جامعة جدة
University of Jeddah

College of Computer Science & Engineering

Department of Computer Science and Artificial Intelligence

CCCS214: Object-Oriented Programming II

Lab 2: A First Look at GUI Applications

OOP2 – Fall 2022

Lab 2: A First Look at GUI Applications

Lab Objectives

- To become familiar with common user-interface components, such as text components, radio buttons and check boxes.
- To build programs that handle events from user-interface components.

Graphical User Interface (GUI)

The graphical user interface or GUI, enables a user to interact with program using graphics instead of text. A GUI consists of a window with elements such as buttons on it that you can click on to perform tasks. The user can click or type on the GUI using mouse or keyboard to input some data to the program. The program can display graphical and text output on the GUI. In this lab you will learn how to use Swing GUI components from the javax.swing package to create GUIs for your program.

Frames

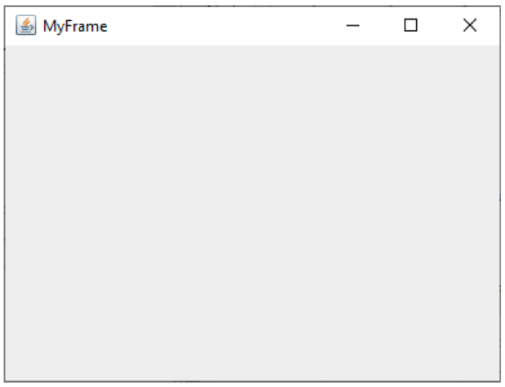
A window is a component that contains other components, so it is more appropriately considered a container. A container is simply a component that holds other components. In GUI terminology, a container that can be displayed as a window is known as a frame.

1. Creating Frame (Window)

To create a frame, use the JFrame class. The following program will create a frame:

```
import javax.swing.JFrame;
public class MyFrame {

	public static void main(String[] args) {

		JFrame frame = new JFrame("MyFrame"); // Create a frame
		frame.setSize(400, 300); // Set the frame size
		frame.setLocationRelativeTo(null); // Center the frame
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setVisible(true); // Display the frame
	}
}
```

Run the program and you will get this output:



The frame is not displayed until the **frame.setVisible(true)** method is invoked. **frame.setSize(400, 300)** specifies that the frame is **400** pixels wide and **300** pixels high. If the **setSize** method is not used, the frame will be sized to display just the title bar. Since the **setSize** and **setVisible** methods are both defined in the **Component** class, they are inherited by the **JFrame** class. Later you will see that these methods are also useful in many other subclasses of **Component**.

2- Adding Components to a Frame Using a Panel

Using the **add** method, you can add components into the frame as shown in below code.

KiloConverter Example

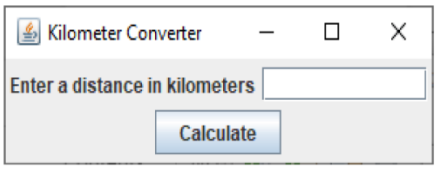
This example was discussed during the lectures. Try to implement it on netbeans so you can practice what you have learned practically. We use a panel to hold the other components. We defined the **buildPanel** method in which we put all the code needed to build the panel. Main is embedded in this class.

```
import javax.swing.*; // Needed for Swing classes
/**
* This class defines a GUI window for converting Kilos to Miles.
* It uses a panel to hold all the components.
* It has a method to build the panel.
* It has an embedded main method.

*/
```

```
public class KiloConverter extends JFrame
{

        private JPanel panel; //To reference a panel
        private JLabel messageLabel; // To reference a label
        private JTextField kiloTextField; // To reference a text field
        private JButton calcButton; // To reference a button
        final int WINDOW_WIDTH = 310; // Window width in pixels
        final int WINDOW_HEIGHT = 100; // Window height in pixels
        /**
        Constructor
       */

        public KiloConverter()
        {
                // Set this window's title.
                setTitle("Kilometer Converter");

                // Set the size of this window.
                setSize(WINDOW_WIDTH, WINDOW_HEIGHT);

                // Specify what happens when the close button is clicked.
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                // Build the panel and add it to the frame.
                  buildPanel();


              // Add the panel to the frame's content pane.
                add(panel);

                // Center the frame
                setLocationRelativeTo(null);

                // Display the window.
                setVisible(true);
        }


        private void buildPanel(){

        // Create a JPanel object and let the panel
        // field reference it.
```

```
        panel = new JPanel();

        // Create a label to display instructions.
        messageLabel = new JLabel("Enter a distance " +
        "in kilometers");

        // Create a text field 10 characters wide.
        kiloTextField = new JTextField(10);

        // Create a button with the caption "Calculate".
        calcButton = new JButton("Calculate");

        // Add the label, text field, and button
        // components to the panel.
        panel.add(messageLabel);
        panel.add(kiloTextField);
        panel.add(calcButton);
        }
        /**
        The main method creates an instance of the KiloConverter
        class, which causes it to display its window.
        */
        public static void main(String[] args)
        {
                KiloConverter kc = new KiloConverter();
        }
}
```

write the code and run the program to get this output:



3. JButton Event Handling – Example

A button is a component the user clicks to trigger a specific action. A Java application can use several types of buttons, including command buttons, checkboxes and radio buttons.

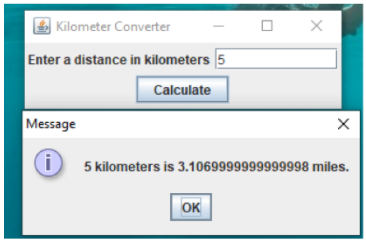
In the previous example add the **bolded** code to handle the event of clicking the button. Event handling for the buttons is performed by a single instance of inner class.

```
import javax.swing.*; // Needed for Swing classes
import java.awt.event.*; // Needed for ActionListener Interface
/**
* This class defines a GUI window for converting Kilos to Miles.
* It uses a panel to hold all the components.
* It has a method to build the panel.
* It has an embedded main method.
*/


public class KiloConverter extends JFrame {

        private JPanel panel; //To reference a panel
        private JLabel messageLabel; // To reference a label
        private JTextField kiloTextField; // To reference a text field
        private JButton calcButton; // To reference a button
        final int WINDOW_WIDTH = 310; // Window width in pixels
        final int WINDOW_HEIGHT = 100; // Window height in pixels

        /**
        Constructor
       */

        public KiloConverter()
        {
                // Set this window's title.
                setTitle("Kilometer Converter");

                // Set the size of this window.
                setSize(WINDOW_WIDTH, WINDOW_HEIGHT);

                // Specify what happens when the close button is clicked.
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                // Build the panel and add it to the frame.
                buildPanel();
```

```
        // Add the panel to the frame's content pane.
        add(panel);

        // Center the frame
        setLocationRelativeTo(null);

        // Display the window.
        setVisible(true);
}


private void buildPanel(){

        // Create a JPanel object and let the panel
        // field reference it.
        panel = new JPanel();

        // Create a label to display instructions.
        messageLabel = new JLabel("Enter a distance " +
        "in kilometers");

        // Create a text field 10 characters wide.
        kiloTextField = new JTextField(10);

        // Create a button with the caption "Calculate".
        calcButton = new JButton("Calculate");

        // Add an action listener to the button.
        calcButton.addActionListener(new CalcButtonListener());

        // Add the label, text field, and button
        // components to the panel.
        panel.add(messageLabel);
        panel.add(kiloTextField);
        panel.add(calcButton);

}

private class CalcButtonListener implements ActionListener{
        /**
        The actionPerformed method executes when the user  clicks on the Calculate button.
        @param e The event object.
        */

public void actionPerformed(ActionEvent e)
{
        final double CONVERSION = 0.6214; //Constant used for the conversion
        String input; // To hold the user's input
```

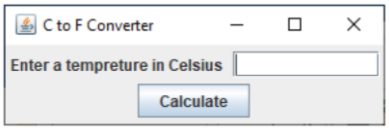
```
                double miles; // The number of miles

                // Get the text entered by the user into the
                // text field.
                input = kiloTextField.getText();

                // Convert the input to miles.
                miles = Double.parseDouble(input) * CONVERSION;

                // Display the result.
                JOptionPane.showMessageDialog(null, input +
                " kilometers is " + miles + " miles.");
                }
        }

        /**
        The main method creates an instance of the KiloConverter
        class, which causes it to display its window.
        */
        public static void main(String[] args)
        {
                KiloConverter kc = new KiloConverter();
        }
}
```

Write the code and run the program to get this output:



Task # 1: (C to F converter)

(Using KiloConverter Example as a guide) Create the following GUI. It is a GUI application that converts Celsius temperatures to Fahrenheit temperatures. You do not have to provide any functionality.



Task # 2: (C to F converter)

(Using KiloConverter Example as a guide) Add functionality to the Calculate button in Task # 1 GUI, and make the program calculate the Fahrenheit temperature from the entered Celsius temperature.
Use the formula:

$$F = \frac{9}{5}C + 32$$

Task # 3: (SwingCounter)
Write a Java Swing GUI application as shown in the Figure. Each time the "Count" button is clicked, the counter value shall increase by 1.