There are many inherited methods of View, TextView, and Button classes in the CheckBox class. Some of them are as follows:

Method	Description
public boolean isChecked()	Returns true if it is checked otherwise false.
public void setChecked(boolean status)	Changes the state of the CheckBox.

Android CheckBox Example

activity_main.xml

Drag the three checkboxes and one button for the layout. Now the activity_main.xml file will look like this:

File: activity_main.xml

```
<?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="example.javatpoint.com.checkbox.MainActivity">


    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="68dp"
        android:text="Pizza"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="28dp"
        android:text="Coffee"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox" />

    <CheckBox
        android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
```

```
        android:layout_marginTop="28dp"
        android:text="Burger"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox2" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="144dp"
        android:layout_marginTop="184dp"
        android:text="Order"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/checkBox3" />

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

Activity class

Let's write the code to check which toggle button is ON/OFF.

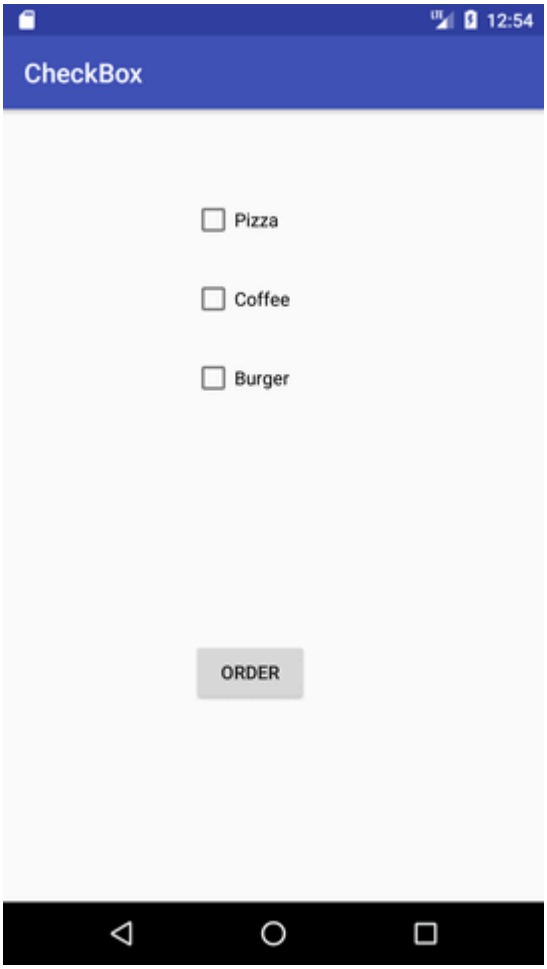
File: MainActivity.java

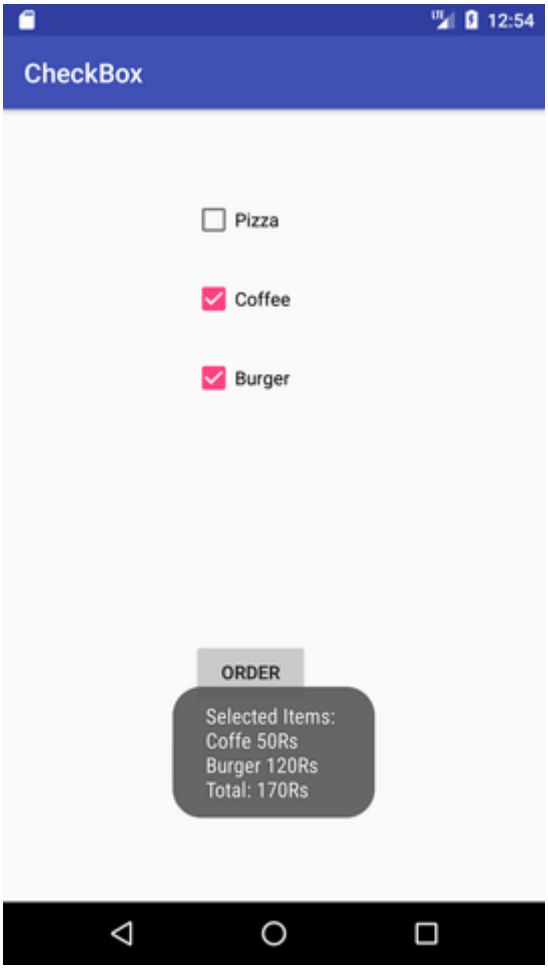
```
package example.com.checkbox;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    CheckBox pizza,coffe,burger;
    Button buttonOrder;
    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    addListenerOnButtonClick();
}
public void addListenerOnButtonClick(){
    //Getting instance of CheckBoxes and Button from the activty_main.xml file
    pizza=(CheckBox)findViewById(R.id.checkBox);
    coffe=(CheckBox)findViewById(R.id.checkBox2);
    burger=(CheckBox)findViewById(R.id.checkBox3);
    buttonOrder=(Button)findViewById(R.id.button);

    //Applying the Listener on the Button click
    buttonOrder.setOnClickListener(new View.OnClickListener(){

        @Override
        public void onClick(View view) {
            int totalamount=0;
            StringBuilder result=new StringBuilder();
            result.append("Selected Items:");
            if(pizza.isChecked()){
                result.append("\nPizza 100Rs");
                totalamount+=100;
            }
            if(coffe.isChecked()){
                result.append("\nCoffe 50Rs");
                totalamount+=50;
            }
            if(burger.isChecked()){
                result.append("\nBurger 120Rs");
                totalamount+=120;
            }
            result.append("\nTotal: "+totalamount+"Rs");
            //Displaying the message on the toast
```

```
                Toast.makeText(getApplicationContext(), result.toString(), Toast.LENGTH_L
ONG).show();
            }

        });
    }
}
```

Output:





Android Custom CheckBox

Android provides facility to customize the UI of view elements rather than default.

You are able to create custom CheckBox in android. So, you can add some different images of checkbox on the layout.

Example of Custom CheckBox

In this example, we create both default as well as custom checkbox. Add the following code in activity_main.xml file.

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"
  tools:context="example.javatpoint.com.customcheckbox.MainActivity">


  <TextView
    android:id="@+id/textView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal"
    android:textSize="25dp"
    android:text="Default Check Box"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

  <CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="New CheckBox"
    android:id="@+id/checkBox"
    android:layout_below="@+id/textView1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="46dp" />

<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="New CheckBox"
    android:id="@+id/checkBox2"
    android:layout_below="@+id/checkBox"
    android:layout_alignLeft="@+id/checkBox"
    android:layout_alignStart="@+id/checkBox" />

<View
    android:layout_width="fill_parent"
    android:layout_height="1dp"
    android:layout_marginTop="200dp"
    android:background="#B8B894"
    android:id="@+id/viewStub" />

<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox 1"
    android:id="@+id/checkBox3"
    android:button="@drawable/customcheckbox"
    android:layout_below="@+id/viewStub"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="58dp" />

<CheckBox
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="CheckBox 2"
        android:id="@+id/checkBox4"
        android:button="@drawable/customcheckbox"
        android:layout_below="@+id/checkBox3"
        android:layout_alignLeft="@+id/checkBox3"
        android:layout_alignStart="@+id/checkBox3" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textSize="25dp"
        android:text="Custom Check Box"
        android:id="@+id/textView"
        android:layout_alignTop="@+id/viewStub"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Checked"
        android:id="@+id/button"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Now implement a selector in another file (checkbox.xml) under drawable folder which customizes the checkbox.

checkbox.xml

File: checkbox.xml

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

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:state_checked="true" android:drawable="@drawable/checked" />

   <item android:state_checked="false" android:drawable="@drawable/unchecked"/
>
</selector>
```

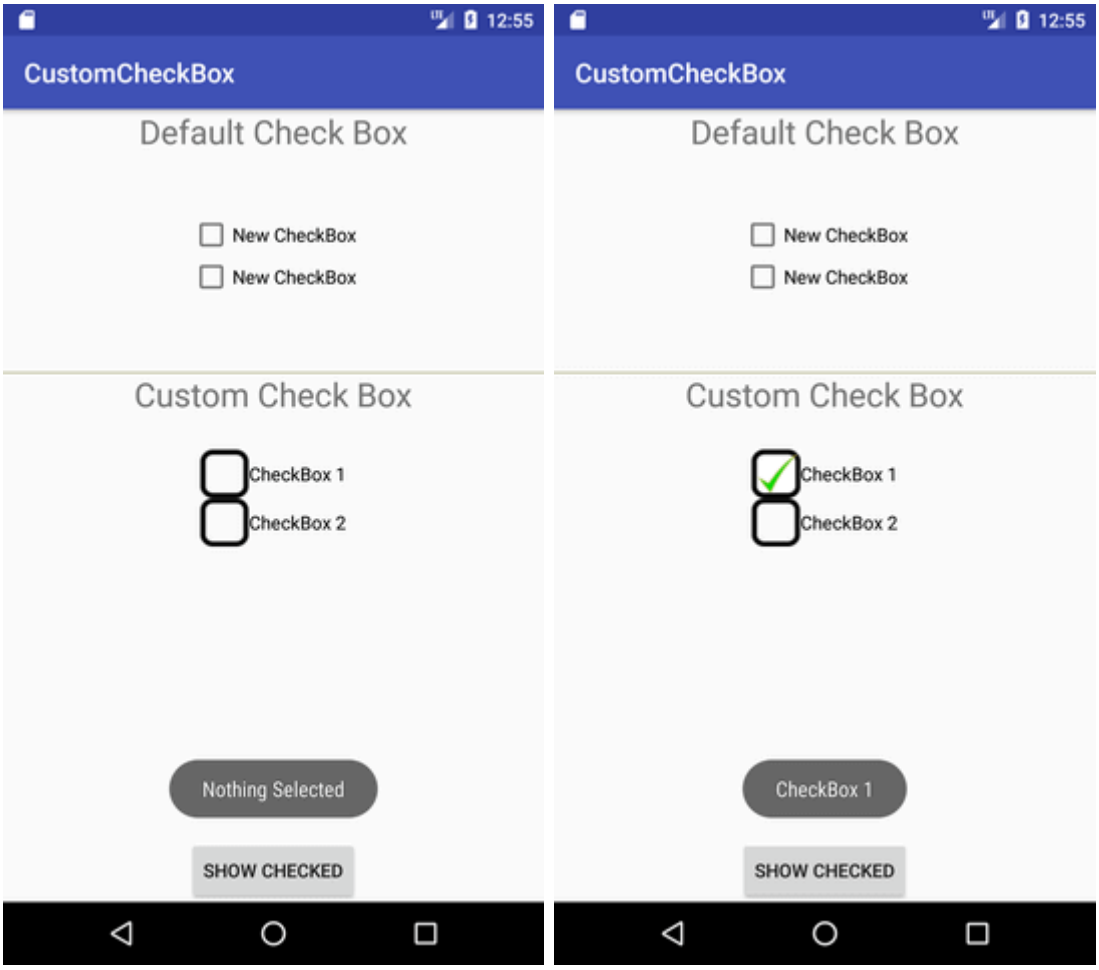
Activity class

File: MainActivity.java

```
package example.com.customcheckbox;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
   CheckBox cb1,cb2;
   Button button;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      cb1=(CheckBox)findViewById(R.id.checkBox3);
      cb2=(CheckBox)findViewById(R.id.checkBox4);
      button=(Button)findViewById(R.id.button);

      button.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            StringBuilder sb=new StringBuilder("");
```

```
            if(cb1.isChecked()){
                String s1=cb1.getText().toString();
                sb.append(s1);
            }

            if(cb2.isChecked()){
                String s2=cb2.getText().toString();
                sb.append("\n"+s2);

            }
            if(sb!=null && !sb.toString().equals("")){
                Toast.makeText(getApplicationContext(), sb, Toast.LENGTH_LONG).show
();

            }
            else{
                Toast.makeText(getApplicationContext(),"Nothing Selected", Toast.LENG
TH_LONG).show();
            }

        }

    });
  }
}
```

Output:


12:55
CustomCheckBox
Default Check Box
New CheckBox
New CheckBox
Custom Check Box
CheckBox 1
CheckBox 2
Nothing Selected
SHOW CHECKED
12:55
CustomCheckBox
Default Check Box
New CheckBox
New CheckBox
Custom Check Box
CheckBox 1
CheckBox 2
CheckBox 1
SHOW CHECKED

1.2.2 Dynamic and Custom RadioButton

Android RadioButton

RadioButton is a two states button which is either checked or unchecked. If a single radio button is unchecked, we can click it to make checked radio button. Once a radio button is checked, it cannot be marked as unchecked by user.

RadioButton is generally used with *RadioGroup*. RadioGroup contains several radio buttons, marking one radio button as checked makes all other radio buttons as unchecked.

Example of Radio Button

In this example, we are going to implement single radio button separately as well as radio button in **RadioGroup**.

activity_main.xml

File: 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"
  tools:context="example.com.radiobutton.MainActivity">

  <TextView
    android:id="@+id/textView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:gravity="center_horizontal"
    android:textSize="22dp"
    android:text="Single Radio Buttons" />
```

```
<!--   Default RadioButtons  -->

<RadioButton
   android:id="@+id/radioButton1"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_gravity="center_horizontal"
   android:text="Radio Button 1"
   android:layout_marginTop="20dp"

   android:textSize="20dp" />
<RadioButton
   android:id="@+id/radioButton2"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="Radio Button 2"
   android:layout_marginTop="10dp"

   android:textSize="20dp" />


<View
   android:layout_width="fill_parent"
   android:layout_height="1dp"
   android:layout_marginTop="20dp"
   android:background="#B8B894" />

<TextView
   android:id="@+id/textView2"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_marginTop="30dp"
   android:gravity="center_horizontal"
   android:textSize="22dp"
```

```
    android:text="Radio button inside RadioGroup" />


<!--   Customized RadioButtons  -->


<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/radioGroup">

    <RadioButton
        android:id="@+id/radioMale"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="  Male"
        android:layout_marginTop="10dp"
        android:checked="false"
        android:textSize="20dp" />

    <RadioButton
        android:id="@+id/radioFemale"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="   Female"
        android:layout_marginTop="20dp"
        android:checked="false"

        android:textSize="20dp" />
</RadioGroup>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:text="Show Selected"
        android:id="@+id/button"
        android:onClick="onclickbuttonMethod"
        android:layout_gravity="center_horizontal" />

    </LinearLayout>
```

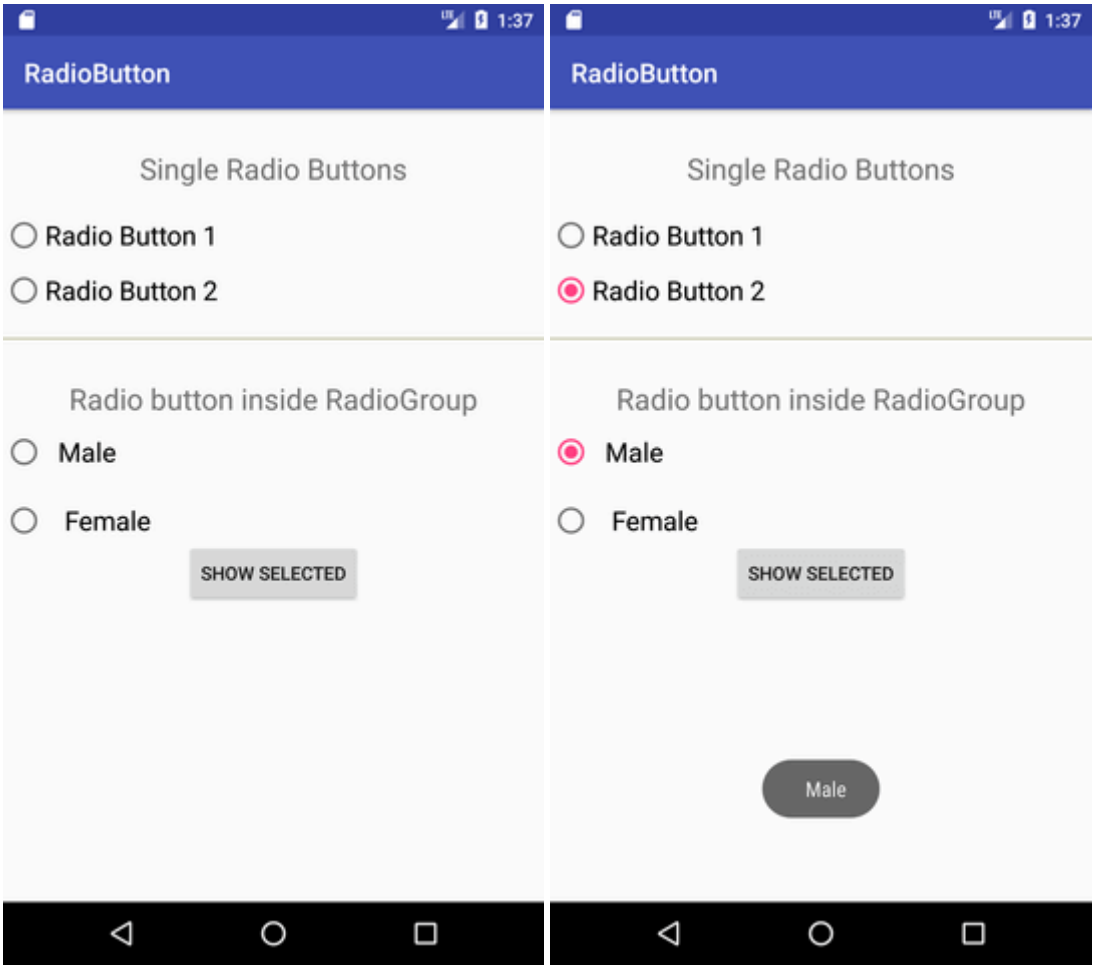
Activity class

File: MainActivity.java

```
    package example.com.radiobutton;

    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.Button;
    import android.widget.RadioButton;
    import android.widget.RadioGroup;
    import android.widget.Toast;

    public class MainActivity extends AppCompatActivity {
        Button button;
        RadioButton genderradioButton;
        RadioGroup radioGroup;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            radioGroup=(RadioGroup)findViewById(R.id.radioGroup);
        }
        public void onclickbuttonMethod(View v){
            int selectedId = radioGroup.getCheckedRadioButtonId();
            genderradioButton = (RadioButton) findViewById(selectedId);
            if(selectedId==-1){
```

```
            Toast.makeText(MainActivity.this,"Nothing selected", Toast.LENGTH_SHORT).
    show();
        }
        else{
            Toast.makeText(MainActivity.this,genderradioButton.getText(), Toast.LENGTH
    _SHORT).show();
        }

    }
}
```

Output:



Android Dynamic RadioButton

Instead of creating RadioButton through drag and drop from palette, android also facilitates you to create it programmatically (dynamically). For creating dynamic RadioButton, we need to use **android.view.ViewGroup.LayoutParams** which configures the width and height of views and implements *setOnCheckedChangeListener()* method of *RadioGroup* class.

Example of Dynamic RadioButton

Let's see an example of Dynamic RadioButton.

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: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"
   android:id="@+id/relativeLayout"
   tools:context="com.example.test.dynamicradiobutton.MainActivity">


</RelativeLayout>
```

Activity class

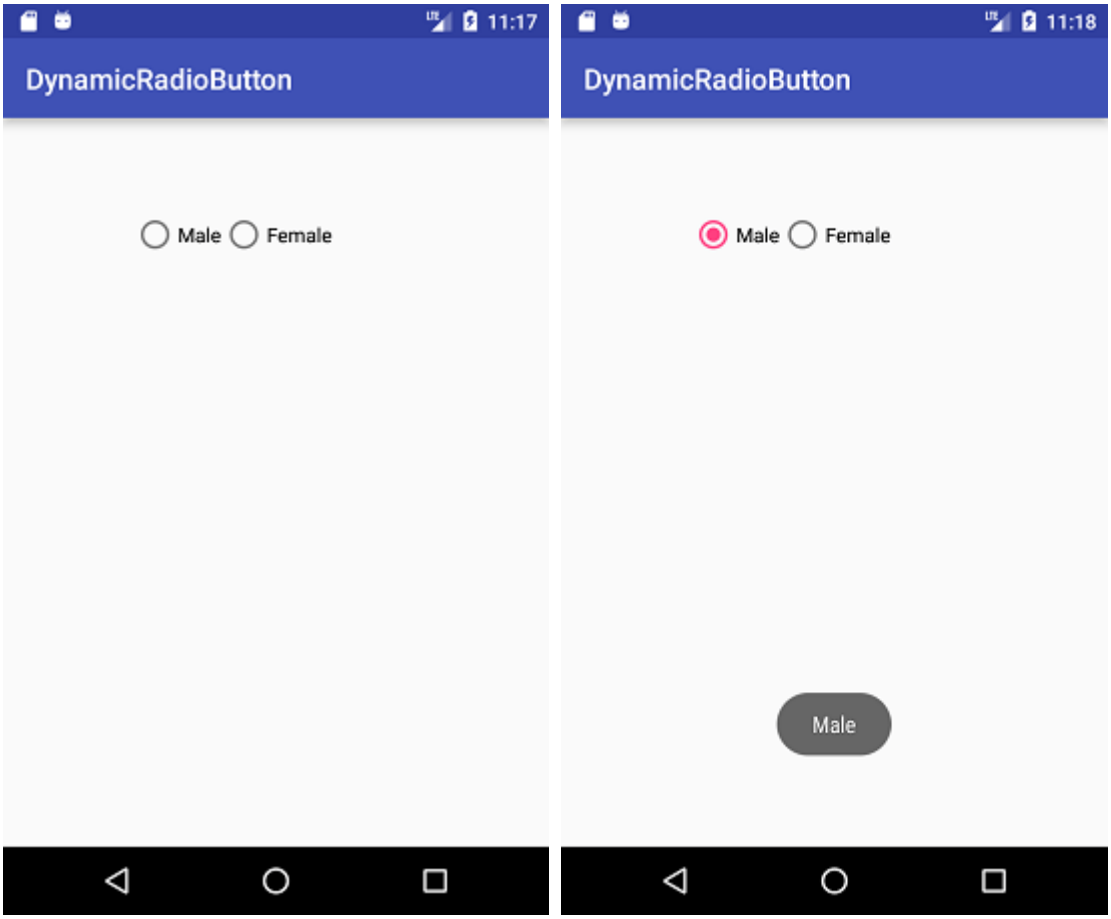
File: MainActivity.java

```
package com.example.test.dynamicradiobutton;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.RadioButton;
import android.widget.RadioGroup;
```

```
import android.widget.RelativeLayout;

import android.widget.RelativeLayout.LayoutParams;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    RadioGroup rg;
    RelativeLayout rl;
    RadioButton rb1,rb2;

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

        rg = new RadioGroup(this);
        rl = (RelativeLayout) findViewById(R.id.relativeLayout);
        rb1 = new RadioButton(this);
        rb2 = new RadioButton(this);

        rb1.setText("Male");
        rb2.setText("Female");
        rg.addView(rb1);
        rg.addView(rb2);
        rg.setOrientation(RadioGroup.HORIZONTAL);

        RelativeLayout.LayoutParams params = new RelativeLayout.LayoutParams((int)
LayoutParams.WRAP_CONTENT,(int)LayoutParams.WRAP_CONTENT);
        params.leftMargin =150;
        params.topMargin = 100;

        rg.setLayoutParams(params);
        rl.addView(rg);
```

```
        rg.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
{
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                RadioButton radioButton = (RadioButton) findViewById(checkedId);
                Toast.makeText(getApplicationContext(),radioButton.getText(),Toast.LENGT
H_LONG).show();
            }
        });
    }
}
```

Output:



Android Custom RadioButton

Rather than default user interface of android RadioButton, we can also implement a custom radio button. Custom RadioButton makes user interface more attractive.

Example of Custom RadioButton

Let's see an example of custom RadioButton.

activity_main.xml

File: 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:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:orientation="vertical"
   tools:context="com.example.test.customradiobutton.MainActivity">



   <TextView
      android:id="@+id/tv"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_marginTop="30dp"
      android:gravity="center_horizontal"
      android:textSize="25dp"
      android:text="Customized Radio Buttons" />
```

```
<!--   Customized RadioButtons  -->


<RadioGroup
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/radioGroup">

   <RadioButton
      android:id="@+id/radioMale"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="  Male"
      android:layout_marginTop="10dp"
      android:checked="false"
      android:button="@drawable/custom_radio_button"
      android:textSize="20dp" />

   <RadioButton
      android:id="@+id/radioFemale"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="   Female"
      android:layout_marginTop="20dp"
      android:checked="false"
      android:button="@drawable/custom_radio_button"
      android:textSize="20dp" />
</RadioGroup>

<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Show Selected"
   android:id="@+id/button"
```

```
        android:onClick="onclickbuttonMethod"
        android:layout_gravity="center_horizontal" />

    </LinearLayout>
```

custom_radio_button.xml

Now implement a selector in another file (custom_radio_button.xml) in drawable and place two different checked and unchecked button images.

File: checkbox.xml

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

    <selector xmlns:android="http://schemas.android.com/apk/res/android">

      <item android:state_checked="true" android:drawable="@drawable/checkedradi
    obutton" />
      <item android:state_checked="false" android:drawable="@drawable/unchekedra
    diobutton" />

    </selector>
```

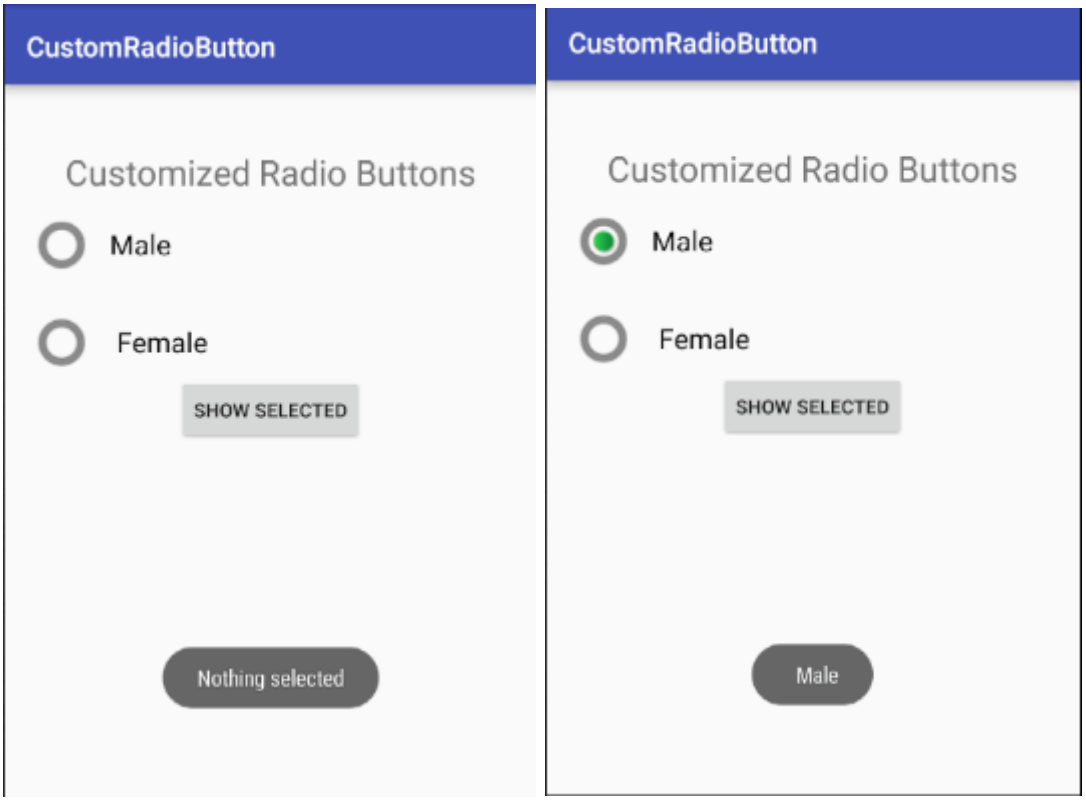
Activity class

File: MainActivity.java

```
    package com.example.test.customradiobutton;

    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.Button;
    import android.widget.RadioButton;
    import android.widget.RadioGroup;
    import android.widget.Toast;

    public class MainActivity extends AppCompatActivity {
```

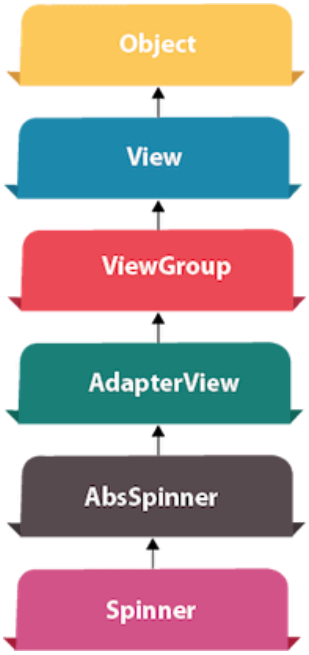
```
    Button button;
    RadioButton genderradioButton;
    RadioGroup radioGroup;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        radioGroup=(RadioGroup)findViewById(R.id.radioGroup);
    }


    public void onclickbuttonMethod(View v){
        int selectedId = radioGroup.getCheckedRadioButtonId();
        genderradioButton = (RadioButton) findViewById(selectedId);
        if(selectedId==-1){
            Toast.makeText(MainActivity.this,"Nothing selected", Toast.LENGTH_SHORT).
show();
        }
        else{
            Toast.makeText(MainActivity.this,genderradioButton.getText(), Toast.LENGTH
_SHORT).show();
        }

    }
}
```

Output:


CustomRadioButton
Customized Radio Buttons
Male
Female
SHOW SELECTED
Nothing selected
CustomRadioButton
Customized Radio Buttons
Male
Female
SHOW SELECTED
Male

1.2.3 Spinner, AlterDialog

Android Spinner



Android Spinner is like the combox box of AWT or Swing. It can be used to display the multiple options to the user in which only one item can be selected by the user.

Android spinner is like the drop down menu with multiple values from which the end user can select only one value.

Android spinner is associated with AdapterView. So you need to use one of the adapter classes with spinner.

Android Spinner class is the subclass of AsbSpinner class.

Android Spinner Example

In this example, we are going to display the country list. You need to use **ArrayAdapter** class to store the country list.

Let's see the simple example of spinner in android.

activity_main.xml

Drag the Spinner from the pallete, now the activity_main.xml file will like this:

File: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.an
droid.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="example.javatpoint.com.spinner.MainActivity">

    <Spinner
```

```
        android:id="@+id/spinner"
        android:layout_width="149dp"
        android:layout_height="40dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.498" />

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

Activity class

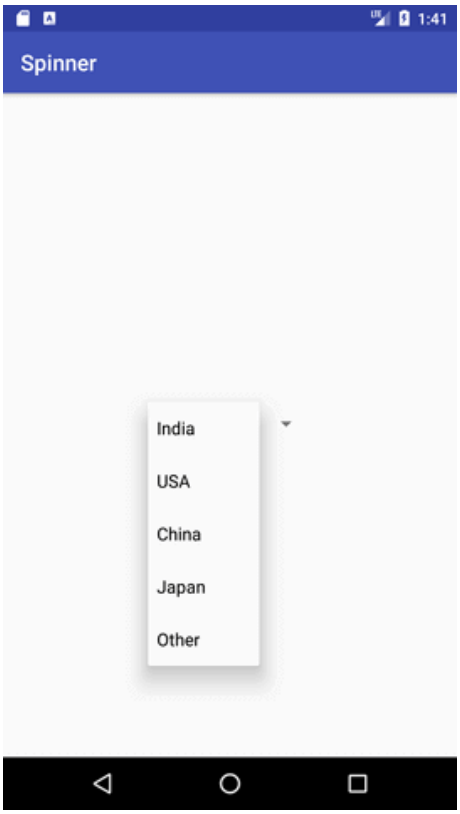
Let's write the code to display item on the spinner and perform event handling.

File: MainActivity.java

```
    package example.javatpoint.com.spinner;

    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.AdapterView;
    import android.widget.ArrayAdapter;
    import android.widget.Spinner;
    import android.widget.Toast;

    public class MainActivity extends AppCompatActivity implements
            AdapterView.OnItemSelectedListener {
        String[] country = { "India", "USA", "China", "Japan", "Other"};
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       //Getting the instance of Spinner and applying OnItemSelectedListener on it
        Spinner spin = (Spinner) findViewById(R.id.spinner);
        spin.setOnItemSelectedListener(this);

        //Creating the ArrayAdapter instance having the country list
        ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,
country);
        aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item
);
        //Setting the ArrayAdapter data on the Spinner
        spin.setAdapter(aa);

    }

    //Performing action onItemSelected and onNothing selected
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long i
d) {
        Toast.makeText(getApplicationContext(),country[position] , Toast.LENGTH_LON
G).show();
    }
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO Auto-generated method stub
    }
}
```


1:41
Spinner
India


1:41
Spinner
India
USA
China
Japan
Other


1:42
Spinner
USA
USA

Android AlertDialog

Android AlertDialog can be used to display the dialog message with OK and Cancel buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.

Android AlertDialog is composed of three regions: title, content area and action buttons.

Android AlertDialog is the subclass of Dialog class.

Methods of AlertDialog class

Method	Description
public AlertDialog.Builder setTitle(CharSequence)	This method is used to set the title of AlertDialog.
public AlertDialog.Builder setMessage(CharSequence)	This method is used to set the message for AlertDialog.
public AlertDialog.Builder setIcon(int)	This method is used to set the icon over AlertDialog.

Android AlertDialog Example

Let's see a simple example of android alert dialog.

activity_main.xml

You can have multiple components, here we are having only a textview.

File: activity_main.xml

```
<?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="example.javatpoint.com.alertdialog.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="Close app"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

strings.xml

Optionally, you can store the dialog message and title in the strings.xml file.

File: strings.xml

```
<resources>
    <string name="app_name">AlertDialog</string>
    <string name="dialog_message">Welcome to Alert Dialog</string>
    <string name="dialog_title">Javatpoint Alert Dialog</string>
</resources>
```

Activity class

Let's write the code to create and show the AlertDialog.

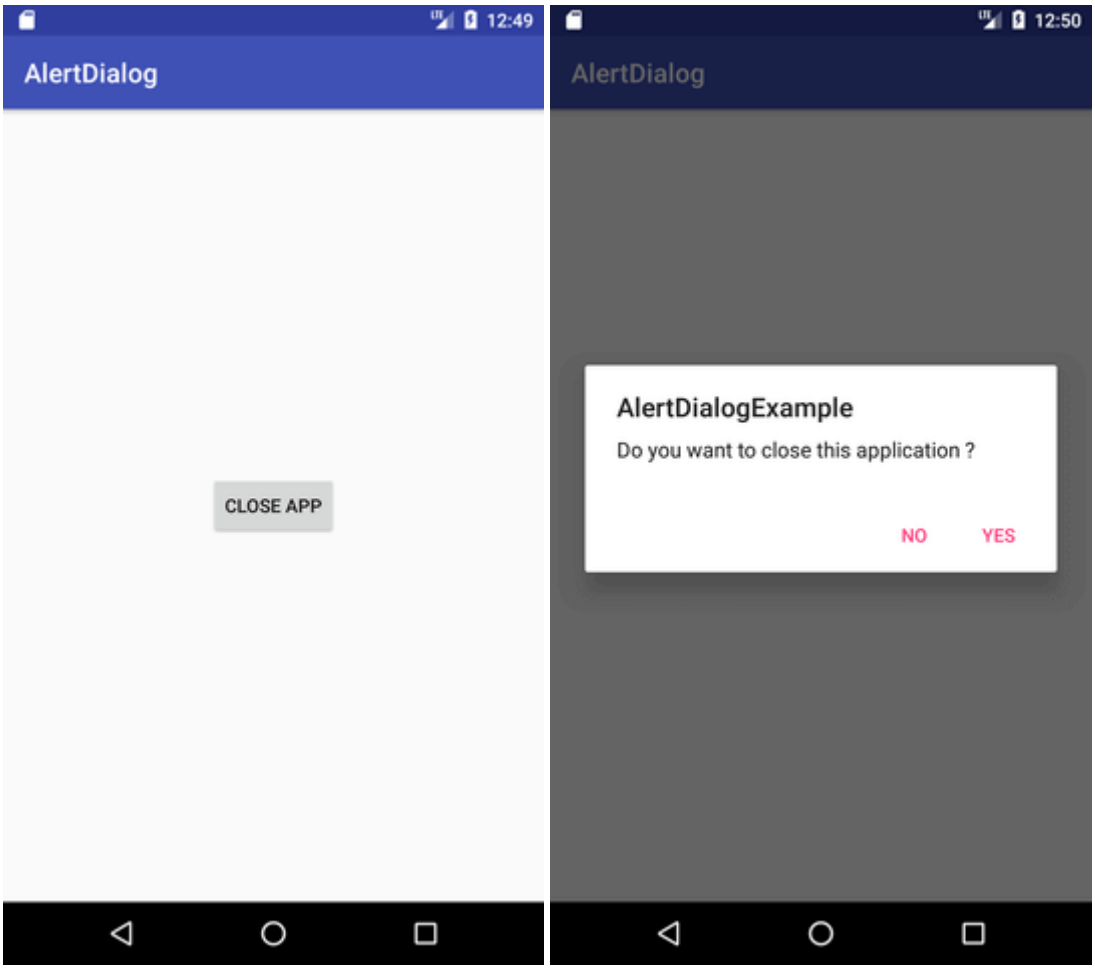
File: MainActivity.java

```
package example.com.alertdialog;

import android.content.DialogInterface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.Button;
import android.app.AlertDialog;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    Button closeButton;
    AlertDialog.Builder builder;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        closeButton = (Button) findViewById(R.id.button);
        builder = new AlertDialog.Builder(this);
        closeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                //Uncomment the below code to Set the message and title from the string
s.xml file
                builder.setMessage(R.string.dialog_message) .setTitle(R.string.dialog_title);

                //Setting message manually and performing action on button click
                builder.setMessage("Do you want to close this application ?")
                        .setCancelable(false)
                        .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                finish();
                                Toast.makeText(getApplicationContext(),"you choose yes action f
or alertbox",
                                Toast.LENGTH_SHORT).show();
                            }
                        })
                        .setNegativeButton("No", new DialogInterface.OnClickListener() {
```

```
                    public void onClick(DialogInterface dialog, int id) {
                        //  Action for 'NO' Button
                        dialog.cancel();
                        Toast.makeText(getApplicationContext(),"you choose no action f
or alertbox",
                        Toast.LENGTH_SHORT).show();
                    }
                });
            //Creating dialog box
            AlertDialog alert = builder.create();
            //Setting the title manually
            alert.setTitle("AlertDialogExample");
            alert.show();
        }
    });
  }
}
```

Output:


12:49
AlertDialog
CLOSE APP
12:50
AlertDialog
AlertDialogExample
Do you want to close this application ?
NO
YES