|  |
| --- |
| **Session 1 Worksheet** |

Table of Contents

[Setting up and using Git repository 1](#_Toc70692837)

[Create an Android Project 2](#_Toc70692838)

[Section A: Create the layout 3](#_Toc70692839)

[Section B: Link UI Elements in layout xml to Java Code 5](#_Toc70692840)

[Section C: Handle User Events 9](#_Toc70692841)

[Section D: Introducing Toggle Button 11](#_Toc70692842)

[Section E: RadioButton 13](#_Toc70692843)

## Setting up and using Git repository

1. Please refer to the "**GitHub Setup Guide.pdf**" for instructions on setting up your own GitHub account.
2. From this lesson onwards, check in your projects into Git.
3. Do provide the URL to your GitHub repository here:

|  |
| --- |
|  |

1. For an Android Project to be deposited into the repository, you need to import into a Git repository. Do refer to the guide to import the project into the repository.
2. During the development, you shall commit the changes into the repository regularly.

## Create an Android Project

1. In Android Studio, create a new project with the following information:-

|  |  |
| --- | --- |
| **Project Template** | Empty Activity |
| **Application Name** | Demo Simple Click |
| **Package Name** | sg.edu.rp.c346.id<your student ID>.demosimpleclick |
| **Project Location** | D:\C346\Workspace\P03\DemoSimpleClick |
| **Language** | Java |
| **Minimum API Level** | API 16 |

Update activity\_main.xml to use the following template.

|  |
| --- |
| *<?***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"**>  </**LinearLayout**> |

Basic LinearLayout Template

**Note:** We will be using LinearLayout throughout this module. Please use LinearLayout when answering questions and quizzes.

## Section A: Create the layout

1. Open the layout file (activity\_main.xml).
2. Verify that you had updated the layout to LinearLayout (vertical).
3. Design the layout as what is shown below.
   1. The Display button takes up 20% of the screen width. The EditText takes up 80% of the screen width
   2. Use attribute "android:hint" to display the hint "Enter something" in the EditText

|  |  |
| --- | --- |
|  |  |

1. In order to display what the user types when the button is clicked, we need to add in the programming logic in Java. Before that, we need to "connect" the UI elements to the variables in the Java code.

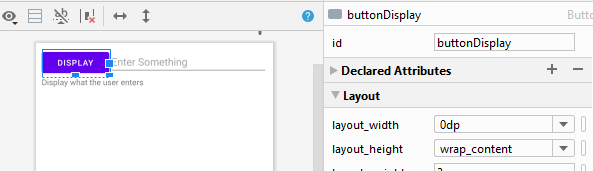
**Preparation: Add IDs for the UI elements**

In order to refer to the UI element in Java code, we need to assign an ID to the UI element. Just like how to modify the text of a UI element, we can either do it in the Text Mode or Design Mode.

**Method 1: Design Mode**

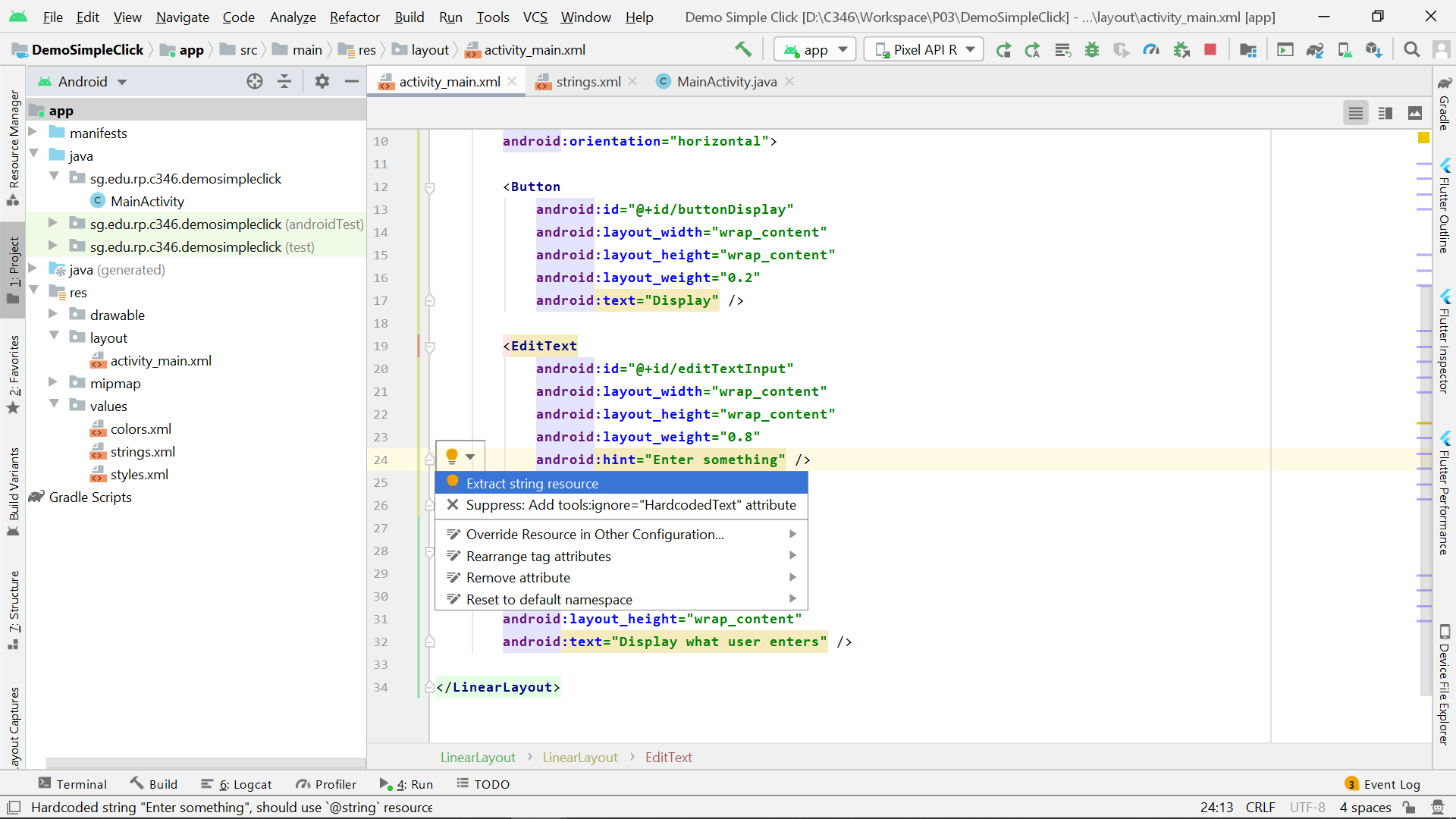
You can modify the attributes of the UI elements or controls via the design mode.

1. Click on the Button and modify the ID to buttonDisplay. Press Enter once you have typed in the ID.

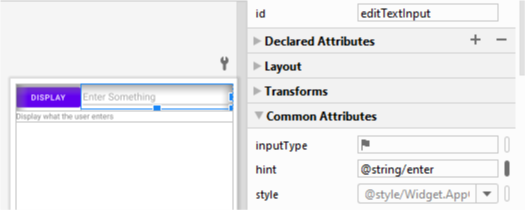


1. Click on the EditText and modify the ID to **editTextInput** and include a hint, "Enter something", using string resource.

You can select the "Extract string resource" option to remove the hardcoding of the value "Enter something" in the layout file and place it in strings.xml directly.



The default control ID is fine if the app has only one EditText or Button. If there are more, it is better to give a more specific ID to avoid any confusion.



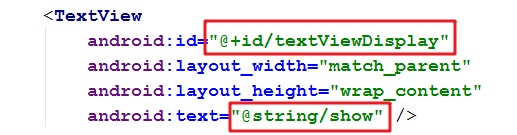
|  |  |
| --- | --- |
|  |  |

**Method 2: Text Mode**

You can modify the attributes of the UI elements or controls via the text mode manually. Let’s try it out to modify the XML code for the last UI element.

1. Switch activity\_main.xml layout file from Design Mode to Code Mode.
2. Let’s change the ID of the TextView to textViewDisplay. Find the TextView element and its id attribute, and modify by changing the default id value to textViewDisplay. Note there is a “@” in front of “+id”.

Add a text “Display what user enters” for this TextView using string resource.



activity\_main.xml

|  |
| --- |
| <**TextView  android:id="@+id/textViewDisplay"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/show"** /> |

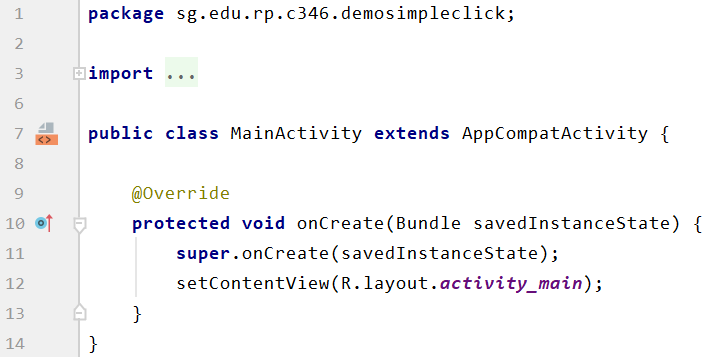
1. Start your emulator and test out the program. Enter any text in the EditText (Text Field), and click DISPLAY.

|  |  |
| --- | --- |
| ? | Noted nothing has happened to the program. Explain why there is no change in the display? |
|  | |

## Section B: Link UI Elements in layout xml to Java Code

**Step 1: Declare the variables**

1. Open the MainActivity.java file in the “java” folder. You will see the following code:



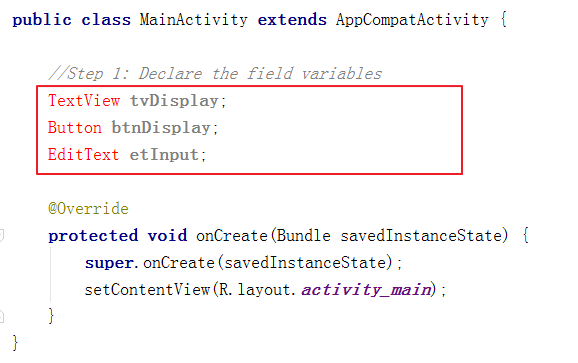
|  |  |
| --- | --- |
| ? | What is the name of the class? |
| **public class** MainActivity **extends** AppCompatActivity | |

|  |  |
| --- | --- |
| ? | What is the parent class of the class? |
|  | |

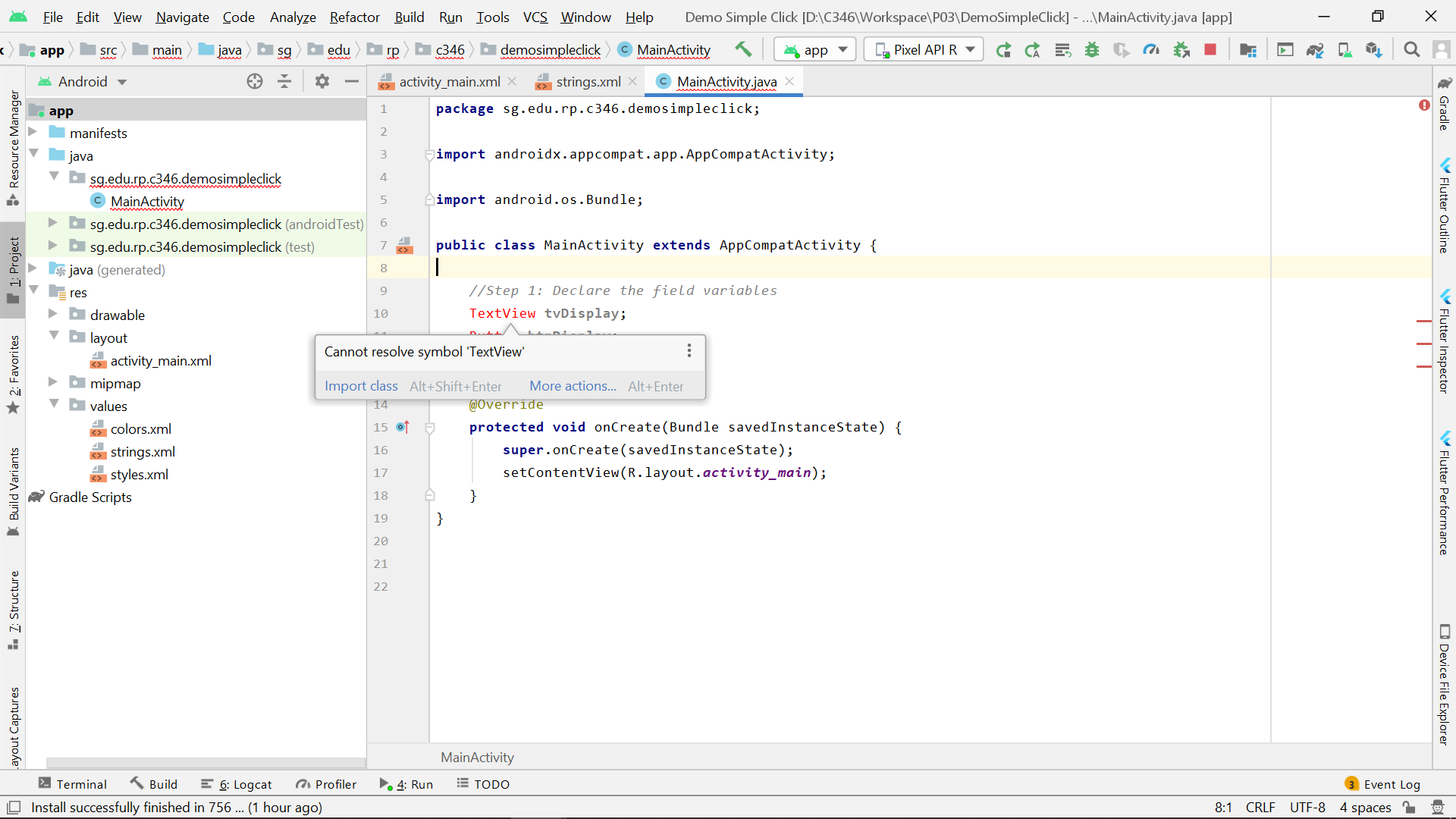
|  |  |
| --- | --- |
| ? | Write down the method name and use your own words to simply summarize the purpose of this method. |
|  | |

|  |  |
| --- | --- |
| ? | Which line of code sets the layout for the UI? |
| Line 21 | |

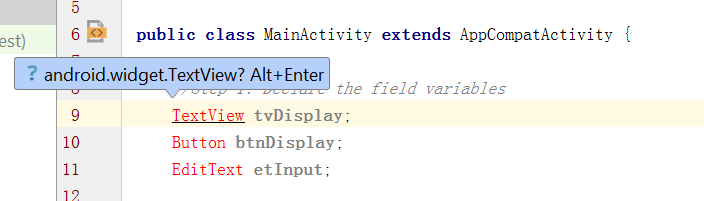
1. Next, add in the code in the red box as highlighted below. The comment above the code is for explanation only.



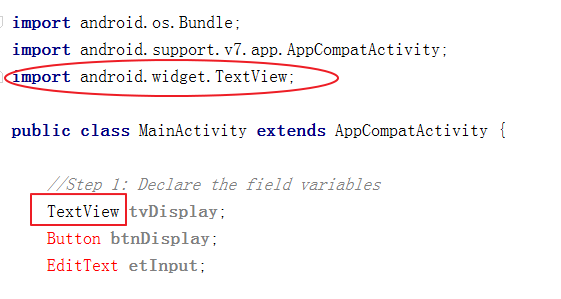
You may realize the data types are in red colour, which implies that the IDE could not find the classes that referred to them.



To import the necessary class for TextView, mouseover the code, click on the suggestion "Import class". A very useful shortcut key is to press "Alt+Enter" to import it.

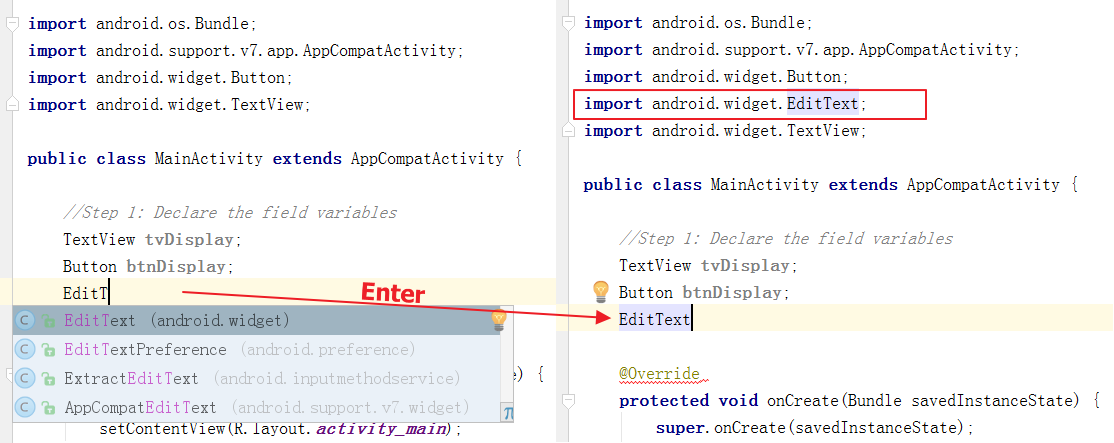


You will realize there is one line of code added above the class declaration. Also, the colour of the TextView is changed to black colour. Proceed to fix the rest of the errors.



This issue usually arises when you type in the code one by one without making use of the autocompletion.

To import automatically while you are typing, when you see the correct class appears in the help message, just press “Enter”. The class would be completed with the necessary class imported. Remove the line of EditText and the import class (if you have done it previously), and use the auto-completion method to import the class automatically.



**Step 2: Connect the variables to the respective UI elements**

1. Now, we need to link the field variable to the UI element in the layout. Follow the sample code below to link the tvDisplay variable to its UI element.

|  |
| --- |
| **tvDisplay** = findViewById(R.id.***textViewDisplay***);  Using ‘R.id’, we are indicating which ID we would like to reference to, which we have created in the XML Layout file  This is to inform we will find the view in the layout (XML Layout file) by the ID given in the argument |

|  |
| --- |
|  |

Add in the code to link the other two field variables as well. Paste the two lines of code below.

|  |
| --- |
| Button button1 = findViewById(R.id.*buttonDisplay*); TextView textViewDisplay = findViewById (R.id.*textViewInput*); |

## Section C: Handle User Events

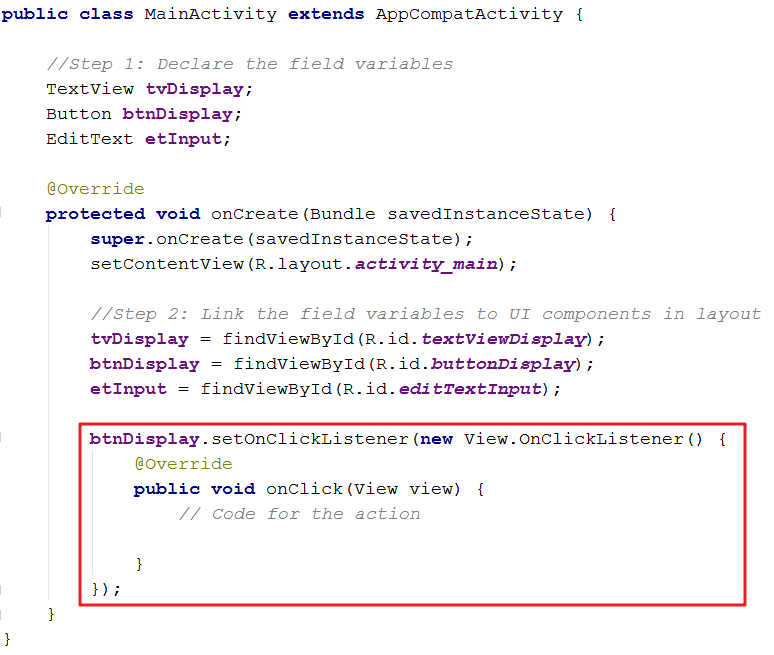
Just one more step to complete before we can make the app work. Now, in the Java class, the code “knows” that:

* The variable btnDisplay refers the Display Button
* The variable tvDisplay refers to the TextView
* The variable etInput refers to the EditText

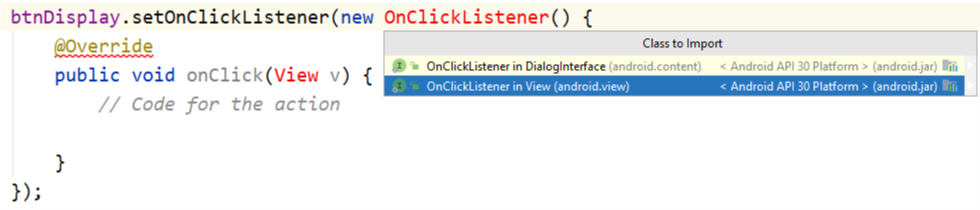
We haven’t done any code to indicate what will happen when the button is clicked. To handle the button click action, add in the following template code below **inside** the onCreate() method.

|  |
| --- |
| buttonName.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View v) {  Toast.*makeText*(MainActivity.**this**,**"Dispaly what User enter"**, Toast.***LENGTH\_SHORT***).show();      }  }); |

In our case, the buttonName should be btnDisplay, so we should replace it with the correct variable name. When you type it, make use of the autocompletion feature Android Studio provides.



If you encounter this issue below about missing class, use Alt+Enter to resolve it. For OnClickListener, there are two options, choose “OnClickListener in View (android.view)”.



Finally, let’s add in code to

* Get the text of the EditText and convert it to a string
* Assign the string to a variable
* Display the string variable in the TextView

Guideline

|  |  |
| --- | --- |
| **What to do?** | **How to do it?** |
| Get the text of EditText etInput and convert it to String. | *etInput.getText().toString();* |
| Assign it to a variable called stringResponse. | *String stringResponse = etInput.getText().toString();* |
| Display the value of *stringResponse* via the TextView tvDisplay. | *tvDisplay.setText(stringResponse);* |

With the guideline above, complete the code below the comment, “//Code for the action”, in the onClick() method. Copy and paste the two lines of code to the box below.

|  |
| --- |
| String stringResponse = **etInput**.getText().toString(); **tvDisplay**.setText(stringResponse); |

**Note:** Data retrieved from EditText will always be string.

1. Run the application. Key in “hello” in the EditText, and click Display, you should see the TextView is updated to “hello”. Try to enter some other text, and it should display the same text as well. Congratulation! You have completed the Demo Simple Click application.

|  |  |
| --- | --- |
| C:\Users\denise_quek\Desktop\Screenshot_1588632767.png | C:\Users\denise_quek\Desktop\Screenshot_1588632770.png |

MainActivity.java

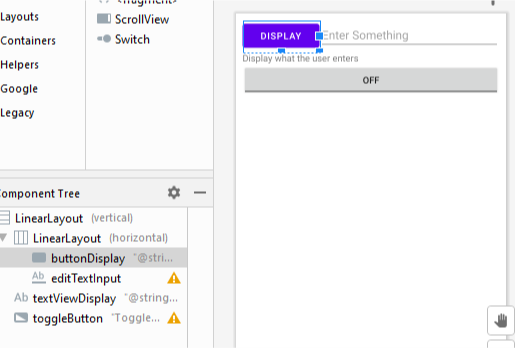
|  |
| --- |
| **package** sg.edu.rp.c346.id22023689.wssectiona;  **import** androidx.appcompat.app.AppCompatActivity;  **import** android.os.Bundle; **import** android.view.View; **import** android.widget.Button; **import** android.widget.EditText; **import** android.widget.TextView; **import** android.widget.Toast;  **public class** MainActivity **extends** AppCompatActivity {   *//Step 1: Declare the field variable* TextView **tvDisplay**;  Button **btnDisplay**;  EditText **etInput**;   @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);   Button button1 = findViewById(R.id.*buttonDisplay*);  TextView textViewDisplay = findViewById (R.id.*textViewInput*);   button1.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View v) {  **etInput**.getText().toString();  String stringResponse = **etInput**.getText().toString();  **tvDisplay**.setText(stringResponse);  }  });  } } |

## Section D: Introducing Toggle Button

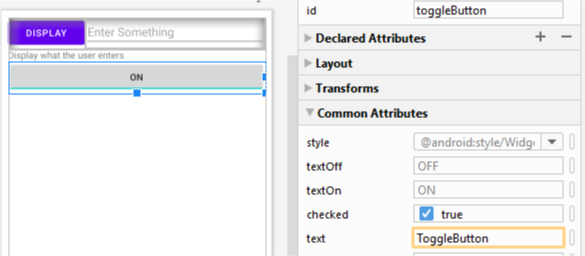
Let’s enhance the app by infusing Toggle Button into its UI. With the Toggle Button,

* the app will disable any text entry after the user unchecked the toggle button
* the app will enable the text entry when the user checks the toggle button

1. Select activity\_main.xml and switch to Design Mode
2. Let’s learn to implement the ToggleButton.
3. Drag ToggleButton from Buttons category and drop it below the TextView.



1. Select checked as the default value.



1. Go to **Attributes**, find **textOff** and **textOn** in the list, and add values to them using string resources based on the info given in the table below.

*You may refer to the worksheet of last problem on how to handle* string resources*.*

|  |  |  |
| --- | --- | --- |
|  | **String Resource Name** | **Text** |
| textOff | disabled | Disabled |
| textOn | enabled | Enabled |

1. Modify the ID of the ToggleButton to **toggleButtonEnabled**.
2. The first step to implement the code is to declare a *ToggleButton* variable in MainActivity.java as shown below. *Import the necessary class as we did in part 1.*

|  |
| --- |
| ToggleButton **tglbtn**; |

1. Next, write the code (as shown below) to bind the toggle button UI element to the variable, *tbtn* inside the *onCreate()* method*.*

|  |
| --- |
| ToggleButton tbtn = findViewById(R.id.*toggleButtonEnabled*); |

1. Using the reference code below, append your code to

* disable the EditText, editTextInput when the toggle button is unchecked
* enable it when it is checked

Test it out and make sure it works properly. Paste your code below.

|  |
| --- |
| **tbtn**.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View view) {  *// Add your code for the action*  } }); |

MainActivity.java

|  |
| --- |
| tbtn.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View v) {  **if** (**boolean** isChecked()){  textViewDisplay.setText();  } **else** {  textViewDisplay.setText(**etInput**.getText().toString());  }  } }); |

## Section E: RadioButton

1. Modify the layout by **first** adding a RadioGroup followed by two RadioButtons inside of it as below.

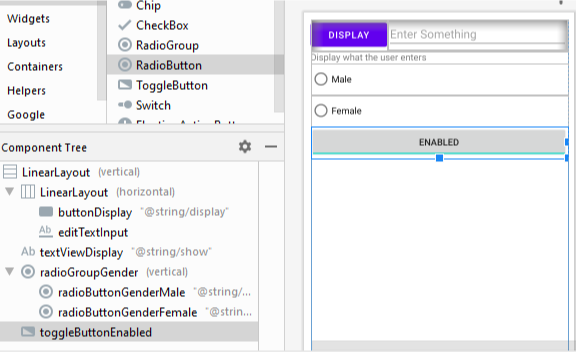
RadioGroup is used to create a **multiple-exclusion** scope for a set of RadioButtons. Checking one RadioButton that belongs to a RadioGroup unchecks any previously checked RadioButton within the same group.

For gender, we can only pick one option out of the two, so we should use RadioGroup here.

For more info about RadioGroup, check this link:

<https://developer.android.com/reference/android/widget/RadioGroup.html>

Change the ID and the text of the RadioGroup and RadioButtons to that as shown below.



1. Declare the variable for the RadioGroup in “MainActivity.java”.

|  |
| --- |
|  |

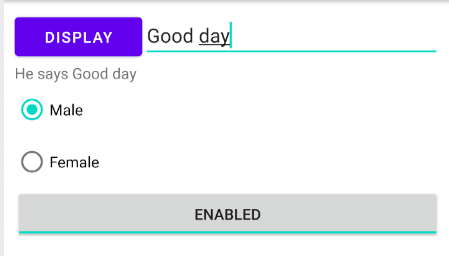
1. Bind the variable to the UI element using “findViewById()”.
2. These are the two RadioGroup methods we may use to get and set the checked RadioButton.

|  |  |
| --- | --- |
| void | [check](https://developer.android.com/reference/android/widget/RadioGroup.html#check(int))(int id)  Sets the selection to the radio button whose identifier is passed in parameter. |
| int | [getCheckedRadioButtonId](https://developer.android.com/reference/android/widget/RadioGroup.html#getCheckedRadioButtonId())()  Returns the identifier of the selected radio button in this group. |

1. Add code to retrieve the checked RadioButton when the Display button is clicked. Instead of just displaying the text from the EditText onto the TextView, append the corresponding “He says “ or “She says “ to the TextView text.

Modify the code when the Display button is clicked and paste the changes below.

|  |
| --- |
| btnDisplay.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  *// Code for the action* String stringResponse = etInput.getText().toString();  int checkedRadioId = rgGender.getCheckedRadioButtonId();  if(checkedRadioId == R.id.*radioButtonGenderMale*){  *// Write the code when male selected*  }  else{  *// Write the code when female selected*  }  tvDisplay.setText(stringResponse);  } }); |

****