UMS

UNIVERSITI MALAYSIA SABAH

Fakulti Komputeran dan Informatik

SEMESTER II

SESSION 2019/2020

Individual Project

KP14603 Object Oriented Programming Concept

Lecturer :Siti Hasnah Binti Tanalol

Name : INTAN NAJIHAH BINTI SAMSUDIN

Matric No : BI19110164

CONTENTS

INTRODUCTION

GUI stands for Graphical User Interface, a term used not only in Java but in all programming languages that support the development of GUIs. A program's graphical user interface presents an easy-to-use visual display to the user. It is made up of graphical components (e.g., buttons, labels, windows) through which the user can interact with the page or application.

A major part of creating a graphical user interface in Java is figuring out how to position and lay out the components of the user interface to match the appearance you desire. Once you have chosen and laid out these components, you must make the events interactive by making them respond to various user events such as button clicks or mouse movements.There are many predefined components, but you can also define components that draw custom two-dimensional graphics, including animations.

For this project, I had done a simple GUI calculator to calculate BMI.

OBJECTIVE

1. To perform bmi calculation using user info.
2. To make sure the gui can perform smoothly without error.
3. To make sure to use OOP concept in gui.

JAVA CODE

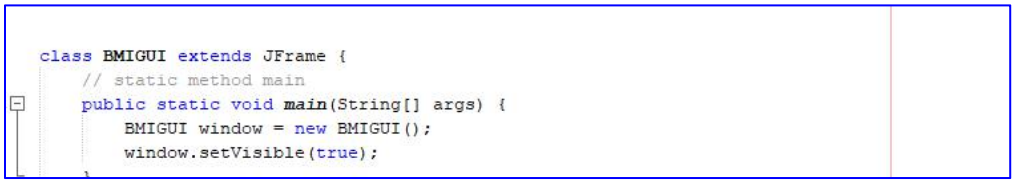
```
//Calculator to calculate BMI
//Name: Intan Najihah Samsudin
//matric No: BI19110164

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;



class BMIGUI extends JFrame {
	// static method main
	public static void main(String[] args) {
		BMIGUI window = new BMIGUI();
		window.setVisible(true);
	}

	// instance variables
	// Declare and initialize instance variables that are

	private final JTextField _mField      = new JTextField(4);    // height
	private final JTextField _kgField     = new JTextField(4);    // weight
	private final JTextField _bmiField = new JTextField(4);    // BMI

	// constructor
	public BMIGUI() {
		//... Create button and add action listener.
		JButton bmiButton = new JButton("Compute BMI");
		bmiButton.addActionListener(new BMIListener());

		//Set layout and add components.
		JPanel content = new JPanel();
		content.setLayout(new FlowLayout());
```

```
        content.add(new JLabel("Weight in kilograms"));
        content.add(_kgField);
        content.add(new JLabel("Height in meters"));
        content.add(_mField);
        content.add(bmiButton);
        content.add(new JLabel("Your BMI is"));
        content.add(_bmiField);

        // Set the window characteristics.
        setContentPane(content);
        setTitle("Body Mass Index");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();                                    // Do layout.
        setLocationRelativeTo(null);          // Center window.
    }

    // inner class BMIListener
    // Inner class is used to access components.
    private class BMIListener implements ActionListener {

  @Override
        public void actionPerformed(ActionEvent e) {
            double kilograms = Double.parseDouble(_kgField.getText());
            double meters       = Double.parseDouble(_mField.getText());
            int       bmi            = (int)computeBMI(kilograms, meters);
            _bmiField.setText("" + bmi);
        }
    }

    //formula to computeBMI
    public static double computeBMI(double weight, double height) {
        return weight / (height * height);
    }
}
```

Object Oriented Concept Implementation

1. Class

CLASS are a blueprint or a set of instructions to build a specific type of object. It is a basic concept of Object-Oriented Programming which revolve around the real-life entities. Class in Java determines how an object will behave and what the object will contain



```
class BMIGUI extends JFrame {
    // static method main
    public static void main(String[] args) {
        BMIGUI window = new BMIGUI();
        window.setVisible(true);
    }
```

2. Object

OBJECT is an instance of a class. An object is nothing but a self-contained component which consists of methods and properties to make a particular type of data useful.

```
class BMIGUI extends JFrame {
    // static method main
    public static void main(String[] args) {
        BMIGUI window = new BMIGUI();
        window.setVisible(true);
    }

    // instance variables
    // Declare and initialize instance variables that are

    private final JTextField _mField   = new JTextField(4);  // height
    private final JTextField _kgField  = new JTextField(4);  // weight
    private final JTextField _bmiField = new JTextField(4);  // BMI
```

3. Encapsulation

Encapsulation in Java is a mechanism to wrap up variables(data) and methods(code) together as a single unit. It is the process of hiding information details and protecting data and behavior of the object. It is one of the four important OOP concepts.

```
44
45      // inner class BMIListener
46      // Inner class is used to access components.
47      private class BMIListener implements ActionListener {
48
49          public void actionPerformed(ActionEvent e) {
50              double kilograms = Double.parseDouble(_kgField.getText());
51              double meters    = Double.parseDouble(_mField.getText());
52              int    bmi       = (int)computeBMI(kilograms, meters);
53              _bmiField.setText("" + bmi);
54          }
55      }
56
57      //formula to computeBMI
58      public static double computeBMI(double weight, double height) {
59          return weight / (height * height);
60      }
61  }
```

4. Variable Declaration

Variable in Java is a data container that stores the data values during Java program execution. Every variable is assigned data type which designates the type and quantity of value it can hold. Variable is a memory location name of the data. To declare a variable, you must specify the data type & give the variable a unique name.

```
45      // inner class BMIListener
46      // Inner class is used to access components.
47      private class BMIListener implements ActionListener {
48
49          public void actionPerformed(ActionEvent e) {
50              double kilograms = Double.parseDouble(_kgField.getText());
51              double meters    = Double.parseDouble(_mField.getText());
52              int    bmi       = (int)computeBMI(kilograms, meters);
53              _bmiField.setText("" + bmi);
54          }
55      }
```

5. Polymorphism

Polymorphism is the concept where an object behaves differently in different situations. Runtime polymorphism is implemented when we have an “IS-A” relationship between objects. This is also called a method overriding because the subclass has to override the superclass method for runtime polymorphism.

```
46      // inner class is used to access components.
47      private class BMIListener implements ActionListener {
48
49          @Override
50          public void actionPerformed(ActionEvent e) {
51              double kilograms = Double.parseDouble(_kgField.getText());
52              double meters    = Double.parseDouble(_mField.getText());
53              int    bmi       = (int)computeBMI(kilograms, meters);
54              _bmiField.setText("" + bmi);
55          }
56      }
57
58      //formula to computeBMI
59      public static double computeBMI(double weight, double height) {
60          return weight / (height * height);
61      }
62  }
```

6. Interface

Interface looks like a class but it is not a class. An interface can have methods and variables just like the class but the methods declared in interface are by default abstract. Private class BMIListener implements ActionListener.

```
    private class BMIListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
            double kilograms = Double.parseDouble(_kgField.getText());
            double meters    = Double.parseDouble(_mField.getText());
            int    bmi       = (int)computeBMI(kilograms, meters);
            _bmiField.setText("" + bmi);
        }
    }

    //formula to computeBMI
    public static double computeBMI(double weight, double height) {
        return weight / (height * height);
    }
}
```

Read and Write Implementation

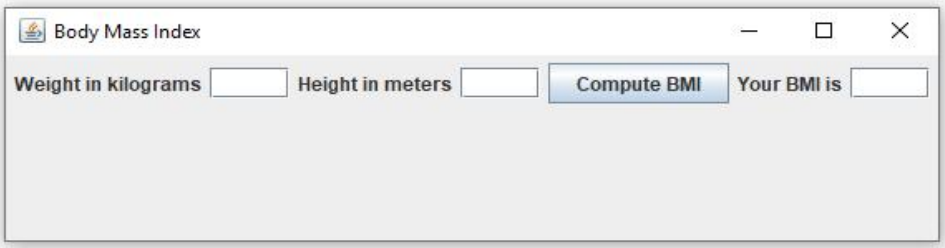
In this project, I use JTextField. It is a lightweight component that allows the editing of a single line of text. . JTextField is intended to be source-compatible with java.awt.TextField where it is reasonable to do so. This component has capabilities not found in the java.awt.TextField class. The superclass should be consulted for additional capabilities. A text field is a box into which the user can type text strings. A text field is represented by a JTextField object, which can be constructed with a parameter specifying the number of characters that should be able to fit in the text field

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

```
class BMIGUI extends JFrame {
    // static method main
    public static void main(String[] args) {
        BMIGUI window = new BMIGUI();
        window.setVisible(true);
    }

    // instance variables
    // Declare and initialize instance variables that are

    private final JTextField _mField   = new JTextField(4);  // height
    private final JTextField _kgField  = new JTextField(4);  // weight
    private final JTextField _bmiField = new JTextField(4);  // BMI
```

User Manual (How to use the system)

1. This is the final output/results of the java code.



2. Insert the weight in kg and height in meters in the both text field.

3. After the both text field are insert with value, click the “ComputeBMI” button. The BMI value will will be calculated and display.

CONCLUSION

GUIs are potentially very complex entities because they involve a large number of interacting objects and classes. Each onscreen component and window is represented by an object, so a programmer starting out with GUIs must learn many new class, method, and package names. In addition, if the GUI is to perform sophisticated tasks, the objects must interact with each other and call each other's methods, which raises tricky communication and scoping issues.

Another factor that makes writing GUIs challenging is that the path of code execution becomes nondeterministic. When a GUI program is running, the user can clickany of the buttons and interact with any of the other onscreen components in any order. Because the program's execution is driven by the series of events that occur, we say that programs with GUIs are event-driven. In this chapter you'll learn how to handle user events so that your event-driven graphical programs will respond appropriately to user interaction.