Southeast University

Department of Computer Science and Engineering (CSE)

School of Sciences and Engineering

Semester: (Summer, Year: 2025)

LAB REPORT NO: **10**

Course Title: Introduction to Programming Language II (Java) Lab

Course Code: CSE282.2

Batch: 65

Lab Experiment Name: Designing a Graphical User Interface in JAVA.

Student Details

	Name	ID
1.	Md. Mahmud Hossain	2023200000799

Submission Date : **03-10-25**

Course Teacher's Name : **Dr. Mohammed Ashikur Rahman**

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

Lab Task 10: Designing a Graphical User Interface in JAVA.

PROBLEM:

1. Enhance the registration form by adding input validation to check if all required fields are filled before submission.
2. Implement functionality to save the registered user's data to a file/database.
3. Add additional fields to the registration form, such as address, phone number, and a file chooser for profile picture upload.
4. Implement a "Reset" button to clear all the form fields.

Solution:

1.

Problem Analysis:

In traditional systems, user registration is done manually or through basic command-line interfaces, which lack user interaction and input validation.

This program provides an interactive registration form that ensures all fields are filled before submission and stores user data efficiently into a file for record-keeping.

The program:

- Accepts Name, Email, Password, Address, Phone, and Profile Picture.
- Validates all input fields before allowing submission.
- Saves the data to **users.txt** with proper formatting.
- Allows the user to **reset** all fields or **choose a profile picture** from the local directory.

This approach demonstrates **GUI design, event-driven programming, and file handling** concepts in Java.

Background Theory:

1. **Java Swing (GUI Components):**
    - Swing provides a rich set of GUI components such as JFrame, JLabel, JTextField, JPasswordField, JButton, and JFileChooser.
    - GridLayout is used to organize UI components in rows and columns.
    - Event handling is done using ActionListener.
2. **Event Handling:**
    - The class implements ActionListener to respond to button clicks (Submit, Reset, and Choose File).
    - The actionPerformed() method detects which button was pressed and executes the corresponding logic.
3. **Input Validation:**
    - Before writing data, the program checks whether all fields are filled using String.isEmpty().
    - Displays error messages using JOptionPane dialogs for missing inputs.
4. **File Handling:**
    - Uses BufferedWriter and FileWriter to write registration data into users.txt.
    - Appends new user data without overwriting existing records.
    - Uses try-with-resources for safe and automatic file closing.

5. **JFileChooser:**
    - Enables users to select a profile picture from their computer.Appends new user data without overwriting existing records.
    - The selected image path is displayed as confirmation in the GUI.

Algorithm Design:

1. Initialize GUI components (labels, text fields, buttons).
2. Arrange all components using a GridLayout.
3. Add ActionListener to handle events for Submit, Reset, and Choose File buttons.
4. On Choose File, open JFileChooser and store the image path.
5. On Submit, validate that no field is empty.
6. If validation passes:

    - Write all data to users.txt using BufferedWriter.
    - Display confirmation via JOptionPane.

7. On Reset, clear all fields and reset status message

8. Handle possible exceptions (e.g., file errors).

Code:

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

public class RegistrationForm extends JFrame implements ActionListener {
    // Labels
    JLabel nameLabel, emailLabel, passLabel, addressLabel, phoneLabel, picLabel, statusLabel;
    JTextField nameField, emailField, phoneField, addressField;
    JPasswordField passField;
    JButton submitBtn, resetBtn, chooseBtn;
    String imagePath = "";

    public RegistrationForm() {
        setTitle("User Registration Form");
        setSize(450, 400);
        setLayout(new GridLayout(8, 2, 10, 10));
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        // Initialize components
        nameLabel = new JLabel("Name:");
        emailLabel = new JLabel("Email:");
        passLabel = new JLabel("Password:");
        addressLabel = new JLabel("Address:");
        phoneLabel = new JLabel("Phone:");
        picLabel = new JLabel("Profile Picture:");
        statusLabel = new JLabel("");

        nameField = new JTextField();
        emailField = new JTextField();
        passField = new JPasswordField();
        addressField = new JTextField();
        phoneField = new JTextField();

        chooseBtn = new JButton("Choose File");
        chooseBtn.addActionListener(this);

        submitBtn = new JButton("Submit");
        resetBtn = new JButton("Reset");
        submitBtn.addActionListener(this);
        resetBtn.addActionListener(this);
```

```
        // Add components to frame
        add(nameLabel); add(nameField);
        add(emailLabel); add(emailField);
        add(passLabel); add(passField);
        add(addressLabel); add(addressField);
        add(phoneLabel); add(phoneField);
        add(picLabel); add(chooseBtn);
        add(submitBtn); add(resetBtn);
        add(statusLabel);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == chooseBtn) {
            JFileChooser fileChooser = new JFileChooser();
            int result = fileChooser.showOpenDialog(this);
            if (result == JFileChooser.APPROVE_OPTION) {
                imagePath = fileChooser.getSelectedFile().getAbsolutePath();
                statusLabel.setText("Selected: " + fileChooser.getSelectedFile().getName());
            }
        } else if (e.getSource() == submitBtn) {
            String name = nameField.getText().trim();
            String email = emailField.getText().trim();
            String password = new String(passField.getPassword()).trim();
            String address = addressField.getText().trim();
            String phone = phoneField.getText().trim();

            //  Input validation
            if (name.isEmpty() || email.isEmpty() || password.isEmpty() ||
                address.isEmpty() || phone.isEmpty() || imagePath.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please fill all fields!", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }

            //  Save to file
            try (BufferedWriter bw = new BufferedWriter(new FileWriter("users.txt", true))) {
                bw.write("Name: " + name);
                bw.newLine();
                bw.write("Email: " + email);
                bw.newLine();
                bw.write("Password: " + password);
```

```
            bw.newLine();
            bw.write("Address: " + address);
            bw.newLine();
            bw.write("Phone: " + phone);
            bw.newLine();
            bw.write("Profile Picture: " + imagePath);
            bw.newLine();
            bw.write("----------------------------");
            bw.newLine();

            JOptionPane.showMessageDialog(this, "Registration Successful!");
            statusLabel.setText("User registered successfully!");
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(this, "Error saving data!", "Error",
JOptionPane.ERROR_MESSAGE);
        }
      } else if (e.getSource() == resetBtn) {
        //  Clear all fields
        nameField.setText("");
        emailField.setText("");
        passField.setText("");
        addressField.setText("");
        phoneField.setText("");
        imagePath = "";
        statusLabel.setText("Form reset successfully!");
      }
   }

   public static void main(String[] args) {
      new RegistrationForm();
   }
}
```

Output:

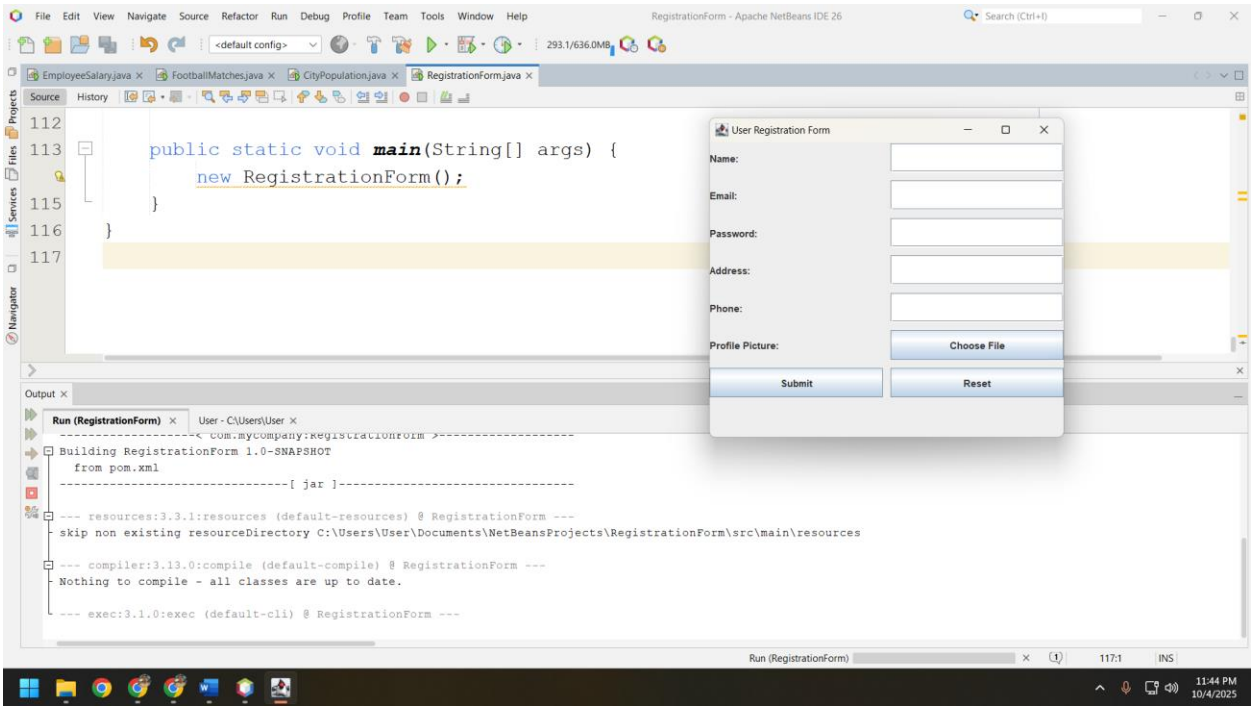

Figure 1: Output