Experiment No 2.2

Create an Android App using various controls such TexEdit, CheckBox, RadioButton, RadioGroup, etc.

<https://www.geeksforgeeks.org/how-to-use-checkbox-in-android/>

**Create an Android App using CheckBox.**

CheckBox belongs to android.widget.CheckBox class. Android CheckBox class is the subclass of CompoundButton class. It is generally used in a place where user can select one or more than choices from a given list of choices. For example, selecting hobbies.

*public class* CheckBox *extends* CompoundButton

**class Hierarchy :**

java.lang.Object

↳ android.view.View

↳ android.widget.TextView

↳ android.widget.Button

↳ android.widget.CompoundButton

↳ android.widget.CheckBox

it has two states – **checked** or **unchecked**.

**Methods of CheckBox class**

* ***public boolean isChecked():*** If CheckBox is in checked state then return true otherwise false.
* ***public void setChecked(boolean status):*** It changes the state of the CheckBox.

Below is the code for an example where the user chooses its hobbies from the given list containing Painting, Reading, Singing and Cooking with the help of CheckBox.

**MainActivity.java**

/Below is the code for MainActivity.java

package com.geeksforgeeks.gfg.checkbox;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.CheckBox;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    CheckBox ch, ch1, ch2, ch3;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        // Binding MainActivity.java with activity\_main.xml file

        setContentView(R.layout.activity\_main);

        // Finding CheckBox by its unique ID

        ch=(CheckBox)findViewById(R.id.checkBox);

        ch1=(CheckBox)findViewById(R.id.checkBox2);

        ch2=(CheckBox)findViewById(R.id.checkBox3);

        ch3=(CheckBox)findViewById(R.id.checkBox4);

    }

    // This function is invoked when the button is pressed.

    public void Check(View v)

    {

        String msg="";

        // Concatenation of the checked options in if

        // isChecked() is used to check whether

        // the CheckBox is in true state or not.

        if(ch.isChecked())

            msg = msg + " Painting ";

        if(ch1.isChecked())

            msg = msg + " Reading ";

        if(ch2.isChecked())

            msg = msg + " Singing ";

        if(ch3.isChecked())

            msg = msg + " Cooking ";

        // Toast is created to display the

        // message using show() method.

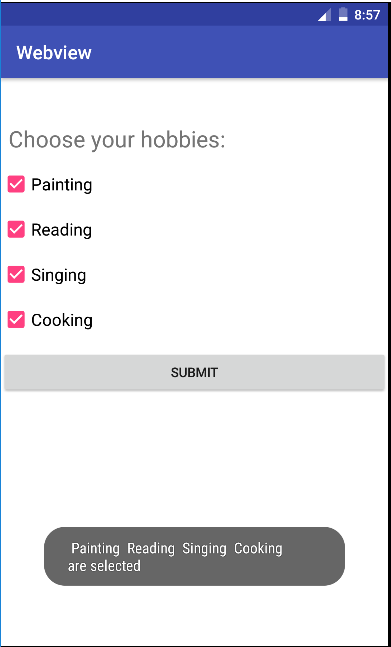
        Toast.makeText(this, msg + "are selected",

                       Toast.LENGTH\_LONG).show();

    }

}

**Output:**



**Creating the Application Choosing Options RadioButton**

<https://www.geeksforgeeks.org/android-how-to-add-radio-buttons-in-an-android-application/>

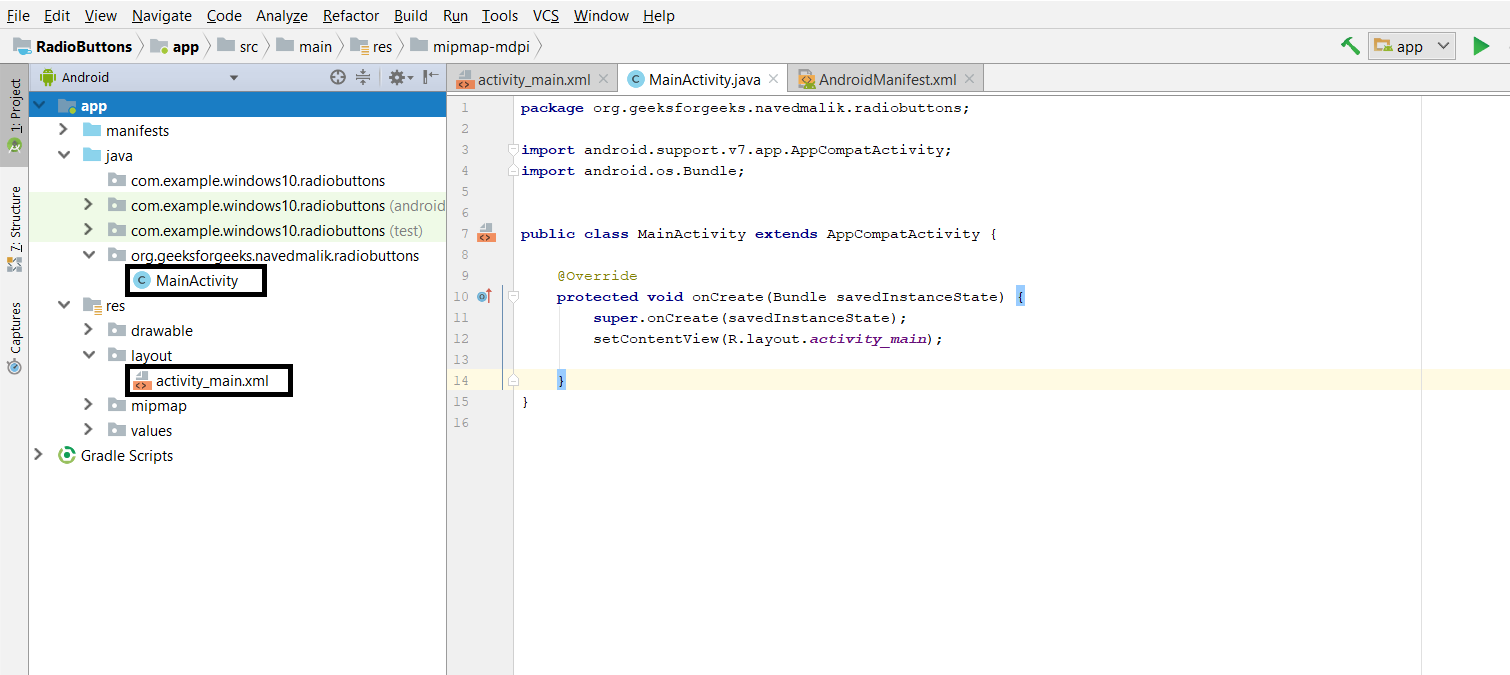
[Android](https://www.geeksforgeeks.org/tag/android/) **radio button** is a widget that can have more than one option to choose from. The user can choose only one option at a time. Each option here refers to a radio button and all the options for the topic are together referred to as Radio Group. Hence, Radio Buttons are used inside a RadioGroup.

**Pre-requisites:**

* [Android App Development Fundamentals for Beginners](https://www.geeksforgeeks.org/android-app-development-fundamentals-for-beginners/)
* [Guide to Install and Set up Android Studio](https://www.geeksforgeeks.org/guide-to-install-and-set-up-android-studio/)
* [Android | Starting with the first app/android project](https://www.geeksforgeeks.org/android-starting-with-first-app-android-project/)
* [Android | Running your first Android app](https://www.geeksforgeeks.org/android-running-your-first-android-app/)

This example will help in developing an Android App that creates Radio Buttons according to the above example:

* **Step 1:** First create a new Android Application. This will create an XML file “activity\_main.xml” and a Java File “MainActivity.Java”. Please refer the pre-requisites to learn more about this step.



**Step 2:** Open the “activity\_main.xml” file and add the following widgets in a [Relative Layout](https://www.geeksforgeeks.org/layouts-android-ui-design/):

* A **TextView** to display the question message
* A **RadioGroup** to hold the option Radio Buttons which are the possible answers
* **4 RadioButtons** to hold an answer each.
* A Submit and a Clear button to store the response.

Also, Assign the **ID** to each of the components along with other attributes as shown in the given image and the code below. The assigned ID on a component helps that component to be easily found and used in the Java files.

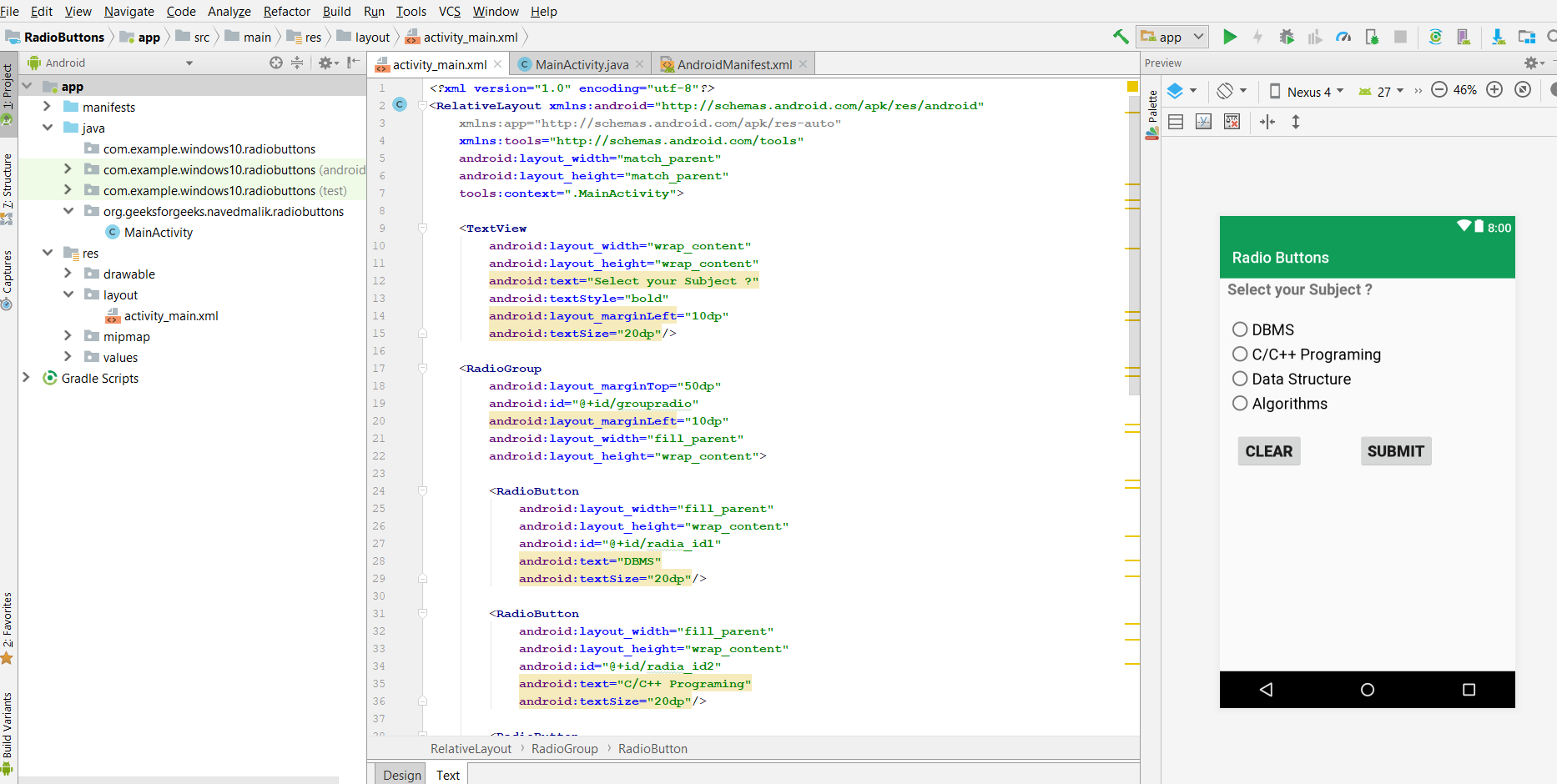
**Syntax:**

android:id="@+id/id\_name"

Here the given IDs are as follows:

* RadioGroup: groupradio
* RadioButton1: radia\_id1
* RadioButton2: radia\_id2
* RadioButton3: radia\_id3
* RadioButton4: radia\_id4
* Submit Button: submit
* Clear Button: clear

This will make the UI of the Application.



**Step 3:** Now, after the UI, this step will create the Backend of Application. For this, open the “MainActivity.java” file and instantiate the components made in the XML file (RadioGroup, TextView, Clear, and Submit Button) using findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

**Syntax:** General

ComponentType object = (ComponentType)findViewById(R.id.IdOfTheComponent);

**Syntax:**Components used

Button submit = (Button)findViewById(R.id.submit);

Button clear = (Button)findViewById(R.id.clear);

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

**Step 4:** This step involves setting up the operations on the RadioGroup, RadioButtons, and the Submit and Clear Buttons.

 These operations are as follows:

* Unset all the Radio Buttons initially as the default value. This is done by the following command:

radioGroup.clearCheck();

* Add the Listener on the RadioGroup. This will help to know whenever the user clicks on any Radio Button, and the further operation will be performed. The listener can be added as follows:

***radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){}***

* Define the operations to be done when a radio button is clicked. This involves getting the specific radio button that has been clicked, using its id. Then this radio button gets set and the rest of the radio button is reset.
* Add the listener on Submit button and clear button. This will be used to check when the user clicks on the button. This is done as follows:

***submit.setOnClickListener(new View.OnClickListener() {}******clear.setOnClickListener(new View.OnClickListener() {}***

* In the Submit Button Listener, set the operations to be performed. This involves displaying the marked answer in the form of **Toast**.
* In the Clear Button Listener, set the operations to be performed. This involves resetting all the radio buttons.

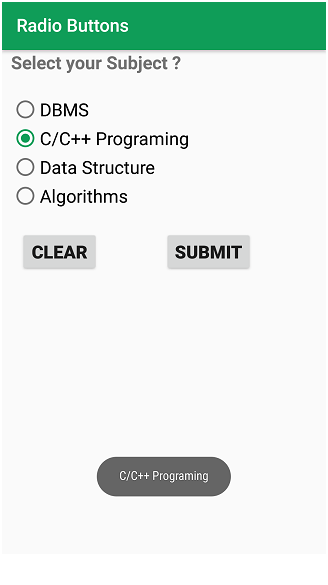
**Step5:** Now run the app and operate as follows:

* When the app is opened, it displays a question with 4 answers and a clear and submit button.
* When any answer is clicked, that radio button gets set.
* Clicking on any other radio button sets that one and resets the others.
* Clicking on Submit button displays the currently marked answer as a Toast.
* Clicking on Clear button resets all the radio buttons to their default state.

**MainActivity.Java**

* Java

|  |
| --- |
| package org.geeksforgeeks.navedmalik.radiobuttons;    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 {        // Define the object for Radio Group,      // Submit and Clear buttons      private RadioGroup radioGroup;      Button submit, clear;        @Override      protected void onCreate(Bundle savedInstanceState)      {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);            // Bind the components to their respective objects          // by assigning their IDs          // with the help of findViewById() method          submit = (Button)findViewById(R.id.submit);          clear = (Button)findViewById(R.id.clear);          radioGroup = (RadioGroup)findViewById(R.id.groupradio);            // Uncheck or reset the radio buttons initially          radioGroup.clearCheck();            // Add the Listener to the RadioGroup          radioGroup.setOnCheckedChangeListener(              new RadioGroup                  .OnCheckedChangeListener() {                      @Override                        // The flow will come here when                      // any of the radio buttons in the radioGroup                      // has been clicked                        // Check which radio button has been clicked                      public void onCheckedChanged(RadioGroup group,                                                   int checkedId)                      {                            // Get the selected Radio Button                          RadioButton                              radioButton                              = (RadioButton)group                                    .findViewById(checkedId);                      }                  });            // Add the Listener to the Submit Button          submit.setOnClickListener(new View.OnClickListener() {                @Override              public void onClick(View v)              {                    // When submit button is clicked,                  // Ge the Radio Button which is set                  // If no Radio Button is set, -1 will be returned                  int selectedId = radioGroup.getCheckedRadioButtonId();                  if (selectedId == -1) {                      Toast.makeText(MainActivity.this,                                     "No answer has been selected",                                     Toast.LENGTH\_SHORT)                          .show();                  }                  else {                        RadioButton radioButton                          = (RadioButton)radioGroup                                .findViewById(selectedId);                        // Now display the value of selected item                      // by the Toast message                      Toast.makeText(MainActivity.this,                                     radioButton.getText(),                                     Toast.LENGTH\_SHORT)                          .show();                  }              }          });            // Add the Listener to the Submit Button          clear.setOnClickListener(new View.OnClickListener() {                @Override              public void onClick(View v)              {                    // Clear RadioGroup                  // i.e. reset all the Radio Buttons                  radioGroup.clearCheck();              }          });      }  }  **Output:** |



Understanding of the interactions between user interface and underlying application infrastructure.(CO4)

**Creating the Application Choosing Options RadioGroup**

<https://www.geeksforgeeks.org/android-how-to-add-radio-buttons-in-an-android-application/>

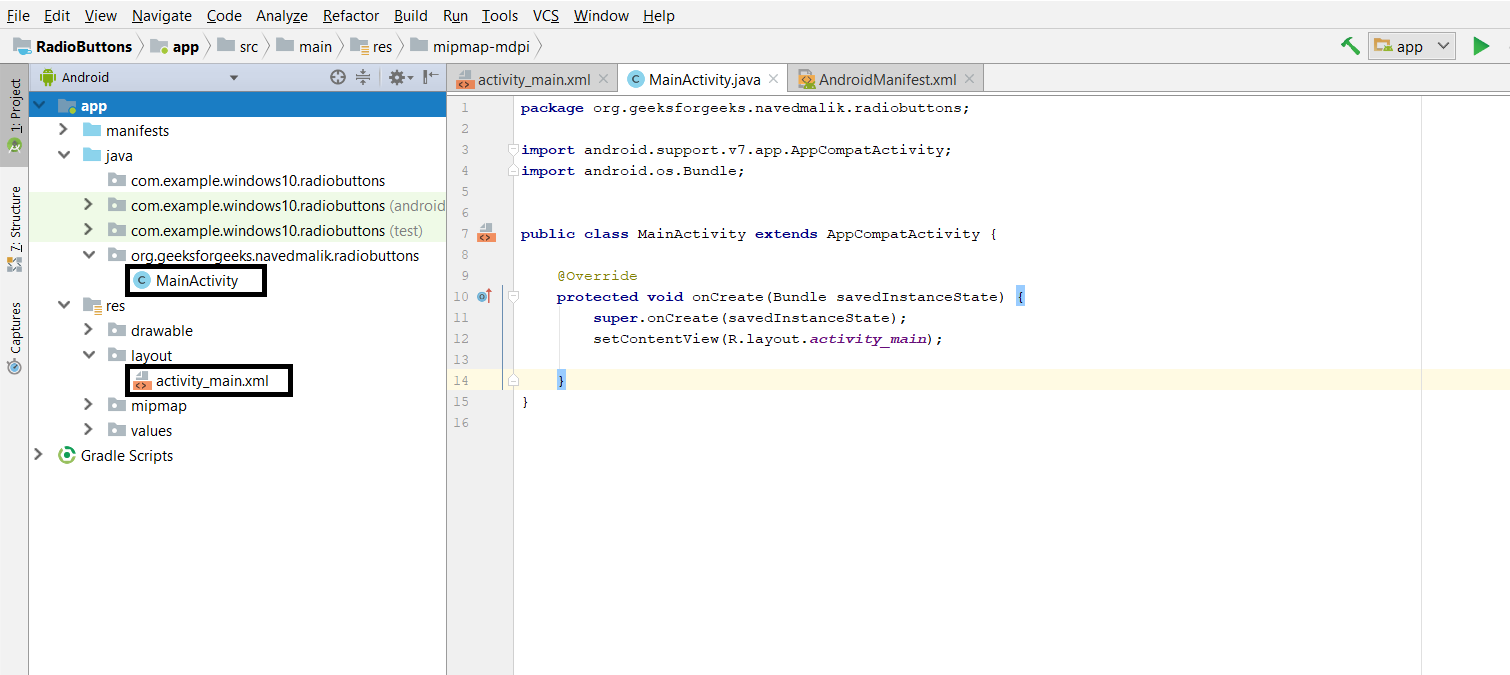
[Android](https://www.geeksforgeeks.org/tag/android/) **radio button** is a widget that can have more than one option to choose from. The user can choose only one option at a time. Each option here refers to a radio button and all the options for the topic are together referred to as Radio Group. Hence, Radio Buttons are used inside a RadioGroup.

**Pre-requisites:**

* [Android App Development Fundamentals for Beginners](https://www.geeksforgeeks.org/android-app-development-fundamentals-for-beginners/)
* [Guide to Install and Set up Android Studio](https://www.geeksforgeeks.org/guide-to-install-and-set-up-android-studio/)
* [Android | Starting with the first app/android project](https://www.geeksforgeeks.org/android-starting-with-first-app-android-project/)
* [Android | Running your first Android app](https://www.geeksforgeeks.org/android-running-your-first-android-app/)

This example will help in developing an Android App that creates Radio Buttons according to the above example:

* **Step 1:** First create a new Android Application. This will create an XML file “activity\_main.xml” and a Java File “MainActivity.Java”. Please refer the pre-requisites to learn more about this step.



**Step 2:** Open the “activity\_main.xml” file and add the following widgets in a [Relative Layout](https://www.geeksforgeeks.org/layouts-android-ui-design/):

* A **TextView** to display the question message
* A **RadioGroup** to hold the option Radio Buttons which are the possible answers
* **4 RadioButtons** to hold an answer each.
* A Submit and a Clear button to store the response.

Also, Assign the **ID** to each of the components along with other attributes as shown in the given image and the code below. The assigned ID on a component helps that component to be easily found and used in the Java files.

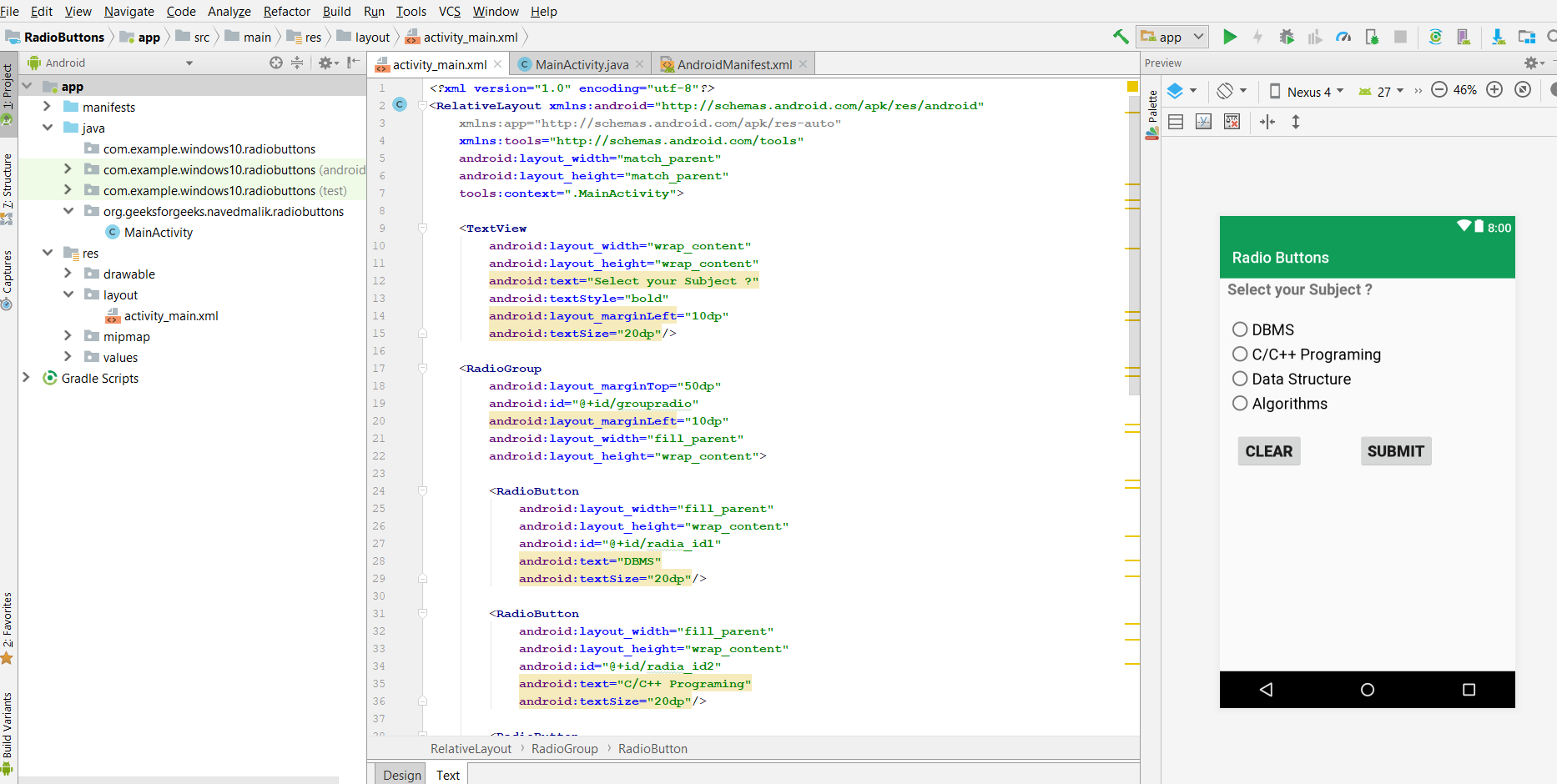
**Syntax:**

android:id="@+id/id\_name"

Here the given IDs are as follows:

* RadioGroup: groupradio
* RadioButton1: radia\_id1
* RadioButton2: radia\_id2
* RadioButton3: radia\_id3
* RadioButton4: radia\_id4
* Submit Button: submit
* Clear Button: clear

This will make the UI of the Application.



**Step 3:** Now, after the UI, this step will create the Backend of Application. For this, open the “MainActivity.java” file and instantiate the components made in the XML file (RadioGroup, TextView, Clear, and Submit Button) using findViewById() method. This method binds the created object to the UI Components with the help of the assigned ID.

**Syntax:** General

ComponentType object = (ComponentType)findViewById(R.id.IdOfTheComponent);

**Syntax:**Components used

Button submit = (Button)findViewById(R.id.submit);

Button clear = (Button)findViewById(R.id.clear);

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

**Step 4:** This step involves setting up the operations on the RadioGroup, RadioButtons, and the Submit and Clear Buttons.

 These operations are as follows:

* Unset all the Radio Buttons initially as the default value. This is done by the following command:

radioGroup.clearCheck();

* Add the Listener on the RadioGroup. This will help to know whenever the user clicks on any Radio Button, and the further operation will be performed. The listener can be added as follows:

***radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener(){}***

* Define the operations to be done when a radio button is clicked. This involves getting the specific radio button that has been clicked, using its id. Then this radio button gets set and the rest of the radio button is reset.
* Add the listener on Submit button and clear button. This will be used to check when the user clicks on the button. This is done as follows:

***submit.setOnClickListener(new View.OnClickListener() {}******clear.setOnClickListener(new View.OnClickListener() {}***

* In the Submit Button Listener, set the operations to be performed. This involves displaying the marked answer in the form of **Toast**.
* In the Clear Button Listener, set the operations to be performed. This involves resetting all the radio buttons.

**Step5:** Now run the app and operate as follows:

* When the app is opened, it displays a question with 4 answers and a clear and submit button.
* When any answer is clicked, that radio button gets set.
* Clicking on any other radio button sets that one and resets the others.
* Clicking on Submit button displays the currently marked answer as a Toast.
* Clicking on Clear button resets all the radio buttons to their default state.

**MainActivity.Java**

* Java

|  |
| --- |
| package org.geeksforgeeks.navedmalik.radiobuttons;    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 {        // Define the object for Radio Group,      // Submit and Clear buttons      private RadioGroup radioGroup;      Button submit, clear;        @Override      protected void onCreate(Bundle savedInstanceState)      {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);            // Bind the components to their respective objects          // by assigning their IDs          // with the help of findViewById() method          submit = (Button)findViewById(R.id.submit);          clear = (Button)findViewById(R.id.clear);          radioGroup = (RadioGroup)findViewById(R.id.groupradio);            // Uncheck or reset the radio buttons initially          radioGroup.clearCheck();            // Add the Listener to the RadioGroup          radioGroup.setOnCheckedChangeListener(              new RadioGroup                  .OnCheckedChangeListener() {                      @Override                        // The flow will come here when                      // any of the radio buttons in the radioGroup                      // has been clicked                        // Check which radio button has been clicked                      public void onCheckedChanged(RadioGroup group,                                                   int checkedId)                      {                            // Get the selected Radio Button                          RadioButton                              radioButton                              = (RadioButton)group                                    .findViewById(checkedId);                      }                  });            // Add the Listener to the Submit Button          submit.setOnClickListener(new View.OnClickListener() {                @Override              public void onClick(View v)              {                    // When submit button is clicked,                  // Ge the Radio Button which is set                  // If no Radio Button is set, -1 will be returned                  int selectedId = radioGroup.getCheckedRadioButtonId();                  if (selectedId == -1) {                      Toast.makeText(MainActivity.this,                                     "No answer has been selected",                                     Toast.LENGTH\_SHORT)                          .show();                  }                  else {                        RadioButton radioButton                          = (RadioButton)radioGroup                                .findViewById(selectedId);                        // Now display the value of selected item                      // by the Toast message                      Toast.makeText(MainActivity.this,                                     radioButton.getText(),                                     Toast.LENGTH\_SHORT)                          .show();                  }              }          });            // Add the Listener to the Submit Button          clear.setOnClickListener(new View.OnClickListener() {                @Override              public void onClick(View v)              {                    // Clear RadioGroup                  // i.e. reset all the Radio Buttons                  radioGroup.clearCheck();              }          });      }  }  **Output:** |

