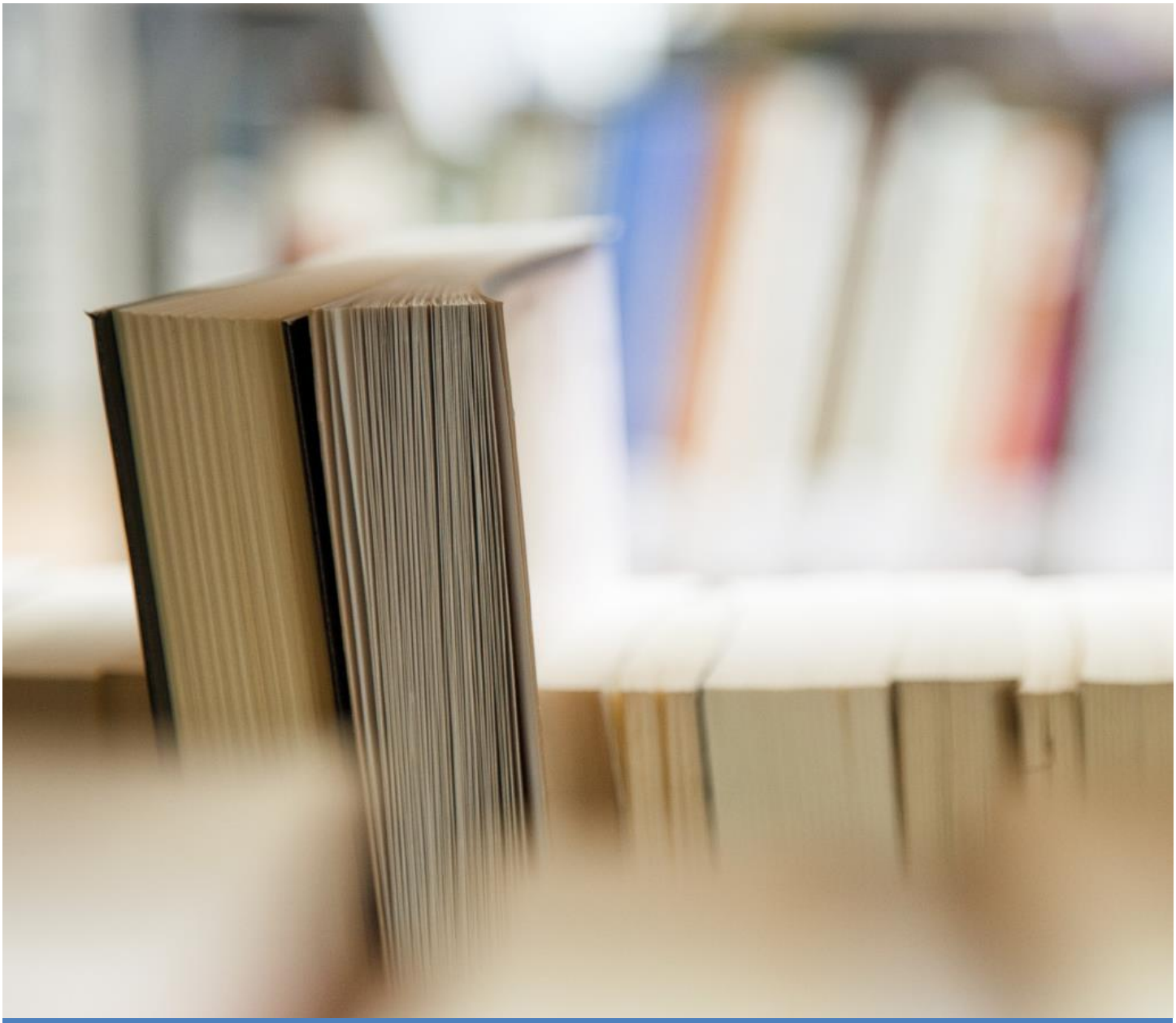


Java GUI Programming using NetBeans

Zulaile Mabni

Table of Contents

MODULE 1: NetBeans IDE Java Quick Start Tutorial

Overview
This module provides a quick introduction to the NetBeans IDE workflow. This module also introduces participants to a simple Java application program named “Hello World”.

Outcome
After completing this module, you should be able to create and run java application in the IDE.

1.1 Java Overview

Java is a high-level programming language originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995. To write your Java programs, you will need a text editor. Some of the IDEs available in the market are as follows:

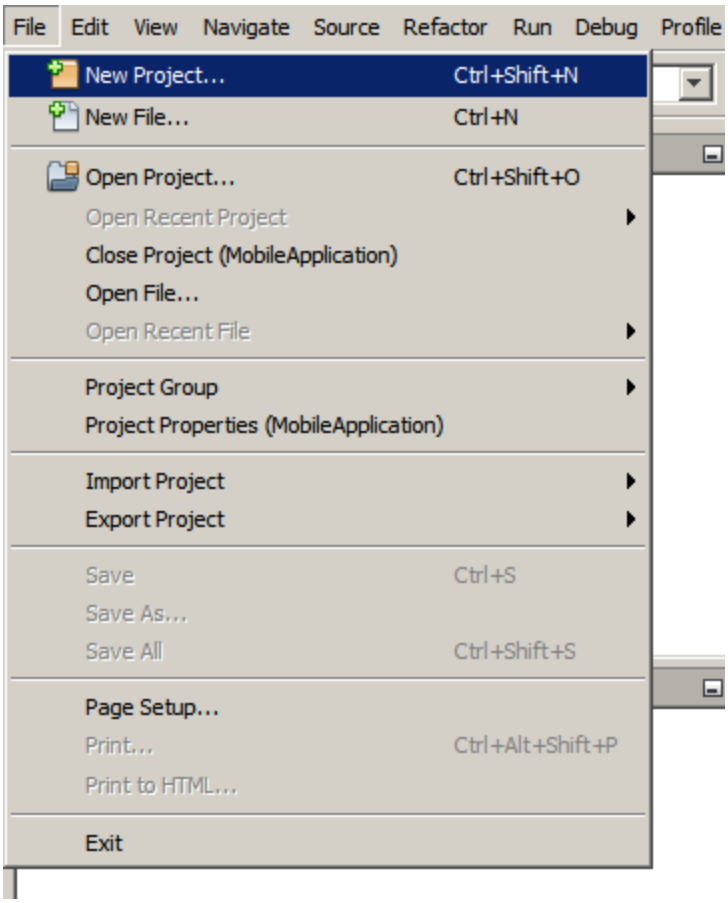
- Notepad − On Windows machine, you can use any simple text editor like Notepad.
- Netbeans − A Java IDE that is open-source and free which can be downloaded from https://www.netbeans.org/index.html.
- Eclipse − A Java IDE developed by the eclipse open-source community and can be downloaded from https://www.eclipse.org/.

1.2 Setting up the Project

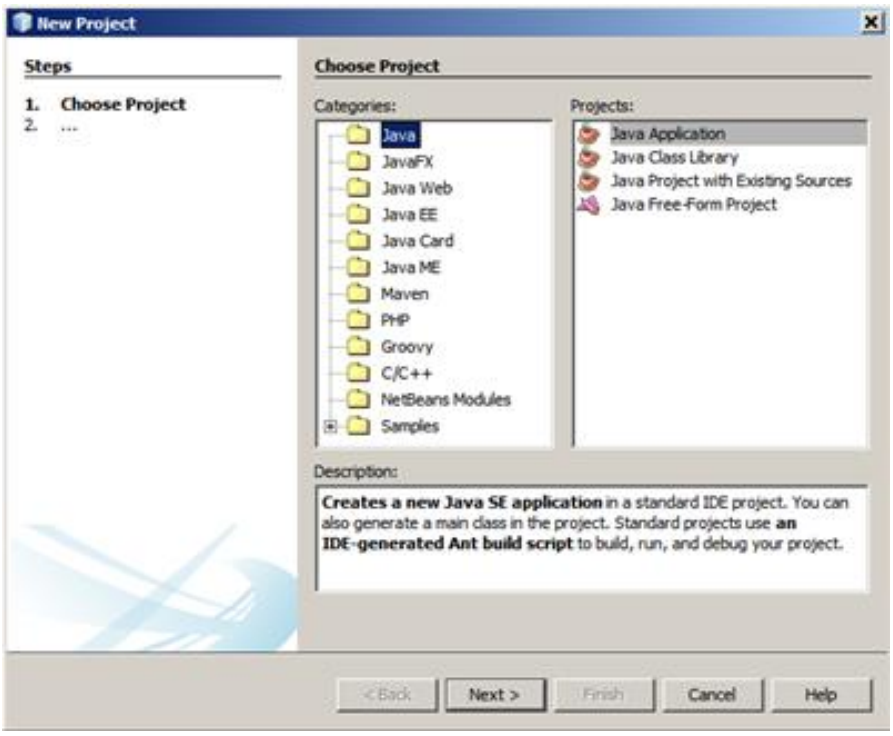
Create an IDE project:

a. Start NetBeans IDE.

b. In the IDE, choose File > New Project, as shown in the figure below.

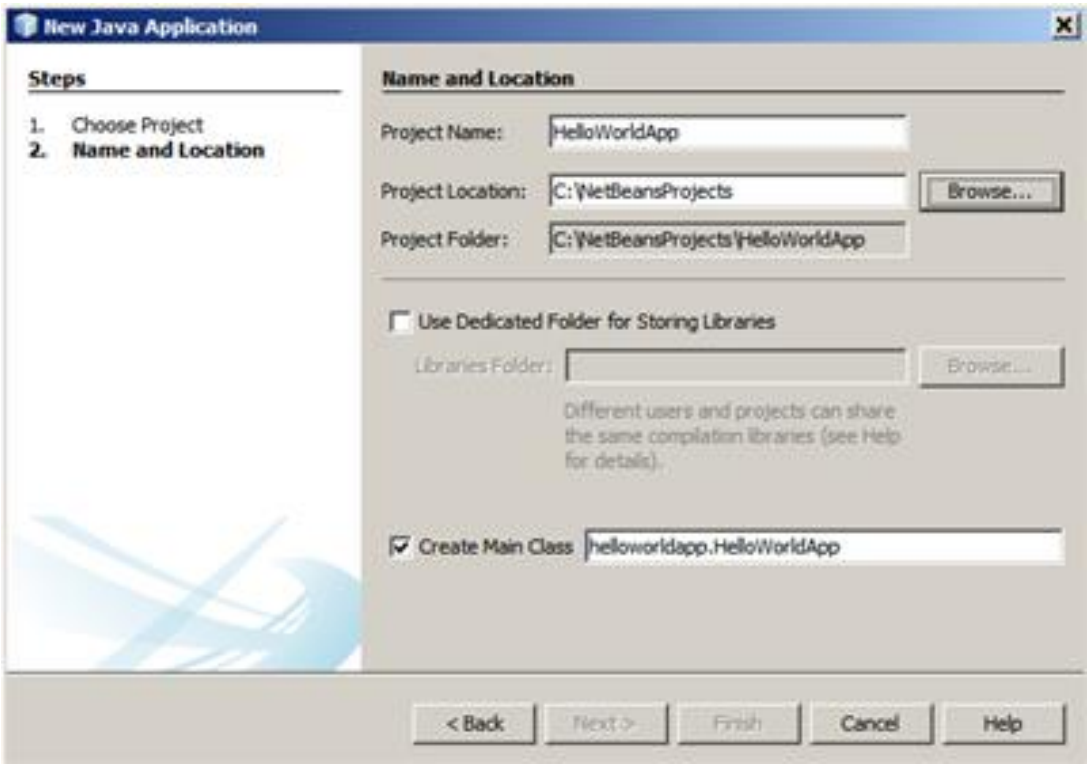


c. In the New Project wizard, expand the Java category and select Java Application as shown in the figure below. Then click Next.



d. In the Name and Location page of the wizard, do the following (as shown in the figure below):

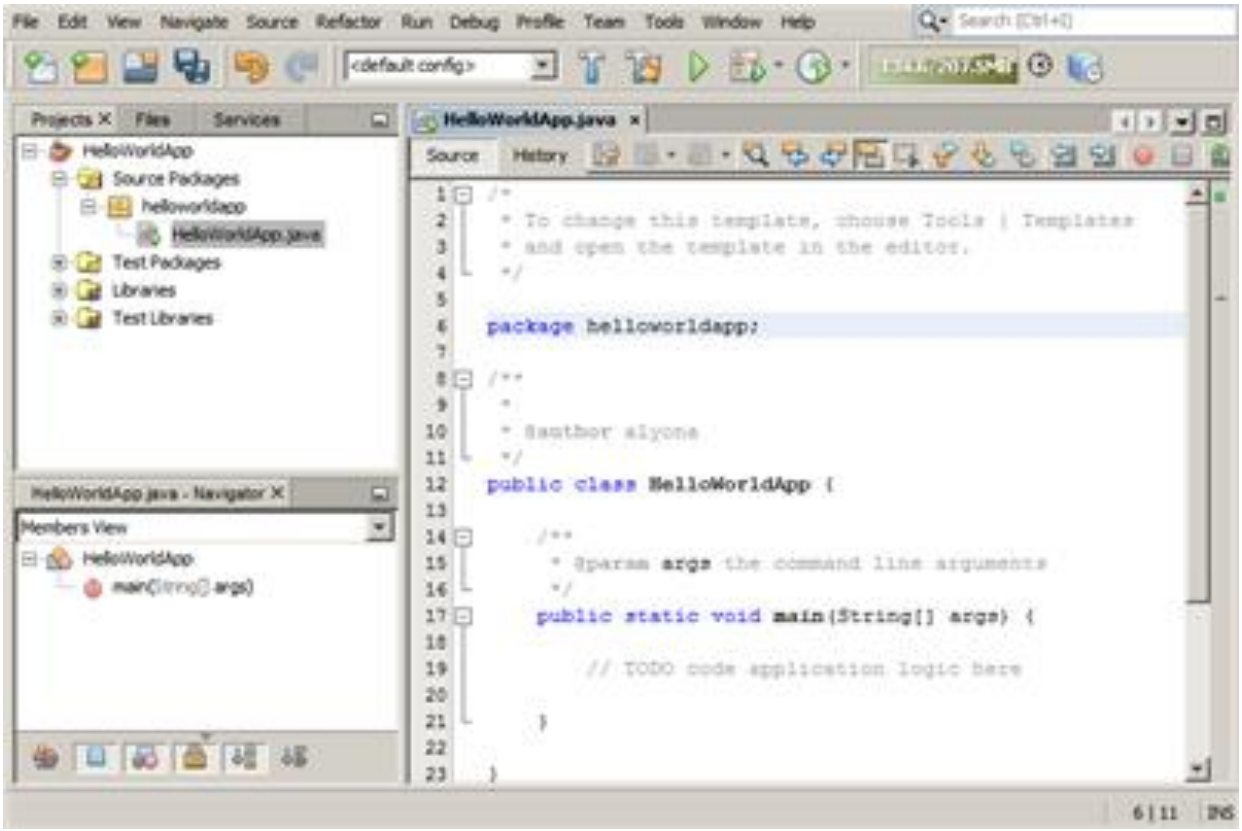
- In the Project Name field, type HelloWorldApp.
- Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
- In the Create Main Class field, type helloworldapp.HelloWorldApp.



Click Finish.

The project is created and opened in the IDE. You should see the following components:

- The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
- The Source Editor window with a file called HelloWorldApp open.
- The Navigator window, which you can use to quickly navigate between elements within the selected class.



1.3 First Java Program Using Console Output

Template for Java Program:

```
public class ClassName{
      public static void main (String[] args){
      //TODO code application logic here;
      }
}
```

Everything that you use within a Java program must be part of a class. When you write `public class ClassName`, you are defining a class named `ClassName`.

Exercise 1.1

Next, in class `HelloWorldApp` you can add the "Hello World!" message to the skeleton code by replacing the line:

```
            // TODO code application logic here

        with the line:
            System.out.println("Hello World!");
```

Save the change by choosing File > Save.

The file should look something like the following code sample.

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package helloworldapp;
/**
 *
 * @author <your name>
 */
public class HelloWorldApp {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
            System.out.println("Hello World!");
    }
}
```

1.4 Compiling and Running the Program

After you write and save an application, you must do the following steps before you can view the application's output:

- Compile the class you wrote (source code) into bytecode.
- Run the class file.

To compile the program:

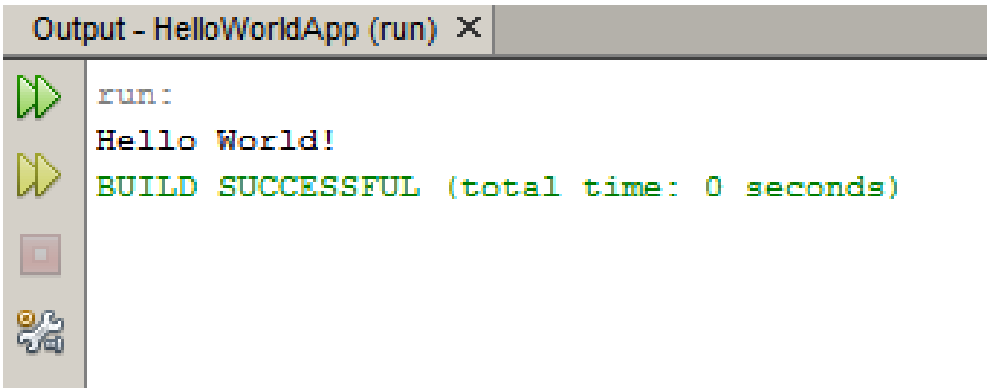
- Choose Run > Compile File

If there are compilation errors, they are marked with red glyphs in the left and right margins of the Source Editor. The glyphs in the left margin indicate errors for the corresponding lines. The glyphs in the right margin show all the areas of the file that have errors, including errors in lines that are not visible. You can mouse over an error mark to get a description of the error. You can click a glyph in the right margin to jump to the line with the error.

To run the program:

- Choose Run > Run Project.

The next figure shows what you should now see.



Congratulations! Your program works!

Exercise 1.2

Modify Exercise 1.1 above:
a. Add the source code below.
b. Save and Run the program.

```
package helloworldapp;

/**
 *
 * @author user
 */
public class HelloWorldApp {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.out.println("Programming is Fun!");
    }

}
```

MODULE 2: Introduction to GUI

Overview

This module teaches you how to create a simple graphical user interface and add simple back-end functionality. We will show how to code the behaviour of buttons and fields in a Swing form.

We will work through the layout and design of a GUI and add a few buttons and text fields. The text fields will be used for receiving user input and for displaying the program output. The button will initiate the functionality built into the front end. The application we create will be a simple but functional calculator.

Outcome

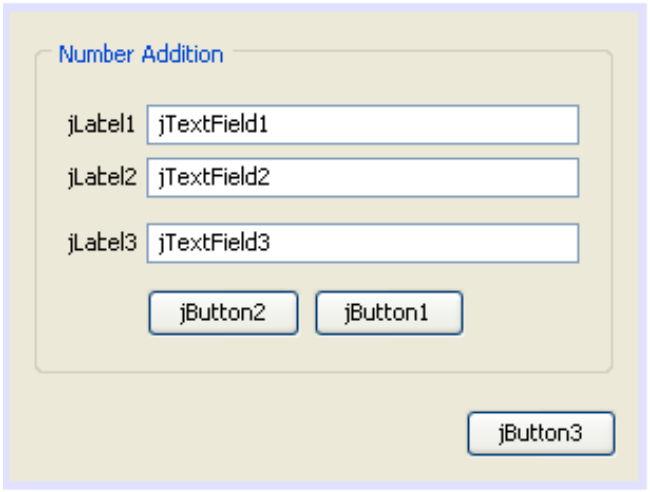
After completing this module, you should be able to create and run a GUI program.

Exercise 2.1

1. Create a Project:
   a. Choose File > New Project. Alternatively, you can click the New Project icon in the IDE toolbar.
   b. In the Categories pane, select the Java node. In the Projects pane, choose Java Application. Click Next.
   c. Type NumberAddition in the Project Name field and specify a path e.g. in your home directory as the project location

1. Create a JFrame container:
   a. In the Projects window, right-click the NumberAddition node and choose New > JFrame Form.
   b. Enter NumberAdditionUI as the class name.
   c. Click Finish.

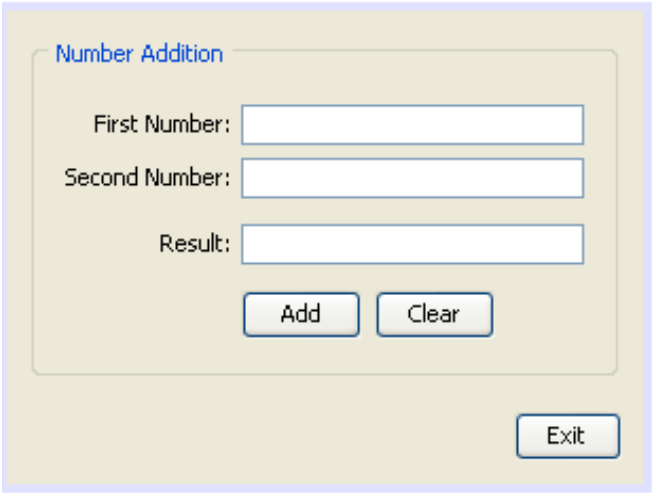
2. Adding components:
   a. Start by selecting a JPanel from the Palette and drop it onto the JFrame. While the JPanel is highlighted, go to the Properties window and click the ellipsis (...) button next to Border to choose a border style.
   b. In the Border dialog, select TitledBorder from the list, and type in Number Addition in the Title field.
   c. Add three JLabels, three JTextFields and three JButtons on the panel as shown in the screenshot below. Click OK to save the changes and exit the dialog.



4. Renaming the components:

   Next step is to rename the display text of all the components on the JFrame.

   a. Double click jLabel1 and change the text property to First Number:
   b. Double click jLabel2 and change the text property to Second Number:
   c. Delete the sample text from jTextField1, jTextField2 and jTextField3. Make the display text editable by right-clicking the text field and choosing Edit Text from the popup menu.
   d. Continue renaming the display text for all the components until you get the interface as shown in the screenshot below.



5. Adding Functionality:

   Exit button

   a. Right click the Exit button. From the pop-up menu choose Events > Action > ActionPerformed
   b. Add code for the Exit Button to do as below:

```
System.exit(0);
```

Clear button

a. Right click the Clear button (jButton1). From the pop-up menu select Events > Action > actionPerformed
b. Add code as below:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

      jTextField1.setText("");
      jTextField2.setText("");
      jTextField3.setText("");
 }
```

Add Button

a. Right-click the Add button (jButton2). From the pop-up menu, select Events > Action > actionPerformed
b. Add code as below:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt){

      float num1, num2, result; // Declare float variables.
      // Parse the text to a type float.
      num1 = Float.parseFloat(jTextField1.getText());
      num2 = Float.parseFloat(jTextField2.getText());

      result = num1 + num2;
      jTextField3.setText(String.valueOf(result));
```

c. Save and Run the program.

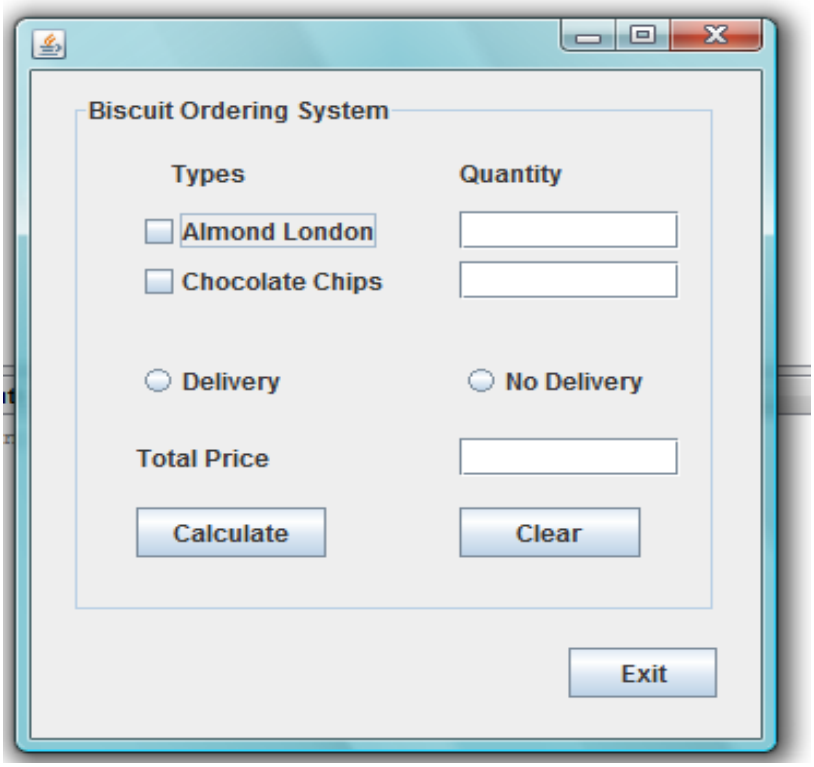
Exercise 2.2

Raya Biscuit Sdn. Bhd. has requested from you to write a GUI program to calculate the total price to be paid by the customer for the biscuits that they purchased. Write a Java program to develop the GUI interface as shown below.

1. Create a Project:
   a. Choose File > New Project. Alternatively, you can click the New Project icon in the IDE toolbar.
   b. In the Categories pane, select the Java node. In the Projects pane, choose Java Application. Click Next.
   c. Type BiscuitOrder in the Project Name field and specify a path e.g. in your home directory as the project location.

2. Create a JFrame container:
   a. In the Projects window, right-click the BiscuitOrder node and choose New > JFrame Form.
   b. Enter BiscuitOrderUI as the class name.
   c. Click Finish.

3. Adding components:
   a. Start by selecting a JPanel from the Palette and drop it onto the JFrame. While the JPanel is highlighted, go to the Properties window and click the ellipsis (...) button next to Border to choose a border style.
   b. In the Border dialog, select TitledBorder from the list, and type in Biscuit Ordering System in the Title field.
   c. Next, add JLabels, JTextFields, JCheckBoxes, JRadioButtons and JButtons on the panel to create an interface as shown in the screenshot below. Click OK to save the changes and exit the dialog.



4. Adding Functionality:

Exit button

a. Right click the Exit button. From the pop-up menu choose Events > Action > ActionPerformed
b. Add code for the Exit Button to do as below:

```
System.exit(0);
```

c. Right click the Clear button (jButton2). From the pop-up menu select Events > Action > actionPerformed
d. Add code as below:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        jTextField1.setText("");
        jTextField2.setText("");
        jTextField3.setText("");
        jCheckBox1.setSelected(false);
        jCheckBox2.setSelected(false);
        jRadioButton1.setSelected(false);
        jRadioButton2.setSelected(false);
}
```

Calculate Button

a. Right-click the Calculate button (jButton1). From the pop-up menu, select Events > Action > actionPerformed
b. Add code as below:

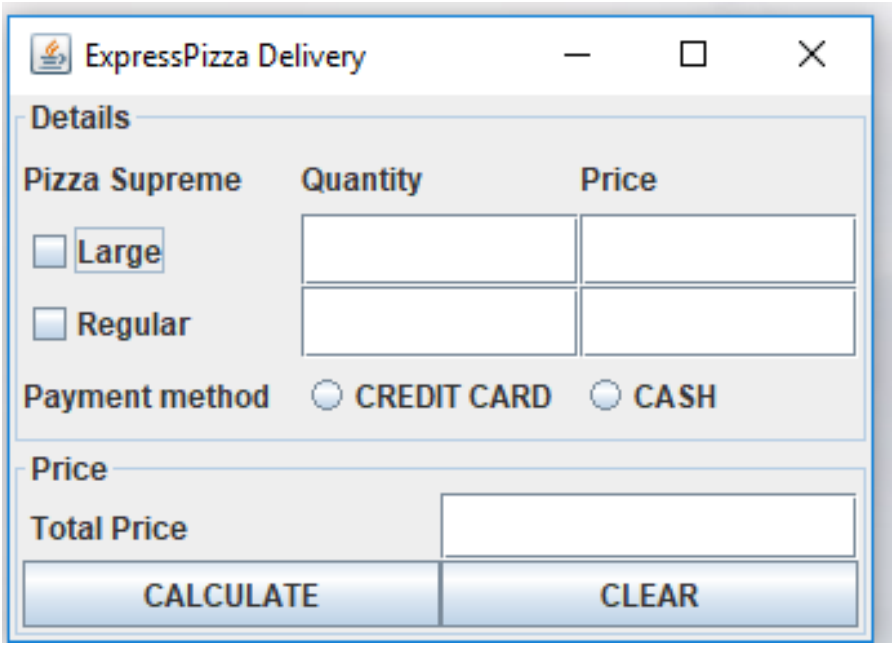
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){

      double almondQty = 0, chocchipQty = 0, total, deliveryChrg;
        if (jCheckBox1.isSelected()) {
            almondQty = Double.parseDouble(jTextField1.getText());
        }
        if (jCheckBox2.isSelected()) {
            chocchipQty = Double.parseDouble(jTextField2.getText());
        }
        if (jRadioButton1.isSelected()) {
            deliveryChrg = 2.50;
        }
        else {
            deliveryChrg = 0;
        }
       total = deliveryChrg + ((almondQty * 0.50) + (chocchipQty * 0.40));
       jTextField3.setText(String.valueOf(total));
}
```

c. Save and Run the program.

Project

Given the following interface:



a. Design the GUI program to illustrate the above interface using NetBeans GUI features.
b. Write the codes for adding functionality to each button.
c. Save and Run the program.

REFERENCES:

Joyce Farrel. *Java Programming*. Fifth Edition, Course Technology. 2015. ISBN 9781285856919.

Y. Daniel Liang. *Introduction to Java Programming*. Eighth Edition. Prentice Hall. 2011.

https://netbeans.org/kb/docs/java/quickstart.html

https://netbeans.org/kb/docs/java/gui-functionality.html

https://www.tutorialspoint.com/java/java_overview.htm