Experiment Number: 01
Experiment Name: Write a JAVA Program to Display Image using JFrame

Theory: This Java program serves as a practical demonstration of building a basic graphical user interface (GUI) application using Swing, which is a library in Java for creating GUIs. Let's delve into the fundamental components and concepts utilized:
Swing Framework: Swing provides a rich set of GUI components for Java applications. These components, such as buttons, labels, and text fields, facilitate the creation of interactive user interfaces.
JFrame: In Swing, JFrame is a key component used to represent a top-level window in a GUI application. It serves as the container for other GUI components and manages their layout and behavior.
Layout Management: The program employs the FlowLayout layout manager. Layout managers in Swing are responsible for arranging components within a container. FlowLayout, specifically, organizes components in a left-to-right flow manner, automatically wrapping to the next line when the current line is filled.
ImageIcon: ImageIcon is a class in Java designed for representing image icons. It supports loading images from various sources such as files or URLs and is commonly used for displaying images in GUI applications.
JLabel: JLabel is a Swing component used for displaying text, images, or a combination of both. In this program, JLabel instances are employed to exhibit the images loaded using ImageIcon.
getResource() Method: This method is utilized to load resources like images from the classpath. It returns a URL object representing the location of the specified resource within the application's classpath.
Constructor: The constructor of the Image class initializes the GUI components. It configures the layout manager, loads images using ImageIcon, creates JLabel instances with these images, and adds them to the JFrame container.
Main Method: The main method serves as the entry point of the program. It instantiates the Image class, configures the JFrame's behavior (e.g., default close operation, visibility), adjusts the frame's size based on its contents, and sets the title of the frame.

Source Code:

```
import java.awt.FlowLayout;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
public class Image extends JFrame {
private ImageIcon image1;
private JLabel label1;
private ImageIcon image2;
private JLabel label2;
Image(){
setLayout(new FlowLayout());
image1 = new ImageIcon(getClass().getResource("ice.png"));
label1 = new JLabel(image1);
add(label1);
image2 = new ImageIcon(getClass().getResource("pust.png"));
label2 = new JLabel(image2);
add(label2);
}
public static void main(String args[]) {
Image gui = new Image();
gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
gui.setVisible(true);
gui.pack();
gui.setTitle("Image Program");
}
}
```

Output:

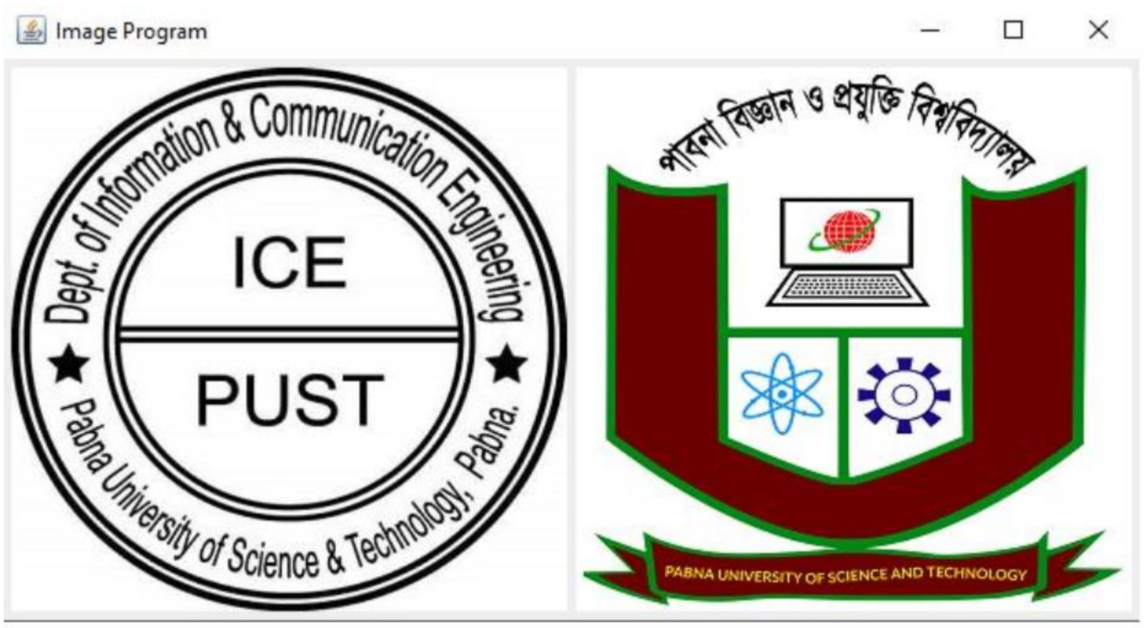


Image Program
ICE
PUST
Dept. of Information & Communication Engineering
Pabna University of Science & Technology, Pabna.
পাবনা বিজ্ঞান ও প্রযুক্তি বিশ্ববিদ্যালয়
PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY

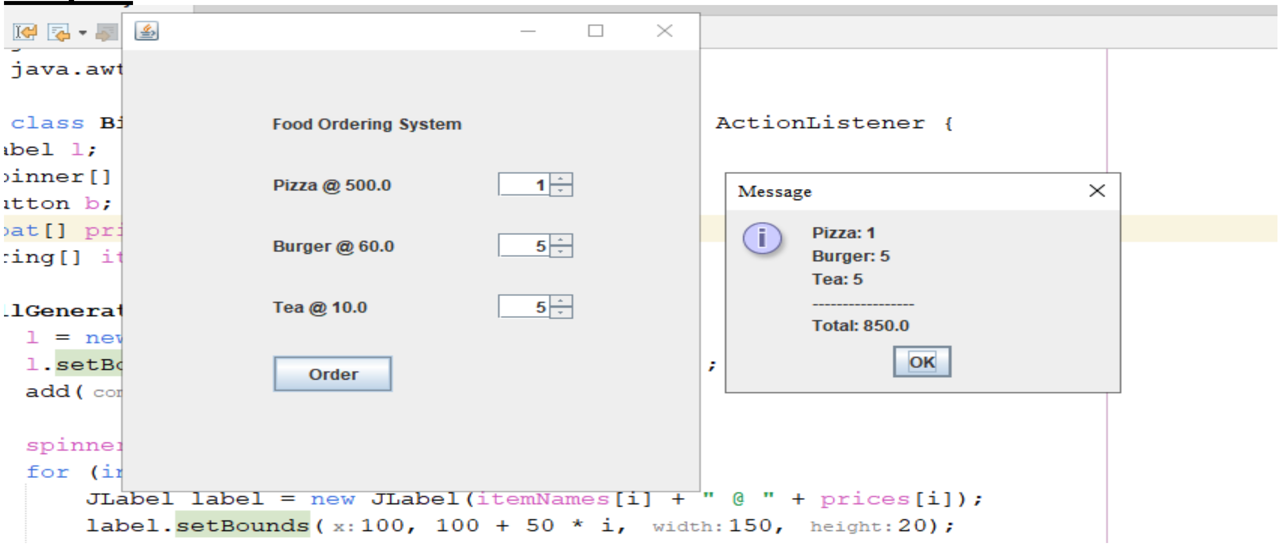
Experiment Number: **02**
Experiment Name: **Write a JAVA Program for generating Restaurant Bill**
Theory:
Swing Framework: The code leverages Swing, a GUI toolkit provided by Java, to develop interactive GUI applications.
JFrame: The BillGeneration class extends JFrame, indicating that it serves as the main window of the application, created using Swing's JFrame class.
Components: JLabel (l): Displays the title "Food Ordering System".
JCheckBox (cb1, cb2, cb3): Checkboxes enabling users to select food items like Pizza, Burger, and Tea.
JButton (b): Labeled "Order", this button facilitates placing the order.
Layout Management: The code adopts absolute positioning (null layout) to place components at specific coordinates on the frame. While simple, this approach is generally discouraged for complex GUIs due to potential issues with scalability and platform independence.
Event Handling: The class implements the ActionListener interface, defining the actionPerformed(ActionEvent e) method.

The "Order" button (b) is associated with an action listener (this), indicating that the BillGeneration class itself listens for events from this button.
Upon clicking the button, the actionPerformed method is invoked.
Action Event Handling: Within the actionPerformed method, the code determines which checkboxes are selected.
It calculates the total amount based on the selected items, with each chosen item contributing to the total. A message is constructed containing details of the selected items and the total amount. This message is displayed using a dialog box (JOptionPane.showMessageDialog).
Main Method: The main method instantiates an object of the BillGeneration class, initializing the GUI application.

Source Code:

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

public class BillGeneration extends JFrame implements ActionListener {
    JLabel l;
    JSpinner[] spinners;
    JButton b;
    float[] prices = {500, 60, 10};
    String[] itemNames = {"Pizza", "Burger", "Tea"};

    BillGeneration() {
        l = new JLabel("Food Ordering System");
        l.setBounds(100, 50, 300, 20);
        add(l);

        spinners = new JSpinner[itemNames.length];
        for (int i = 0; i < itemNames.length; i++) {
            JLabel label = new JLabel(itemNames[i] + " @ " + prices[i]);
            label.setBounds(100, 100 + 50 * i, 150, 20);
            add(label);

            spinners[i] = new JSpinner();
            spinners[i].setBounds(250, 100 + 50 * i, 50, 20);
            add(spinners[i]);
        }
        b = new JButton("Order");
        b.setBounds(100, 100 + 50 * itemNames.length, 80, 30);
        b.addActionListener(this);
        add(b);

        setSize(400, 400);
        setLayout(null);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
    public void actionPerformed(ActionEvent e) {
        float amount = 0;
        String msg = "";
        for (int i = 0; i < spinners.length; i++) {
            int quantity = (int) spinners[i].getValue();
            if (quantity > 0) {
                amount += prices[i] * quantity;
                msg += itemNames[i] + ": " + quantity + "\n";
            }
        }
        msg += "-----------------\n";
        JOptionPane.showMessageDialog(this, msg + "Total: " + amount);
    }
    public static void main(String[] args) {
        new BillGeneration();
    }
}
```

Output:



```
Food Ordering System

Pizza @ 500.0        1
Burger @ 60.0        5
Tea @ 10.0           5

Order
```

```
Message

Pizza: 1
Burger: 5
Tea: 5
-----------------
Total: 850.0

OK
```

```
java.awt
class Bi                          ActionListener {
abel l;
pinner[]
utton b;
oat[] pri
ring[] it
lGenerat
  l = new
  l.setBo
  add( co                         ;

  spinne
  for (i
      JLabel label = new JLabel(itemNames[i] + " @ " + prices[i]);
      label.setBounds( x: 100, 100 + 50 * i,  width: 150,  height: 20);
```

Experiment Number: **03**

Experiment Name: Write a JAVA Program to Create a Student form in GUI

Theory:

Swing Components:

- JFrame: Serves as the main window of the application.
- JPanel: A container component used to organize and hold other components.
- JLabel: Displays text to provide instructions or labels for input fields.
- JTextField: Allows users to input text or data.
- JButton: Triggers an action, such as saving data or submitting a form, when clicked.

Main Method:

- Creates an instance of JFrame named frame.

- Initializes a JPanel named panel and sets its layout to null, enabling custom positioning of components.
- Sets the size of the frame to 400x300 pixels.
- Sets the default close operation to exit the application when the frame is closed.
- Adds the panel to the frame.

Labels and TextFields:

- Creates labels (label1, label2, label3) for "Name", "Roll", and "Department", respectively.
- Positions these labels on the panel using setBounds.
- Creates text fields (userText1, userText2, userText3) for user input.
- Positions these text fields on the panel using setBounds.
- Sets default text in the text fields, which may serve as placeholders or initial values.

Button:

- Creates a button named "Save".
- Positions the button on the panel using setBounds.
- Registers an instance of the form class (which implements ActionListener) as the listener for button clicks.
- Adds the button to the panel.

Action Event Handling:

- The actionPerformed method is invoked when the "Save" button is clicked.
- It sets the text of a success label to "Saved Successfully", indicating that the data has been successfully saved.

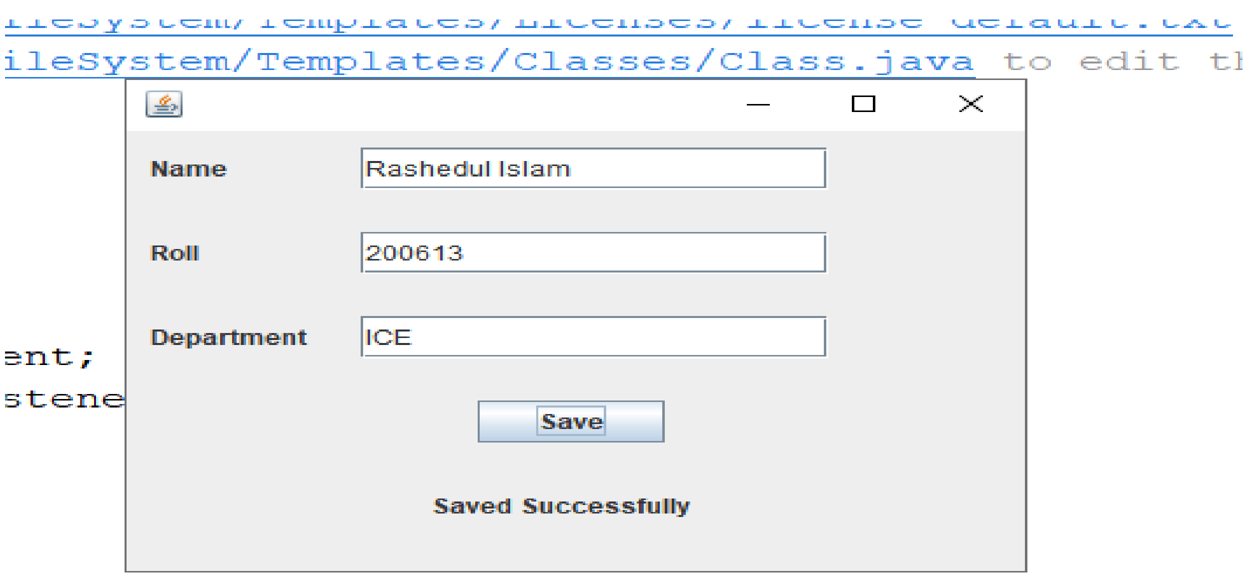
Visibility:

Sets the visibility of the frame to true, making it visible to the user and rendering the GUI components for interaction.

Source Code:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
public class form implements ActionListener {
private static JLabel success;
private static JFrame frame;
private static JLabel label1,label2,label3;
private static JPanel panel;
private static JButton button;
private static JTextField userText1, userText2, userText3;
public static void main(String[] args) {
frame = new JFrame();
panel = new JPanel();
frame.setSize(400, 300);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.add(panel);
panel.setLayout(null);
//Setting all Three Lebels
label1= new JLabel("Name");
label1.setBounds(10,10,80,25);
panel.add(label1);label2 = new JLabel("Roll");
label2.setBounds(10,60,80,25);
panel.add(label2);
label3 = new JLabel("Department");
label3.setBounds(10,110,80,25);
panel.add(label3);
//Creating all Textfields
userText1 = new JTextField("Enter Your Name");
userText1.setBounds(100,10,200,25);
panel.add(userText1);
JTextField userText2 = new JTextField("Enter Your Name");
userText2.setBounds(100,60,200,25);
panel.add(userText2);
JTextField userText3 = new JTextField("Enter Your Name");
userText3.setBounds(100,110,200,25);
panel.add(userText3);
button = new JButton("Save");
button.setBounds(150, 160, 80, 25);
button.addActionListener(new form());
panel.add(button);
success = new JLabel("");
success.setBounds(130,210,300,25);
panel.add(success);
frame.setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e) {
// TODO Auto-generated method stub
success.setText("Saved Successfully");
}
}
```

Output:


```
ileSystem/Templates/Licenses/license-default.txt
ileSystem/Templates/Classes/Class.java to edit tl
ent;
stene
```

Name: Rashedul Islam

Roll: 200613

Department: ICE

Save

Saved Successfully

<u>Experiment Number:</u> 04
<u>Experiment Name</u>: Write a JAVA Program to develop a simple calculator in GUI
<u>Theory:</u>
<u>Constructor:</u>

- Sets the title of the frame to "Calculator".
- Creates a panel with a grid layout to organize the buttons.
- Initializes an array of buttons for numbers (0-9) and adds them to the panel.
- Registers action listeners for each button.
- Creates additional buttons for arithmetic operations and clear, adds them to the panel, and registers action listeners for them.
- Creates a text field to display input and output.
- Adds the panel to the center and the text field to the top of the frame.
- Sets the visibility of the frame to true and sets its size.

Action Event Handling:

- Implements the ActionListener interface to handle button clicks.
- Defines the actionPerformed method to determine the source of the action event (i.e., which button was clicked).
- If the clear button is clicked, resets the input and output.
- If the equals button is clicked, evaluates the expression and displays the result.
- If an arithmetic operation button is clicked, stores the first operand and the operation.
- If a number button is clicked, appends the corresponding digit to the input.

Evaluation Method:

- Evaluates the expression based on the stored operation and operands.
- Updates the result accordingly.

Main Method:
Creates an instance of the calculator class to start the application.

Source Code:

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

public class Calculator extends JFrame implements ActionListener {
    JButton b10, b11, b12, b13, b14, b15, b16, b17, b18, b19; // Added b17, b18, b19 for %, (,
)
    JButton b[] = new JButton[10];
    double n1, n2, r;
    JTextField res;
    char op;
    boolean decimalClicked = false; // Track if decimal point button is clicked

    public Calculator() {
        super("Calculator");
        setLayout(new BorderLayout());
        JPanel p = new JPanel();
        p.setLayout(new GridLayout(5, 4)); // Increased grid layout to accommodate the new
buttons
        for (int i = 0; i <= 9; i++) {
            b[i] = new JButton(i + "");
            p.add(b[i]);
            b[i].addActionListener(this);
        }
        b10 = new JButton("+");
        p.add(b10);
        b10.addActionListener(this);
        b11 = new JButton("-");
        p.add(b11);
        b11.addActionListener(this);
        b12 = new JButton("*");
        p.add(b12);
        b12.addActionListener(this);
        b13 = new JButton("/");
        p.add(b13);
        b13.addActionListener(this);
        b14 = new JButton("=");
        p.add(b14);
        b14.addActionListener(this);
        b15 = new JButton("C");
        p.add(b15);
        b15.addActionListener(this);
        b16 = new JButton(".");
        p.add(b16);
        b16.addActionListener(this);
        b17 = new JButton("%"); // Button for percentage
        p.add(b17);
        b17.addActionListener(this);
        b18 = new JButton("("); // Button for opening bracket
        p.add(b18);
        b18.addActionListener(this);
        b19 = new JButton(")"); // Button for closing bracket
        p.add(b19);
        b19.addActionListener(this);
        res = new JTextField(20); // Increased number of columns to 20
        add(p, BorderLayout.CENTER);
        add(res, BorderLayout.NORTH);
        setVisible(true);
        setSize(400, 500); // Increased frame height to accommodate the new row of buttons
    }

    public void actionPerformed(ActionEvent ae) {
        JButton pb = (JButton) ae.getSource();
        if (pb == b15) {
            r = n1 = n2 = 0;
            res.setText("");
            decimalClicked = false; // Reset decimalClicked flag
        } else if (pb == b14) {
            n2 = Double.parseDouble(res.getText());
            eval();
            res.setText("" + r);
            decimalClicked = false; // Reset decimalClicked flag
        } else if (pb == b16) { // Process decimal point button
            if (!decimalClicked) {
              res.setText(res.getText() + ".");
              decimalClicked = true;
            }
        } else {
            boolean opf = false;
            if (pb == b10) {
              op = '+';
              opf = true;
            }
            if (pb == b11) {
              op = '-';
              opf = true;
            }
            if (pb == b12) {
              op = '*';
              opf = true;
            }
            if (pb == b13) {
```

```
                op = '/';
                opf = true;
            }
            if (!opf) {
                for (int i = 0; i < 10; i++) {
                    if (pb == b[i]) {
                        String t = res.getText();
                        t += i;
                        res.setText(t);
                    }
                }
            } else {
                n1 = Double.parseDouble(res.getText());
                res.setText("");
                decimalClicked = false; // Reset decimalClicked flag
            }
        }
    }
    void eval() {
        switch (op) {
            case '+':
                r = n1 + n2;
                break;
            case '-':
                r = n1 - n2;
                break;
            case '*':
                r = n1 * n2;
                break;
            case '/':
                r = n1 / n2;
                break;
        }
    }

    public static void main(String arg[]) {
        new Calculator();
    }
}
```

Output:

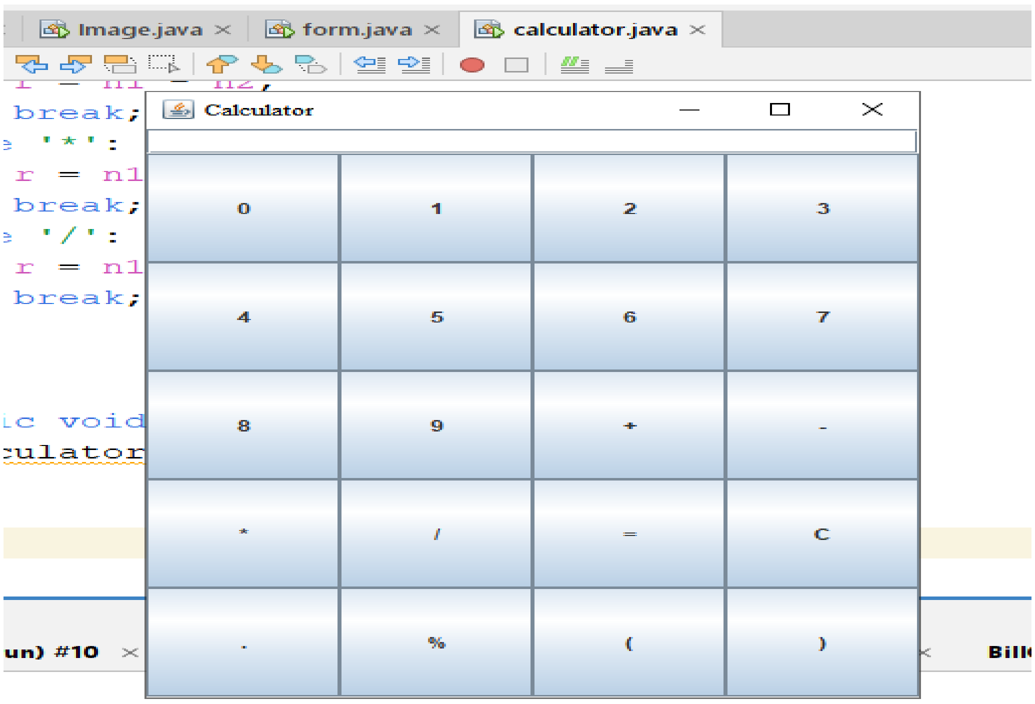


Image.java
form.java
calculator.java
Calculator
0
1
2
3
4
5
6
7
8
9
+
-
*
/
=
C
.
%
(
)
un) #10