**Lab 1: Create a program to accept 10 integers from the user, store them in an array, and calculate their sum and average using a for-each loop.**

import java.util.Scanner;

public class lab1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int[] numbers = new int[10];

        int sum = 0;

        System.out.println("Enter 10 integers:");

        for (int i = 0; i < numbers.length; i++) {

            numbers[i] = sc.nextInt();

        }

        for (int num : numbers) {

            sum += num;

        }

        double average = sum / 10.0;

        System.out.println("Sum: " + sum);

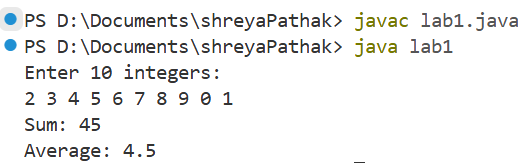
        System.out.println("Average: " + average);

        sc.close();

    }

}

Output:



**Lab 2: Write a program with an interface Shape containing methods area() and perimeter(). Implement it in two classes, Circle and Rectangle. Include an inner class to calculate diagonal for the Rectangle.**

import java.util.Scanner;

interface Shape {

    double area();

    double perimeter();

}

class Circle implements Shape {

    private double radius;

    public Circle(double radius) {

        this.radius = radius;

    }

    @Override

    public double area() {

        return Math.PI \* radius \* radius;

    }

    @Override

    public double perimeter() {

        return 2 \* Math.PI \* radius;

    }

}

class Rectangle implements Shape {

    private double length;

    private double width;

    public Rectangle(double length, double width) {

        this.length = length;

        this.width = width;

    }

    @Override

    public double area() {

        return length \* width;

    }

    @Override

    public double perimeter() {

        return 2 \* (length + width);

    }

    class Diagonal {

        public double calculateDiagonal() {

            return Math.sqrt(length \* length + width \* width);

        }

    }

}

public class lab2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter radius of the circle: ");

        double radius = sc.nextDouble();

        Circle circle = new Circle(radius);

        System.out.println("Circle Area: " + circle.area());

        System.out.println("Circle Perimeter: " + circle.perimeter());

        // Rectangle input

        System.out.print("Enter length of rectangle: ");

        double length = sc.nextDouble();

        System.out.print("Enter width of rectangle: ");

        double width = sc.nextDouble();

        Rectangle rectangle = new Rectangle(length, width);

        System.out.println("Rectangle Area: " + rectangle.area());

        System.out.println("Rectangle Perimeter: " + rectangle.perimeter());

        // Calculate diagonal using inner class

        Rectangle.Diagonal diagonal = rectangle.new Diagonal();

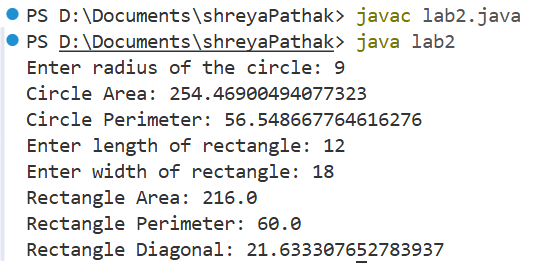
        System.out.println("Rectangle Diagonal: " + diagonal.calculateDiagonal());

        sc.close();

    }

}

Output:



**Lab 3: Design a base class Employee with attributes name and salary, and a derived class Manager with additional attributes department. Override a method displayDetails() in both classes.**

class Employee {

    protected String name;

    protected double salary;

    public Employee(String name, double salary) {

        this.name = name;

        this.salary = salary;

    }

    public void displayDetails() {

        System.out.println("Employee Name: " + name);

        System.out.println("Employee Salary: " + salary);

    }

}

class Manager extends Employee {

    private String department;

    public Manager(String name, double salary, String department) {

super(name, salary);

        this.department = department;}

    @Override

    public void displayDetails() {

        System.out.println("Manager Name: " + name);

        System.out.println("Manager Salary: " + salary);

        System.out.println("Department: " + department);

    }

}

public class lab3 {

    public static void main(String[] args) {

        Employee emp = new Employee("Shreya", 50000);

        emp.displayDetails();

        System.out.println();

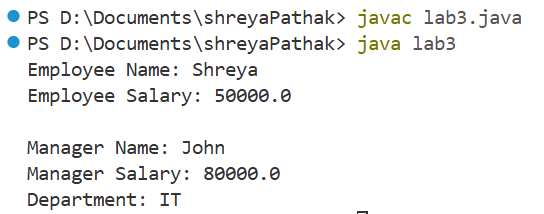
        Manager mgr = new Manager("John", 80000, "IT");

        mgr.displayDetails();

    }

}

Output:



**Lab 4: Write a program that prompts the user to enter two integers and performs division. Handle exceptions for invalid inputs (e.g., non-numeric input) and division by zero.**

import java.util.Scanner;

public class lab4 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        try {

            System.out.print("Enter the first integer: ");

            int num1 = sc.nextInt();

            System.out.print("Enter the second integer: ");

            int num2 = sc.nextInt();

            int result = num1 / num2;

            System.out.println("Result: " + num1 + " / " + num2 + " = " + result);

        }

        catch (ArithmeticException e) {

            System.out.println("Error: Division by zero is not allowed.");

        }

        catch (java.util.InputMismatchException e) {

            System.out.println("Error: Invalid input. Please enter integers only.");

        }

        finally {

            sc.close();

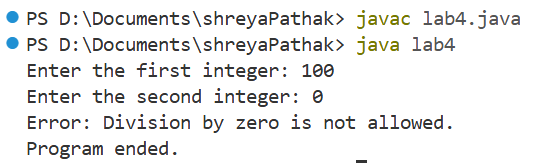
            System.out.println("Program ended.");

        }

    }

}

Output:



**Lab 5: Create a program where multiple threads update a shared counter. Use synchronization to ensure thread-safe operations.**

class Counter {

    private int count = 0;

    public synchronized void increment() {

        count++;

    }

    public int getCount() {

        return count;

    }

}

class CounterThread extends Thread {

    private Counter counter;

    public CounterThread(Counter counter) {

        this.counter = counter;

    }

    @Override

    public void run() {

        for (int i = 0; i < 2000; i++) {

            counter.increment();

        }

    }

}

public class lab5 {

    public static void main(String[] args) {

        Counter counter = new Counter();

        Thread t1 = new CounterThread(counter);

        Thread t2 = new CounterThread(counter);

        Thread t3 = new CounterThread(counter);

        t1.start();

        t2.start();

        t3.start();

        try {

            t1.join();

            t2.join();

            t3.join();

        } catch (InterruptedException e) {

            e.printStackTrace();

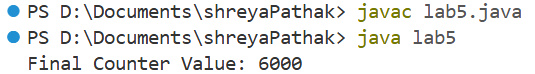
        }

        System.out.println("Final Counter Value: " + counter.getCount());

    }

}

Output:



**Lab 6: Write a program to create three threads with different priorities and observe their execution order.**

class MyThread extends Thread {

    private String name;

    public MyThread(String name) {

        this.name = name;

    }

    @Override

    public void run() {

    for (int i = 1; i <= 5; i++) {

    System.out.println(name + " is running, iteration: " + i);

            try {

                Thread.sleep(100);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }}

}

public class lab6 {

    public static void main(String[] args) {

        MyThread t1 = new MyThread("Thread 1");

        MyThread t2 = new MyThread("Thread 2");

        MyThread t3 = new MyThread("Thread 3");

        t1.setPriority(Thread.MIN\_PRIORITY);

        t2.setPriority(Thread.NORM\_PRIORITY);

        t3.setPriority(Thread.MAX\_PRIORITY);

        t1.start();

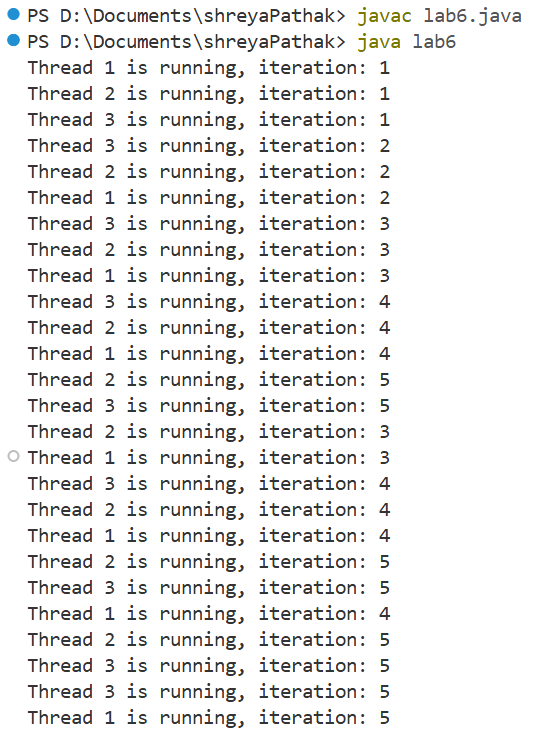
        t2.start();

        t3.start();

    }

}

Output:



**Lab 7: Write a program to copy the contents of one file to another using byte streams.**

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

public class lab7 {

    public static void main(String[] args) {

        String sourceFile = "input.txt";

        String destFile = "output.txt";

        FileInputStream fis = null;

        FileOutputStream fos = null;

        try {

            fis = new FileInputStream(sourceFile);

            fos = new FileOutputStream(destFile);

            byte[] buffer = new byte[1024];

            int bytesRead;

            while ((bytesRead = fis.read(buffer)) != -1) {

                fos.write(buffer, 0, bytesRead);

            }

            System.out.println("File copied successfully.");

        } catch (IOException e) {

            System.out.println("Error: " + e.getMessage());

        } finally {

            try {

                if (fis != null) fis.close();

                if (fos != null) fos.close();

            } catch (IOException e) {

                e.printStackTrace();

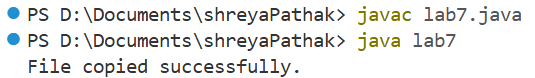
            }

        }

    }

}

Output:



Source File: Destination File:

**Lab 8: Create a program to read from a text file and write its content to another file, line by line.**

import java.io.\*;

public class lab8 {

    public static void main(String[] args) {

        String sourceFile = "input1.txt";

        String destinationFile = "output1.txt";

        BufferedReader br = null;

        BufferedWriter bw = null;

        try {

            br = new BufferedReader(new FileReader(sourceFile));

            bw = new BufferedWriter(new FileWriter(destinationFile));

            String line;

            while ((line = br.readLine()) != null) {

                bw.write(line);

                bw.newLine();

            }

            System.out.println("File copied line by line successfully.");

        } catch (IOException e) {

            System.out.println("Error: " + e.getMessage());

        } finally {

            try {

                if (br != null) br.close();

                if (bw != null) bw.close();

            } catch (IOException e) {

                e.printStackTrace();

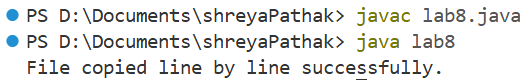
            }

        }

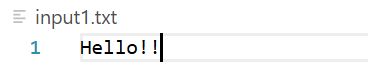
    }

}

Output:



Source File: Destination File:



**Lab 9: Write a program to demonstrate the use of RandomAccessFile by writing data to a file at specific positions and reading it back.**

import java.io.IOException;

import java.io.RandomAccessFile;

public class lab9 {

    public static void main(String[] args) {

        String filename = "randomfile.txt";

        RandomAccessFile raf = null;

        try {

            raf = new RandomAccessFile(filename, "rw");

            raf.seek(0);

            raf.writeUTF("Hello");

            raf.seek(20);

            raf.writeUTF("World");

            raf.seek(10);

            raf.writeUTF("Java");

            System.out.println("Data written successfully.");

            raf.seek(0);

            System.out.println("Data at position 0: " + raf.readUTF());

            raf.seek(10);

            System.out.println("Data at position 10: " + raf.readUTF());

            raf.seek(20);

            System.out.println("Data at position 20: " + raf.readUTF());

        } catch (IOException e) {

            e.printStackTrace();

        } finally {

            try {

                if (raf != null) raf.close();

            } catch (IOException e) {

                e.printStackTrace();

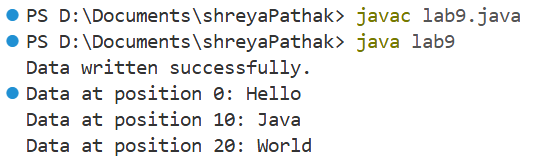
            }

        }

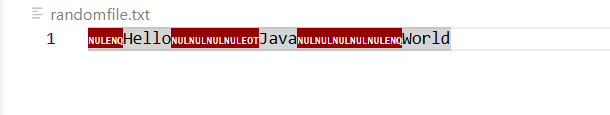
    }

}

Output:



Random File:



**Lab 10: Create a Java Applet that displays a "Hello, World!" message.**

**lab10.java**

import java.applet.Applet;

import java.awt.Graphics;

public class Lab\_10 extends Applet {

    public void paint(Graphics g) {

        g.drawString("Hello, World!", 50, 50);

    }

}

**lab10.html**

<html>

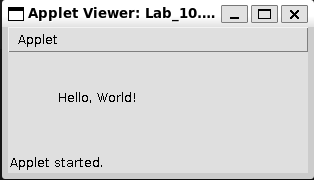
  <body>

    <applet code="Lab\_10.class" width="300" height="100"></applet>

  </body>

</html>

Output:



**Lab 11: Create a Swing application with components like JButton, JLabel, and JTextField added to a JPanel, which is then added to a JFrame.**

import javax.swing.\*;

public class lab11 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Simple Form");

        frame.setSize(400, 150);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JPanel panel = new JPanel();

        JLabel label = new JLabel("Enter your name:");

        JTextField textField = new JTextField(15);

        JButton button = new JButton("Submit");

        panel.add(label);

        panel.add(textField);

        panel.add(button);

        frame.add(panel);

        // Corrected action listener

        button.addActionListener(e -> {

            String name = textField.getText();

            if (!name.isEmpty()) {

                JOptionPane.showMessageDialog(frame, "Hello, " + name + "!");

            } else {

                JOptionPane.showMessageDialog(frame, "Please enter your name!");

            }

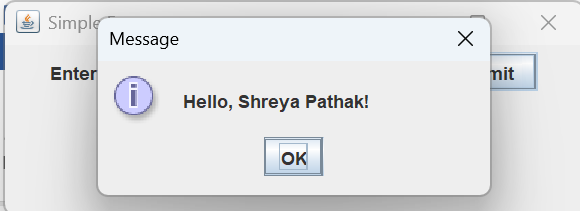
        });

        frame.setVisible(true);

    }

}

Output:



**Lab 12: Create a GUI application to demonstrate FlowLayout by adding buttons in a flow.**

import javax.swing.\*;

import java.awt.\*;

public class lab12 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("FlowLayout Demo");

        frame.setSize(400, 150);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JPanel panel = new JPanel(new FlowLayout());

        for (int i = 1; i <= 5; i++) {

            panel.add(new JButton("Button " + i));

        }

        frame.add(panel);

        frame.setVisible(true);

    }

}

Output:





**Lab 13: Design a calculator-like GUI using BorderLayout with buttons at different positions (NORTH, SOUTH, EAST, WEST, and CENTER).**

import javax.swing.\*;

import java.awt.\*;

public class lab13 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("BorderLayout Demo");

        frame.setSize(400, 300);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setLayout(new BorderLayout());

        frame.add(new JButton("North"), BorderLayout.NORTH);

        frame.add(new JButton("South"), BorderLayout.SOUTH);

        frame.add(new JButton("East"), BorderLayout.EAST);

        frame.add(new JButton("West"), BorderLayout.WEST);

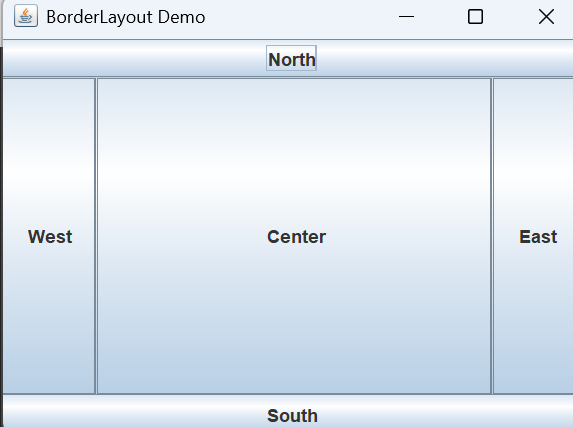
        frame.add(new JButton("Center"), BorderLayout.CENTER);

        frame.setVisible(true);

    }

}

Output:



**Lab 14: Create a GUI with a GridLayout that displays a 3x3 grid of buttons labeled 1 to 9.**

import javax.swing.\*;

import java.awt.\*;

public class lab14 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("GridLayout Demo");

        frame.setSize(300, 300);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JPanel panel = new JPanel(new GridLayout(3, 3));

        for (int i = 1; i <= 9; i++) {

            panel.add(new JButton(i + ""));

        }

        frame.add(panel);

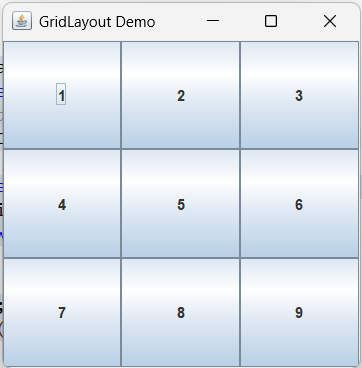
        frame.setVisible(true);

    }

}

Output:





**Lab 15: Implement a GridBagLayout to arrange components with varying sizes and positions.**

import javax.swing.\*;

import java.awt.\*;

public class lab15 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("GridBagLayout Demo");

        frame.setSize(400, 300);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JPanel panel = new JPanel(new GridBagLayout());

        GridBagConstraints gbc = new GridBagConstraints();

        JButton b1 = new JButton("Button 1");

        gbc.gridx = 0;

        gbc.gridy = 0;

        panel.add(b1, gbc);

        JButton b2 = new JButton("Button 2");

        gbc.gridx = 2;

        gbc.gridy = 0;

        panel.add(b2, gbc);

        JButton b3 = new JButton("Button 3");

        gbc.gridx = 0;

        gbc.gridy = 1;

        panel.add(b3, gbc);

        JButton b4 = new JButton("Button 4");

        gbc.gridx = 1;

        gbc.gridy = 1;

        panel.add(b4, gbc);

        JButton b5 = new JButton("Button 5");

        gbc.gridx = 2;

        gbc.gridy = 1;

        panel.add(b5, gbc);

        frame.add(panel);

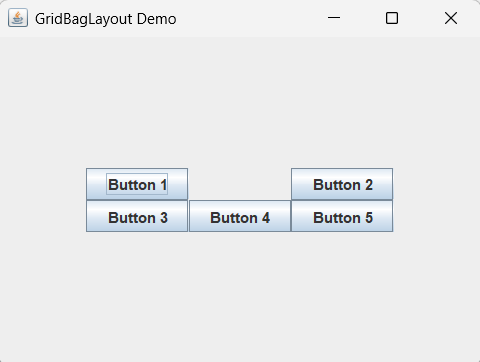
        frame.setVisible(true);

    }

}

Output:





**Lab 16: Create a form with JTextField, JPasswordField, and JTextArea. Add validation to ensure non-empty inputs.**

import javax.swing.\*;

public class lab16 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("User Form");

        frame.setSize(400, 300);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JPanel panel = new JPanel();

        panel.setLayout(null);

        JLabel nameLabel = new JLabel("Username:");

        nameLabel.setBounds(20, 20, 100, 25);

        JTextField nameField = new JTextField();

        nameField.setBounds(130, 20, 200, 25);

        JLabel passLabel = new JLabel("Password:");

        passLabel.setBounds(20, 60, 100, 25);

        JPasswordField passField = new JPasswordField();

        passField.setBounds(130, 60, 200, 25);

        JLabel addressLabel = new JLabel("Address:");

        addressLabel.setBounds(20, 100, 100, 25);

        JTextArea addressArea = new JTextArea();

        JScrollPane scrollPane = new JScrollPane(addressArea);

        scrollPane.setBounds(130, 100, 200, 80);

        JButton submitBtn = new JButton("Submit");

        submitBtn.setBounds(150, 200, 100, 30);

        submitBtn.addActionListener(e -> {

            String username = nameField.getText().trim();

            String password = new String(passField.getPassword()).trim();

            String address = addressArea.getText().trim();

            if (username.isEmpty() || password.isEmpty() || address.isEmpty()) {

                JOptionPane.showMessageDialog(frame, "All fields are required!", "Error", JOptionPane.ERROR\_MESSAGE);

            } else {

                JOptionPane.showMessageDialog(frame, "Form submitted successfully!", "Success", JOptionPane.INFORMATION\_MESSAGE);

            }

        });

        panel.add(nameLabel);

        panel.add(nameField);

        panel.add(passLabel);

        panel.add(passField);

        panel.add(addressLabel);

        panel.add(scrollPane);

        panel.add(submitBtn);

        frame.add(panel);

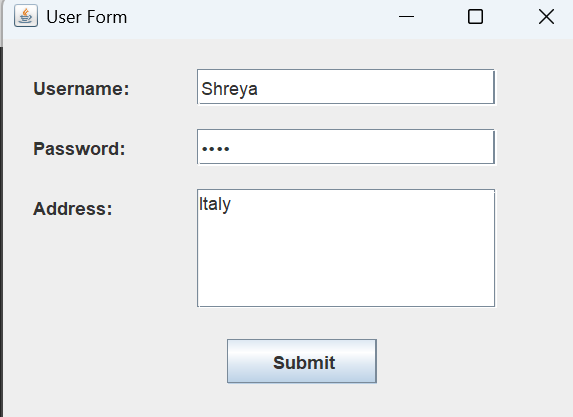
        frame.setVisible(true);

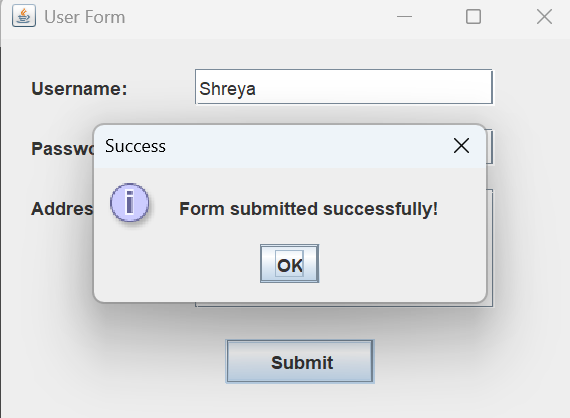
    }

}

Output:







**Lab 17: Create a GUI with a JCheckBox and JRadioButton to select favorite programming languages. Show the selected options on a JLabel.**

import javax.swing.\*;

public class lab17 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Favorite Programming Languages");

        frame.setSize(400, 300);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JPanel panel = new JPanel();

        panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));

        frame.add(panel);

        JLabel languageLabel = new JLabel("Select your favorite programming languages:");

        JCheckBox javaBox = new JCheckBox("Java");

        JCheckBox pythonBox = new JCheckBox("Python");

        JCheckBox cppBox = new JCheckBox("C++");

        JLabel levelLabel = new JLabel("Select your proficiency level:");

        JRadioButton beginner = new JRadioButton("Beginner");

        JRadioButton advanced = new JRadioButton("Advanced");

        ButtonGroup levelGroup = new ButtonGroup();

        levelGroup.add(beginner);

        levelGroup.add(advanced);

        JLabel resultLabel = new JLabel("Your selection will appear here");

        JButton showBtn = new JButton("Show Selection");

        showBtn.addActionListener(e -> {

            StringBuilder result = new StringBuilder("You selected: Languages: ");

            if (javaBox.isSelected()) result.append("Java ");

            if (pythonBox.isSelected()) result.append("Python ");

            if (cppBox.isSelected()) result.append("C++ ");

            result.append("| Level: ");

            if (beginner.isSelected()) result.append("Beginner");

            else if (advanced.isSelected()) result.append("Advanced");

            else result.append("None");

            resultLabel.setText(result.toString());

        });

        panel.add(languageLabel);

        panel.add(javaBox);

        panel.add(pythonBox);

        panel.add(cppBox);

        panel.add(levelLabel);

        panel.add(beginner);

        panel.add(advanced);

        panel.add(showBtn);

        panel.add(resultLabel);

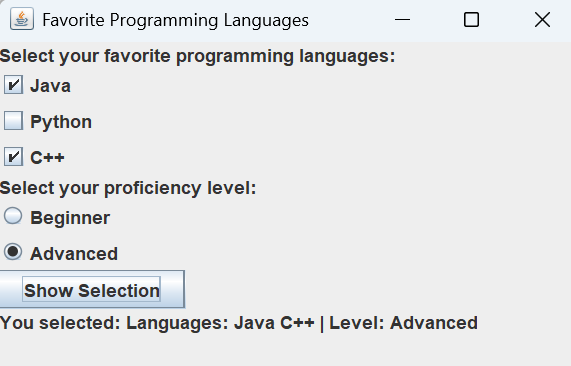
        frame.setVisible(true);

    }

}

Output:





**Lab 18: Create a menu bar with menus for "File" and "Edit." Add menu items such as "Open," "Save," and "Exit." Enable and disable them programmatically.**

import javax.swing.\*;

public class lab18 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Menu Example");

        frame.setSize(300, 200);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JMenuBar menuBar = new JMenuBar();

        JMenu fileMenu = new JMenu("File");

        JMenuItem open = new JMenuItem("Open");

        JMenuItem save = new JMenuItem("Save");

        JMenuItem exit = new JMenuItem("Exit");

        fileMenu.add(open);

        fileMenu.add(save);

        fileMenu.add(exit);

        JMenu editMenu = new JMenu("Edit");

        JMenuItem cut = new JMenuItem("Cut");

        JMenuItem copy = new JMenuItem("Copy");

        editMenu.add(cut);

        editMenu.add(copy);

        menuBar.add(fileMenu);

        menuBar.add(editMenu);

        frame.setJMenuBar(menuBar);

        open.setEnabled(true);   // enabled

        save.setEnabled(false);  // disabled

        exit.addActionListener(e -> System.exit(0));

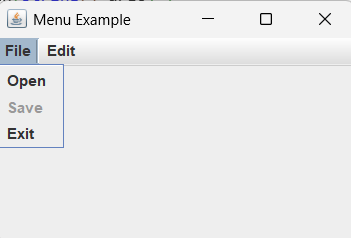
        frame.setVisible(true);

    }

}

Output:





**Lab 19: Add a button to open a color chooser dialog. Change the background color of a panel based on the selected color.**

import javax.swing.\*;

import java.awt.\*;

public class lab19 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Color Chooser Example");

        frame.setSize(400, 300);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JPanel panel = new JPanel(); // panel whose background will change

        JButton btn = new JButton("Choose Color");

        btn.addActionListener(e -> {

            Color color = JColorChooser.showDialog(frame, "Pick a Color", panel.getBackground());

            if (color != null) {

                panel.setBackground(color);

            }

        });

        frame.add(panel, BorderLayout.CENTER);

        frame.add(btn, BorderLayout.SOUTH);

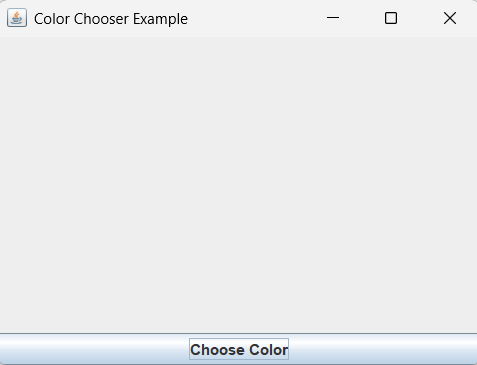
        frame.setVisible(true);

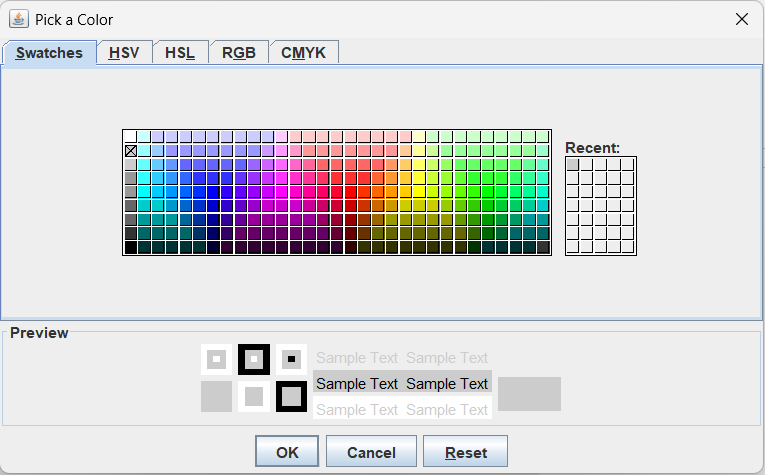
    }

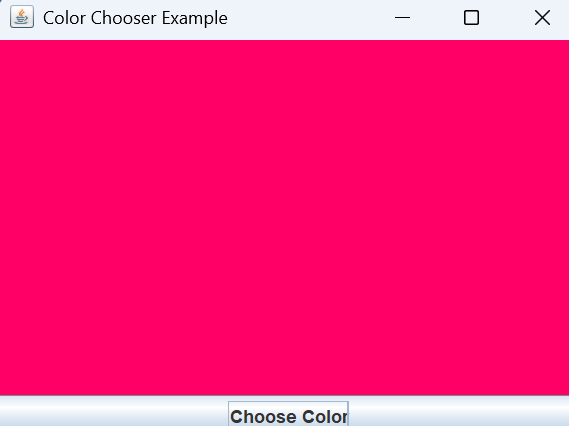
}

Output:









**Lab 20**: **Implement a mouse listener using an adapter class to handle mouse events like mouseClicked() and mouseEntered().**

import javax.swing.\*;

import java.awt.event.\*;

public class lab20 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Mouse Listener Example");

        frame.setSize(400, 300);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JLabel label = new JLabel("Click or hover here!", SwingConstants.CENTER);

        frame.add(label);

        label.addMouseListener(new MouseAdapter() {

            @Override

            public void mouseClicked(MouseEvent e) {

                label.setText("Mouse Clicked!");

            }

            @Override

            public void mouseEntered(MouseEvent e) {

                label.setText("Mouse Entered!");

            }

        });

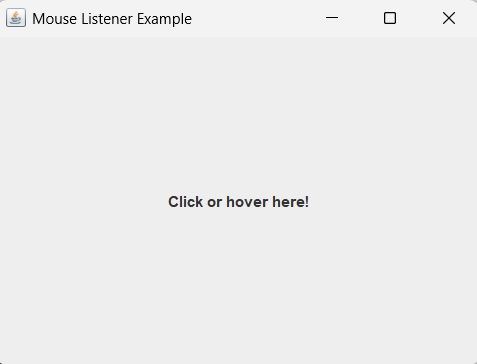
        frame.setVisible(true);

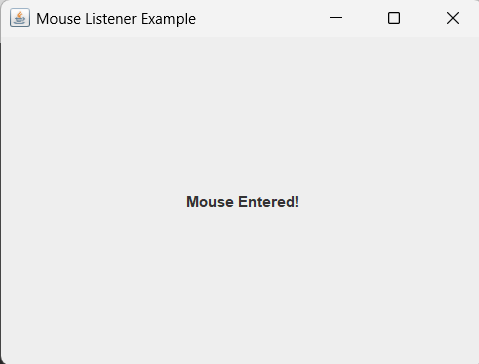
    }

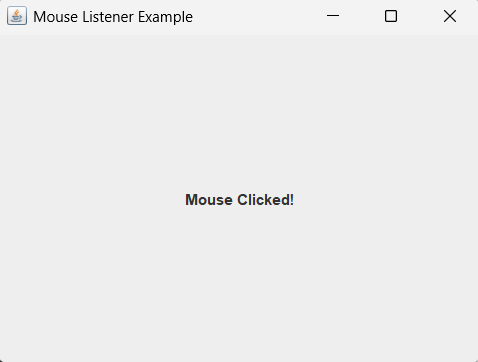
}

Output:









**Lab 21: Write a program that shows a confirmation dialog when the user attempts to close the window.**

import javax.swing.\*;

import java.awt.event.\*;

public class lab21 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Confirm on Close");

        frame.setSize(300, 200);

        frame.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

        frame.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {

                int result = JOptionPane.showConfirmDialog(

                        frame, "Exit the program?", "Confirm",

                        JOptionPane.YES\_NO\_OPTION);

                if (result == JOptionPane.YES\_OPTION) {

                    frame.dispose();

                }

            }

        });

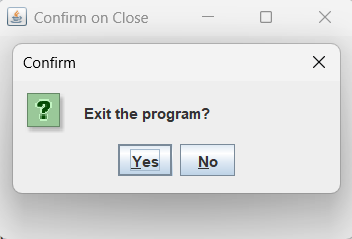
        frame.setVisible(true);

    }

}

Output:





**Lab 22: Create a GUI with checkboxes for hobbies. Display the selected hobbies in real time.**

import javax.swing.\*;

import java.awt.event.\*;

public class lab22 {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Select Hobbies");

        frame.setSize(400, 300);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        JPanel panel = new JPanel();

        panel.setLayout(new BoxLayout(panel, BoxLayout.Y\_AXIS));

        JLabel hobbiesLabel = new JLabel("Choose your hobbies:");

        JCheckBox readingBox = new JCheckBox("Reading");

        JCheckBox travelingBox = new JCheckBox("Traveling");

        JCheckBox musicBox = new JCheckBox("Music");

        JLabel resultLabel = new JLabel("Selected hobbies: None");

        panel.add(hobbiesLabel);

        panel.add(readingBox);

        panel.add(travelingBox);

        panel.add(musicBox);

        panel.add(resultLabel);

        frame.add(panel);

        ActionListener updateLabel = e -> {

            StringBuilder hobbies = new StringBuilder();

            if (readingBox.isSelected()) hobbies.append("Reading, ");

            if (travelingBox.isSelected()) hobbies.append("Traveling, ");

            if (musicBox.isSelected()) hobbies.append("Music, ");

            if (hobbies.length() > 0) {

                hobbies.setLength(hobbies.length() - 2); // Remove trailing comma and space

                resultLabel.setText("Selected hobbies: " + hobbies);

            } else {

                resultLabel.setText("Selected hobbies: None");

            }

        };

        readingBox.addActionListener(updateLabel);

        travelingBox.addActionListener(updateLabel);

        musicBox.addActionListener(updateLabel);

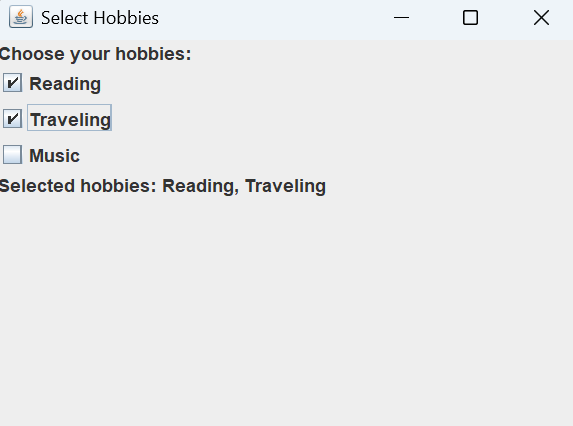
        frame.setVisible(true);

    }

}

Output:





**Lab 23: Write a program to connect to a MySQL/PostgreSQL database and display the database metadata.**

import java.sql.\*;

public class lab23 {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/demo\_db";

        String user = "root";

        String password = "oracle123";

        Connection conn = null;

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

            conn = DriverManager.getConnection(url, user, password);

            DatabaseMetaData meta = conn.getMetaData();

            System.out.println("Database Product: " + meta.getDatabaseProductName());

            System.out.println("Database Version: " + meta.getDatabaseProductVersion());

            System.out.println("Driver Name: " + meta.getDriverName());

            System.out.println("Driver Version: " + meta.getDriverVersion());

        } catch (ClassNotFoundException e) {

            e.printStackTrace();

        } catch (SQLException e) {

            e.printStackTrace();

        } finally {

            try {

                if (conn != null) conn.close();

            } catch (SQLException ex) {

                ex.printStackTrace();

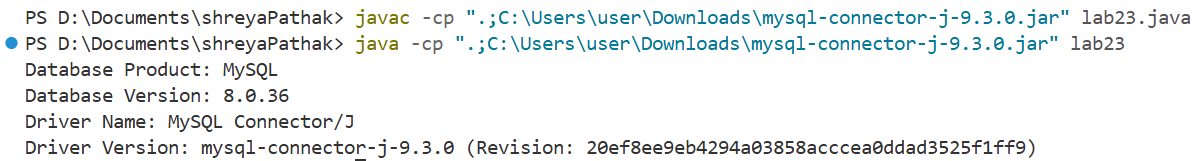
            }

        }

    }

}

Output:



**Lab 24: Write a program to insert, update, and delete records from the Students table.**

import java.sql.\*;

public class lab24 {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/demo\_db";

        String user = "root";

        String password = "oracle123";

        Connection con = null;

        Statement stmt = null;

        try {

            con = DriverManager.getConnection(url, user, password);

            stmt = con.createStatement();

            String insertData = "INSERT INTO Students (id, name, email) VALUES (1, 'Shreya', 'shreya@gmail.com')";

            stmt.executeUpdate(insertData);

            System.out.println("Data inserted.");

            String updateData = "UPDATE Students SET name = 'Pathak' WHERE id = 1";

            stmt.executeUpdate(updateData);

            System.out.println("Data updated.");

            String deleteData = "DELETE FROM Students WHERE id = 1";

            stmt.executeUpdate(deleteData);

            System.out.println("Operations completed.");

        } catch (SQLException e) {

            e.printStackTrace();

        } finally {

            try {

                if (stmt != null) stmt.close();

                if (con != null) con.close();

            } catch (SQLException ex) {

                ex.printStackTrace();

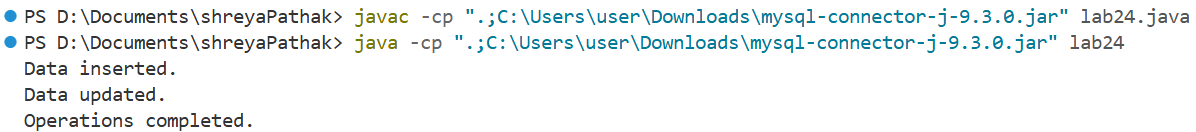
            }

        }

    }

}

Output:



**Lab 25: Create a program that fetches student records in both forward and reverse directions using scrollable result sets.**

import java.sql.\*;

public class lab25 {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/demo\_db";

        String user = "root";

        String password = "oracle123";

        Connection con = null;

        Statement stmt = null;

        ResultSet rs = null;

        try {

            con = DriverManager.getConnection(url, user, password);

            stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,

                    ResultSet.CONCUR\_READ\_ONLY);

            rs = stmt.executeQuery("SELECT \* FROM Students");

            System.out.println("Forward:");

            while (rs.next()) {

                System.out.println(rs.getInt("id") + " - " + rs.getString("name"));

            }

            System.out.println("\nReverse:");

            rs.afterLast();

            while (rs.previous()) {

                System.out.println(rs.getInt("id") + " - " + rs.getString("name"));

            }

        } catch (SQLException e) {

            e.printStackTrace();

        } finally {

            try {

                if (rs != null) rs.close();

                if (stmt != null) stmt.close();

                if (con != null) con.close();

            } catch (SQLException ex) {

                ex.printStackTrace();

            }

        }}

}

Output:



**Lab 26: Write a program to perform a transaction with multiple SQL statements. Rollback the transaction in case of an error.**

import java.sql.\*;

public class lab26 {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/demo\_db";

        String user = "root";

        String password = "oracle123";

        Connection con = null;

        Statement stmt = null;

        try {

            con = DriverManager.getConnection(url, user, password);

            stmt = con.createStatement();

            con.setAutoCommit(false);

            stmt.executeUpdate("INSERT INTO Students (id, name, email) VALUES (2, 'Rob', 'rob@gmail.com')");

            stmt.executeUpdate("INSERT INTO Students (id, name, email) VALUES (3, 'Dave', 'dave@gmail.com')");

            stmt.executeUpdate("INSERT INTO Students (id, name, age) VALUES (2, 'Error', 20)");

            con.commit();

            System.out.println("Transaction committed.");

        } catch (SQLException e) {

            e.printStackTrace();

            if (con != null) {

                try {

                    System.out.println("Rolling back transaction...");

                    con.rollback();

                } catch (SQLException ex) {

                    ex.printStackTrace();

                }

            }

        } finally {

            try {

                if (stmt != null) stmt.close();

                if (con != null) con.close();

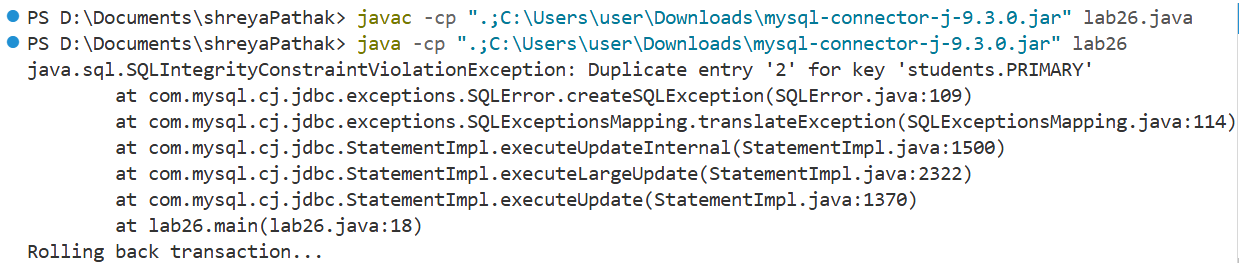
            } catch (SQLException ex) {

                ex.printStackTrace();

            }}}

}

Output:



**Lab 27**: **Demonstrate the use of CachedRowSet to fetch data from a database and update it offline.**

import javax.sql.rowset.CachedRowSet;

import javax.sql.rowset.RowSetProvider;

import java.sql.\*;

public class lab27 {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/demo\_db";

        String user = "root";

        String password = "oracle123";

        Connection con = null;

        CachedRowSet crs = null;

        try {

            // 1. Create CachedRowSet

            crs = RowSetProvider.newFactory().createCachedRowSet();

            // 2. Set DB connection info and query

            crs.setUrl(url);

            crs.setUsername(user);

            crs.setPassword(password);

            crs.setCommand("SELECT id, name, email FROM Students");

            crs.execute();

            System.out.println("Original Data");

            while (crs.next()) {

                System.out.println(crs.getInt("id") + " - " + crs.getString("name") + " - " + crs.getString("email"));

            }

            // 3. Update offline

            crs.beforeFirst();

            while (crs.next()) {

                if (crs.getInt("id") == 1) {

                    crs.updateString("name", "UpdatedName");

                    crs.updateRow();

                }

            }

            // 4. Reconnect manually and apply changes

            con = DriverManager.getConnection(url, user, password);

            con.setAutoCommit(false);

            crs.acceptChanges(con);

            // 5. Print updated data

            crs.beforeFirst();

            System.out.println("\nUpdated Data");

            while (crs.next()) {

                System.out.println(crs.getInt("id") + " - " + crs.getString("name") + " - " + crs.getString("email"));

            }

        } catch (SQLException e) {

            e.printStackTrace();

        } finally {

            try {

                if (crs != null) crs.close();

                if (con != null) con.close();

            } catch (SQLException ex) {

                ex.printStackTrace();

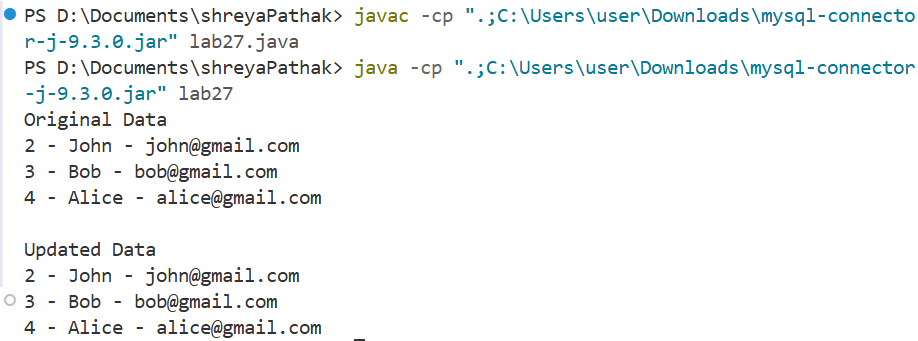
            }

        }

    }

}

Output:



**Lab 28: Use the InetAddress class to find the hostname and IP address of a given domain name.**

import java.net.\*;

public class lab28 {

    public static void main(String[] args) {

        try {

            String domain = "www.google.com";

            InetAddress inet = InetAddress.getByName(domain);

            System.out.println("Host Name: " + inet.getHostName());

            System.out.println("IP Address: " + inet.getHostAddress());

        } catch (UnknownHostException e) {

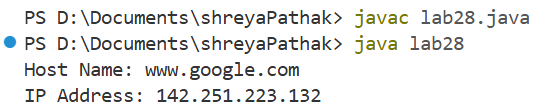
            e.printStackTrace();

        }

    }

}

Output:



**Lab 29: Create a client-server application where the client sends a message, and the server responds with a reversed version of the message.**

**lab29\_Client.java**

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class lab29\_Client {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter message to send to server: ");

        String message = sc.nextLine();

        try {

            Socket socket = new Socket("localhost", 5000);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            // Send message to server

            out.println(message);

            // Receive reversed message from server

            String reversed = in.readLine();

            System.out.println("Server reversed message: " + reversed);

            socket.close();

        } catch (IOException e) {

            e.printStackTrace();

        }

        sc.close();

    }

}

**lab29\_Server.java**

import java.io.\*;

import java.net.\*;

public class lab29\_Server {

    public static void main(String[] args) {

        try {

            ServerSocket serverSocket = new ServerSocket(5000);

            System.out.println("Server started. Waiting for client...");

            Socket socket = serverSocket.accept();

            System.out.println("Client connected.");

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            // receive message from client

            String message = in.readLine();

            // Reverse the message

            String reversed = new StringBuilder(message).reverse().toString();

            // Send reversed message back to client

            out.println(reversed);

            socket.close();

            serverSocket.close();

        } catch (IOException e) {

            e.printStackTrace();

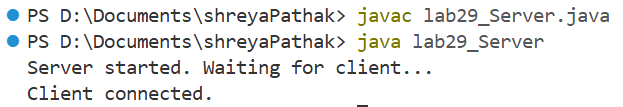
        }

    }

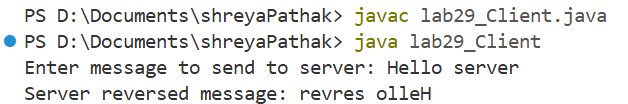
}

Output:

Server Console:



Client Console:



**Lab 30: Implement a UDP-based program where the client sends a number, and the server responds with its square.**

**lab30\_UDPClient.java**

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class lab30\_UDPServer {

   public lab30\_UDPServer() {

   }

   public static void main(String[] var0) throws IOException {

      DatagramSocket var1 = new DatagramSocket(9876);

      System.out.println("UDP Server running...");

      byte[] var2 = new byte[256];

      DatagramPacket var3 = new DatagramPacket(var2, var2.length);

      var1.receive(var3);

      InetAddress var4 = var3.getAddress();

      int var5 = var3.getPort();

      int var6 = Integer.parseInt(new String(var3.getData(), 0, var3.getLength()));

      int var7 = var6 \* var6;

      String var8 = String.valueOf(var7);

      var2 = var8.getBytes();

      var3 = new DatagramPacket(var2, var2.length, var4, var5);

      var1.send(var3);

      var1.close();

   }

}

**lab30\_UDPServer.java**

import java.net.\*;

import java.io.\*;

public class lab30\_UDPServer {

    public static void main(String[] args) throws IOException {

        DatagramSocket socket = new DatagramSocket(9876);

        System.out.println("UDP Server running...");

        byte[] buf = new byte[256];

        DatagramPacket packet = new DatagramPacket(buf, buf.length);

        socket.receive(packet);

        InetAddress address = packet.getAddress();

        int port = packet.getPort();

        int number = Integer.parseInt(new String(packet.getData(), 0, packet.getLength()));

        int square = number \* number;

        String s = String.valueOf(square);

        buf = s.getBytes();

        packet = new DatagramPacket(buf, buf.length, address, port);

        socket.send(packet);

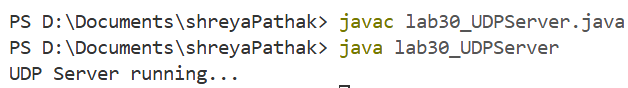
        socket.close();

    }

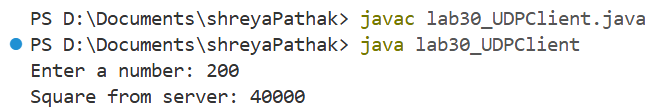
}

Output:

Server Console:



Client Console:



**Lab31: Create a program to download and save an image file from a given URL.**

import java.io.\*;

import java.net.\*;

public class lab31 {

    public static void main(String[] args) {

        try {

            URL url = new URL("https://upload.wikimedia.org/wikipedia/commons/3/3f/Fronalpstock\_big.jpg");

            // Open connection and set User-Agent

            URLConnection connection = url.openConnection();

            connection.setRequestProperty("User-Agent", "Mozilla/5.0");

            InputStream in = connection.getInputStream();

            FileOutputStream out = new FileOutputStream("downloaded.jpg");

            int data;

            while ((data = in.read()) != -1) {

                out.write(data);

            }

            in.close();

            out.close();

            System.out.println("Image downloaded successfully as downloaded.jpg");

        } catch (Exception e) {

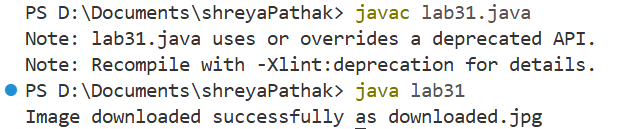
            e.printStackTrace();

        }

    }

}

Output:



**Lab 32: Implement a program to connect to a mail server and fetch emails from the inbox.**

import java.util.Properties;

import javax.mail.\*;

public class lab32 {

    public static void main(String[] args) {

        String host = "imap.gmail.com";

        String username = "shreyapathak.9818@gmail.com";

        String password = "kugd tfrp enmm nplg";

        Properties props = new Properties();

        props.put("mail.store.protocol", "imaps");

        try {

            Session session = Session.getDefaultInstance(props, null);

            Store store = session.getStore("imaps");

            store.connect(host, username, password);

            Folder inbox = store.getFolder("INBOX");

            inbox.open(Folder.READ\_ONLY);

            Message[] messages = inbox.getMessages();

            System.out.println("Total messages: " + messages.length);

            for (int i = 0; i < Math.min(5, messages.length); i++) {

                System.out.println("--------------------------------");

                System.out.println("Subject: " + messages[i].getSubject());

                System.out.println("From: " + messages[i].getFrom()[0]);

            }

            inbox.close(false);

            store.close();

        } catch (Exception e) {

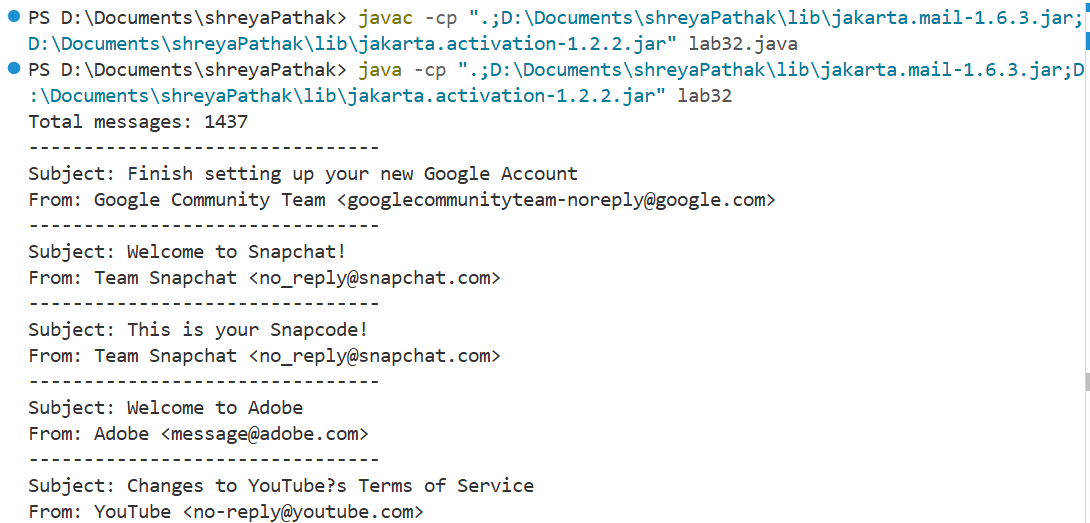
            e.printStackTrace();

        }

    }

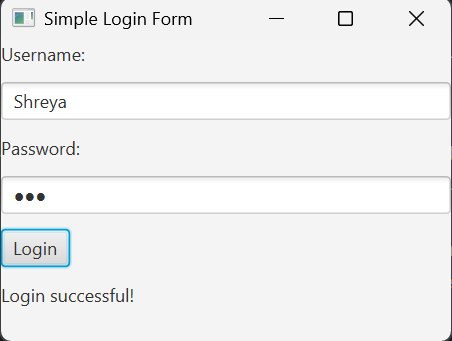
}

Output:



**Lab 33: Create a simple login form using JavaFX.**  
  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
public class lab33 extends Application {  
 @Override  
 public void start(Stage stage) {  
 stage.setTitle("Simple Login Form");  
 Label userLabel = new Label("Username:");  
 TextField userField = new TextField();  
 Label passLabel = new Label("Password:");  
 PasswordField passField = new PasswordField();  
 Button loginBtn = new Button("Login");  
 Label message = new Label();  
 loginBtn.setOnAction(e -> {  
 String username = userField.getText();  
 String password = passField.getText();  
 if (username.isEmpty() || password.isEmpty()) {  
 message.setText("Please enter both username and password!");  
 } else {  
 message.setText("Login successful!");  
 }  
 });  
 VBox vbox = new VBox(10, userLabel, userField, passLabel, passField, loginBtn, message);  
 Scene scene = new Scene(vbox, 300, 200);  
 stage.setScene(scene);  
 stage.show();  
 }  
 public static void main(String[] args) {  
 launch(args);  
 }  
}

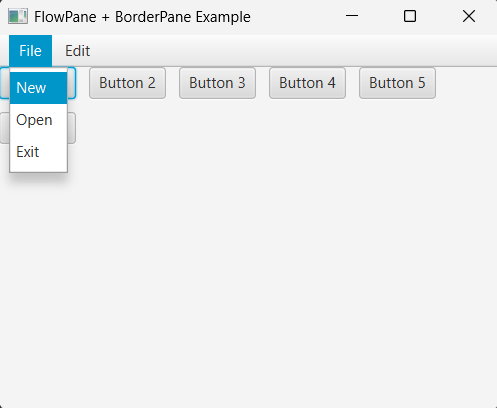
Output:



**Lab 34: Design a GUI with a FlowPane to arrange buttons and a BorderPane to place a menu bar at the top and buttons at the center.**

import javafx.application.Application;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.Menu;  
import javafx.scene.control.MenuBar;  
import javafx.scene.control.MenuItem;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.FlowPane;  
import javafx.stage.Stage;  
  
public class lab34 extends Application {  
  
 @Override  
 public void start(Stage stage) {  
 stage.setTitle("FlowPane + BorderPane Example");  
  
 //Menu Bar at Top  
 MenuBar menuBar = new MenuBar();  
  
 Menu fileMenu = new Menu("File");  
 fileMenu.getItems().addAll(  
 new MenuItem("New"),  
 new MenuItem("Open"),  
 new MenuItem("Exit")  
 );  
  
 Menu editMenu = new Menu("Edit");  
 editMenu.getItems().addAll(  
 new MenuItem("Cut"),  
 new MenuItem("Copy"),  
 new MenuItem("Paste")  
 );  
  
 menuBar.getMenus().addAll(fileMenu, editMenu);  
  
 //FlowPane at Center with Buttons  
 FlowPane flowPane = new FlowPane();  
 flowPane.setHgap(10);  
 flowPane.setVgap(10);  
  
 for (int i = 1; i <= 6; i++) {  
 flowPane.getChildren().add(new Button("Button " + i));  
 }  
  
 //BorderPane Layout  
 BorderPane borderPane = new BorderPane();  
 borderPane.setTop(menuBar);  
 borderPane.setCenter(flowPane);  
  
 Scene scene = new Scene(borderPane, 400, 300);  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args) {  
 launch(args);  
 }  
}

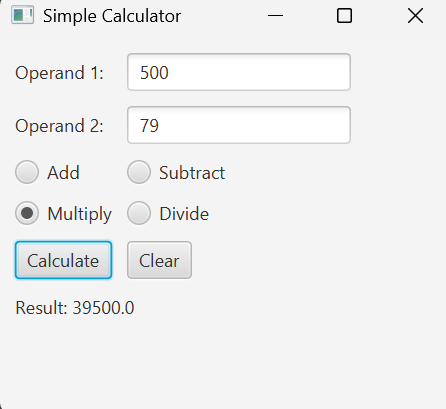
Output:



**Lab 35: Design a calculator interface using a GridPane layout.**

import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.GridPane;  
import javafx.stage.Stage;  
  
public class Lab\_35 extends Application {  
  
 private TextField op1Field = new TextField();  
 private TextField op2Field = new TextField();  
 private Label resultLabel = new Label("Result: ");  
  
 @Override  
 public void start(Stage stage) {  
 GridPane grid = new GridPane();  
 grid.setPadding(new Insets(10));  
 grid.setHgap(10);  
 grid.setVgap(10);  
  
 op1Field.setPromptText("Enter first number");  
 op2Field.setPromptText("Enter second number");  
  
 grid.add(new Label("Operand 1:"), 0, 0);  
 grid.add(op1Field, 1, 0);  
 grid.add(new Label("Operand 2:"), 0, 1);  
 grid.add(op2Field, 1, 1);  
  
 // Radio buttons for operations  
 ToggleGroup operations = new ToggleGroup();  
  
 RadioButton add = new RadioButton("Add");  
 RadioButton sub = new RadioButton("Subtract");  
 RadioButton mul = new RadioButton("Multiply");  
 RadioButton div = new RadioButton("Divide");  
  
 add.setToggleGroup(operations);  
 sub.setToggleGroup(operations);  
 mul.setToggleGroup(operations);  
 div.setToggleGroup(operations);  
  
 grid.add(add, 0, 2);  
 grid.add(sub, 1, 2);  
 grid.add(mul, 0, 3);  
 grid.add(div, 1, 3);  
  
 // Buttons  
 Button calcButton = new Button("Calculate");  
 Button clearButton = new Button("Clear");  
  
 grid.add(calcButton, 0, 4);  
 grid.add(clearButton, 1, 4);  
 grid.add(resultLabel, 0, 5, 2, 1);  
  
 // Button actions  
 calcButton.setOnAction(e -> calculate(operations));  
 clearButton.setOnAction(e -> clearAll(operations));  
  
 stage.setTitle("Simple Calculator");  
 Scene scene = new Scene(grid, 300, 250);  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 private void calculate(ToggleGroup operations) {  
 try {  
 double op1 = Double.*parseDouble*(op1Field.getText());  
 double op2 = Double.*parseDouble*(op2Field.getText());  
 double result = 0;  
  
 RadioButton selected = (RadioButton) operations.getSelectedToggle();  
 if (selected == null) {  
 resultLabel.setText("Result: Select an operation");  
 return;  
 }  
  
 switch (selected.getText()) {  
 case "Add": result = op1 + op2; break;  
 case "Subtract": result = op1 - op2; break;  
 case "Multiply": result = op1 \* op2; break;  
 case "Divide":  
 if (op2 != 0) result = op1 / op2;  
 else {  
 resultLabel.setText("Result: Error (Division by zero)");  
 return;  
 }  
 break;  
 }  
  
 resultLabel.setText("Result: " + result);  
  
 } catch (NumberFormatException ex) {  
 resultLabel.setText("Result: Invalid Input");  
 }  
 }  
  
 private void clearAll(ToggleGroup operations) {  
 op1Field.clear();  
 op2Field.clear();  
 resultLabel.setText("Result: ");  
 operations.getSelectedToggle().setSelected(false);  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

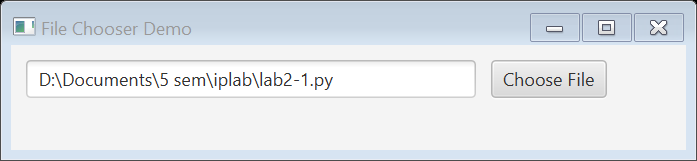
Output:



**Lab 36: Implement a program that allows the user to choose a file and display its absolute path in a TextField.**

import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.HBox;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
import java.io.File;  
  
public class lab36 extends Application {  
  
 @Override  
 public void start(Stage stage) {  
 stage.setTitle("File Chooser Demo");  
  
 TextField pathField = new TextField();  
 pathField.setPrefWidth(300);  
 pathField.setEditable(false);  
  
 Button chooseButton = new Button("Choose File");  
  
 chooseButton.setOnAction(e -> {  
 FileChooser fileChooser = new FileChooser();  
 File selectedFile = fileChooser.showOpenDialog(stage);  
 if (selectedFile != null) {  
 pathField.setText(selectedFile.getAbsolutePath());  
 }  
 });  
 HBox hbox = new HBox(10, pathField, chooseButton);  
 hbox.setPadding(new Insets(10));  
  
 Scene scene = new Scene(hbox, 450, 70);  
 stage.setScene(scene);  
 stage.show();  
 }  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

Output:



**Lab 37: Create a servlet that uses cookies to store user preferences and a session to manage login information.**

package com.example;

import jakarta.servlet.\*;

import jakarta.servlet.http.\*;

import jakarta.servlet.annotation.WebServlet;

import java.io.IOException;

@WebServlet("/UserPreferenceServlet")

public class PreferenceServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override

    protected void doGet(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        HttpSession session = request.getSession();

        // Simulate login

        String username = (String) session.getAttribute("username");

        if (username == null) {

            username = "JohnDoe";

            session.setAttribute("username", username);

        }

        // Check for theme cookie

        String theme = "light";

        Cookie[] cookies = request.getCookies();

        if (cookies != null) {

            for (Cookie c : cookies) {

                if ("theme".equals(c.getName())) {

                    theme = c.getValue();

                }

            }

        }

        // Update theme if user selects new

        String newTheme = request.getParameter("theme");

        if (newTheme != null && !newTheme.isEmpty()) {

            theme = newTheme;

            Cookie themeCookie = new Cookie("theme", theme);

            themeCookie.setMaxAge(60 \* 60 \* 24 \* 30); // 30 days

            response.addCookie(themeCookie);

        }

        // Output

        response.setContentType("text/html");

        response.getWriter().println("<html><body>");

        response.getWriter().println("<h2>Welcome, " + username + "!</h2>");

        response.getWriter().println("<p>Current Theme: " + theme + "</p>");

        response.getWriter().println("<form method='get'>");

        response.getWriter().println("Select Theme: <select name='theme'>");

        response.getWriter().println("<option value='light'>Light</option>");

        response.getWriter().println("<option value='dark'>Dark</option>");

        response.getWriter().println("</select>");

        response.getWriter().println("<input type='submit' value='Save'>");

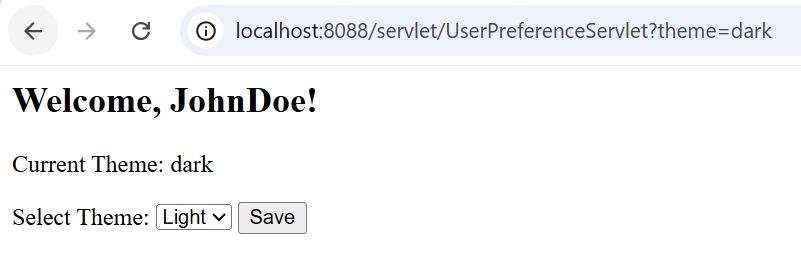
        response.getWriter().println("</form>");

        response.getWriter().println("</body></html>");

    }

}

Output:



**Lab 38: Create a JSP page and servlet that insert, update, delete and fetches data from a database and displays it in an HTML table.**

**StudentServlet.java**

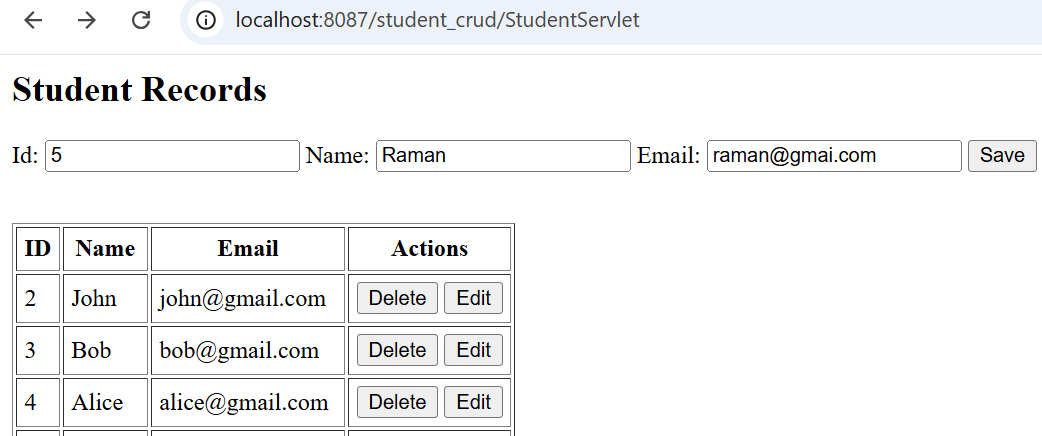
package com.example;  
  
import jakarta.servlet.\*;  
import jakarta.servlet.http.\*;  
import jakarta.servlet.annotation.WebServlet;  
import java.io.\*;  
import java.sql.\*;  
  
@WebServlet("/StudentServlet")  
public class StudentServlet extends HttpServlet {  
 private static final long *serialVersionUID* = 1L;  
  
 private final String url = "jdbc:mysql://localhost:3306/demo\_db";  
 private final String user = "root";  
 private final String password = "oracle123";  
  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 String action = request.getParameter("action");  
 String idStr = request.getParameter("id");  
  
 Connection conn = null;  
 PreparedStatement ps = null;  
 ResultSet rs = null;  
  
 try {  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 response.getWriter().println("JDBC Driver not found: " + e.getMessage());  
 return;  
 }  
 conn = DriverManager.*getConnection*(url, user, password);  
  
 if ("delete".equals(action) && idStr != null) {  
 int id = Integer.*parseInt*(idStr);  
 ps = conn.prepareStatement("DELETE FROM students WHERE id=?");  
 ps.setInt(1, id);  
 ps.executeUpdate();  
 }  
  
 // Fetch all students  
 ps = conn.prepareStatement("SELECT \* FROM students");  
 rs = ps.executeQuery();  
 request.setAttribute("resultSet", rs);  
  
 RequestDispatcher rd = request.getRequestDispatcher("index.jsp");  
 rd.forward(request, response);  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 response.getWriter().println("Database error: " + e.getMessage());  
 } finally {  
 try {  
 if (rs != null)  
 rs.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 if (ps != null)  
 ps.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 if (conn != null)  
 conn.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 String idStr = request.getParameter("id");  
 String name = request.getParameter("name");  
 String emailStr = request.getParameter("email");  
 String submitValue = request.getParameter("submit");  
  
 Connection conn = null;  
 PreparedStatement ps = null;  
  
 try {  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 response.getWriter().println("JDBC Driver not found: " + e.getMessage());  
 return;  
 }  
 conn = DriverManager.*getConnection*(url, user, password);  
  
 int id = Integer.*parseInt*(idStr);  
 if ("Update".equals(request.getParameter("submit"))) {  
 // Update  
 System.*out*.println("Updating student: id=" + id + ", name=" + name + ", email=" + emailStr);  
 ps = conn.prepareStatement("UPDATE students SET name=?, email=? WHERE id=?");  
 ps.setString(1, name);  
 ps.setString(2, emailStr);  
 ps.setInt(3, id);  
 int rows = ps.executeUpdate();  
 System.*out*.println("Rows updated: " + rows);  
 } else {  
 // Insert  
 ps = conn.prepareStatement("INSERT INTO students (id, name, email) VALUES (?, ?, ?)");  
 ps.setInt(1, id);  
 ps.setString(2, name);  
 ps.setString(3, emailStr);  
 ps.executeUpdate();  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if (ps != null)  
 ps.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 try {  
 if (conn != null)  
 conn.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 response.sendRedirect("StudentServlet");  
 }  
}

**index.jsp**

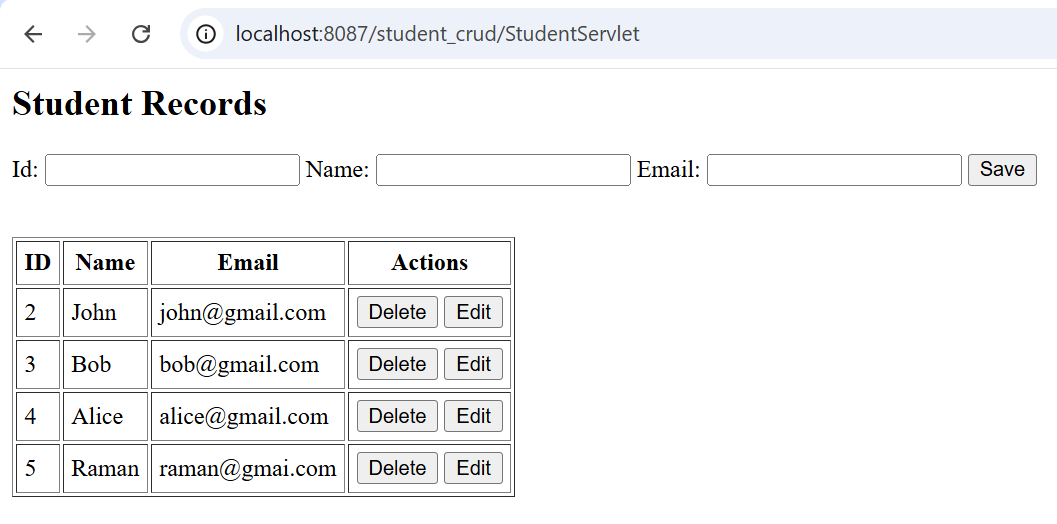
<%@ **page** import="java.sql.\*, java.util.\*" %>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Student Management</title>  
</head>  
<body>  
<h2>Student Records</h2>  
  
<!-- Student Form -->  
<form action="StudentServlet" method="post">  
 Id: <input type="text" name="id" value="<%= request.getParameter("editId") != null ? request.getParameter("editId") : "" %>" <%= request.getParameter("editId") != null ? "readonly" : "" %>>  
 Name: <input type="text" name="name" value="<%= request.getParameter("editName") != null ? request.getParameter("editName") : "" %>" required>  
 Email: <input type="text" name="email" value="<%= request.getParameter("editEmail") != null ? request.getParameter("editEmail") : "" %>" required>  
 <input type="submit" name="submit" value="<%= request.getParameter("editId") != null ? "Update" : "Save" %>">  
</form>  
  
<br/>  
  
<!-- Student Table -->  
<table border="1" cellpadding="5">  
 <tr>  
 <th>ID</th>  
 <th>Name</th>  
 <th>Email</th>  
 <th>Actions</th>  
 </tr>  
 <%  
 ResultSet rs = (ResultSet) request.getAttribute("resultSet");  
 if (rs != null) {  
 while (rs.next()) {  
 int id = rs.getInt("id");  
 String name = rs.getString("name");  
 String email = rs.getString("email");  
  
 %>  
 <tr>  
 <td><%= id %></td>  
 <td><%= name %></td>  
 <td><%= email %></td>  
 <td>  
 <form action="StudentServlet" method="get" style="display:inline;">  
 <input type="hidden" name="action" value="delete">  
 <input type="hidden" name="id" value="<%= id %>">  
 <input type="submit" value="Delete">  
 </form>  
 <form action="index.jsp" method="get" style="display:inline;">  
 <input type="hidden" name="editId" value="<%= id %>">  
 <input type="hidden" name="editName" value="<%= name %>">  
 <input type="hidden" name="editEmail" value="<%= email %>">  
 <input type="submit" value="Edit">  
 </form>  
 </td>  
 </tr>  
 <%  
 }  
 rs.close();  
 }  
 %>  
</table>  
</body>  
</html>

Output:

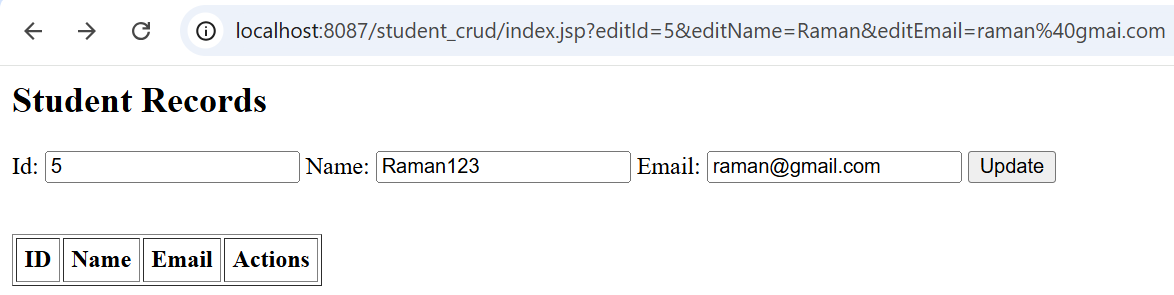
Insert Data:



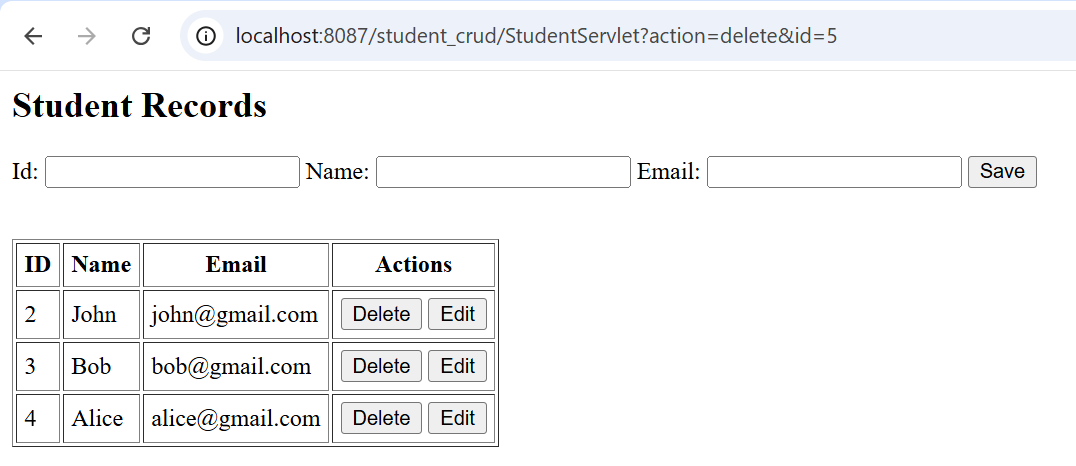
Fetch Data:



Edit Data:



Delete Data:



**Lab 39: Create an RMI application where the client sends a number, and the server responds with its factorial.**

**Factorial.java**

package rmi\_practical;

import java.rmi.\*;

public interface Factorial extends Remote {

    long getFactorial(int n) throws RemoteException;

}

**FactorialImpl.java**

package rmi\_practical;

import java.rmi.server.\*;

import java.rmi.\*;

public class FactorialImpl extends UnicastRemoteObject implements Factorial {

    public FactorialImpl() throws RemoteException {

        super();

    }

    public long getFactorial(int n) throws RemoteException {

        long fact = 1;

        for (int i = 1; i <= n; i++) {

            fact \*= i;

        }

        return fact;

    }

}

**Client.java**

package rmi\_practical;

import java.rmi.\*;

import java.util.Scanner;

public class Client {

    public static void main(String[] args) {

        try {

            Factorial stub = (Factorial) Naming.lookup("//localhost/FactorialService");

            Scanner sc = new Scanner(System.in);

            System.out.print("Enter a number: ");

            int n = sc.nextInt();

            long result = stub.getFactorial(n);

            System.out.println("Factorial of " + n + " is: " + result);

        } catch (Exception e) {

            System.out.println("Client failed: " + e);

        }

    }

}

**Server.java**

package rmi\_practical;

import java.rmi.\*;

public class Server {

    public static void main(String[] args) {

        try {

            FactorialImpl skeleton = new FactorialImpl();

            Naming.rebind("//localhost/FactorialService", skeleton);

            System.out.println("Factorial Server is ready.");

        } catch (Exception e) {

            System.out.println("Server failed: " + e);

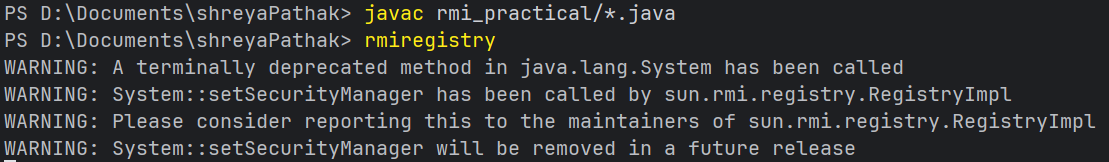
        }

    }

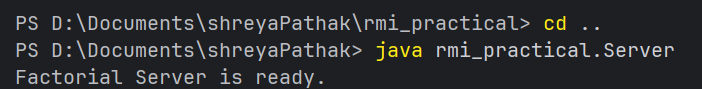
}

Output:

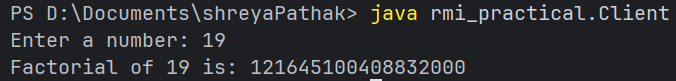
Compile all files and start RMI registry:



Server Console:



Client Console:



**Lab 40: Write and execute a CORBA program using IDL to define the interface and implement the server and client.**

**Hello.idl**

module HelloApp {

interface Hello {

string sayHello();

};

};

**HelloImpl.java**

import HelloApp.\*;

public class HelloImpl extends HelloPOA {

public String sayHello() {

return "Hello from CORBA!";

}

}

**HelloServer.java**

import HelloApp.\*;

import org.omg.CORBA.\*;

import org.omg.PortableServer.\*;

import org.omg.PortableServer.POA;

public class HelloServer {

public static void main(String args[]) {

try {

// Initialize ORB

ORB orb = ORB.init(args, null);

// Get reference to RootPOA & activate the POAManager

POA rootpoa = POAHelper.narrow(orb.resolve\_initial\_references("RootPOA"));

rootpoa.the\_POAManager().activate();

// Create servant & register it with ORB

HelloImpl helloImpl = new HelloImpl();

// Get object reference from servant

org.omg.CORBA.Object ref = rootpoa.servant\_to\_reference(helloImpl);

Hello href = HelloHelper.narrow(ref);

// Save object reference to file

java.io.PrintWriter out = new java.io.PrintWriter("Hello.ref");

out.println(orb.object\_to\_string(href));

out.close();

System.out.println("Server ready and waiting...");

// Wait for invocations

orb.run();

} catch (Exception e) {

e.printStackTrace();

}

}

}

**HelloClient.java**

import HelloApp.\*;

import org.omg.CORBA.\*;

public class HelloClient {

public static void main(String args[]) {

try {

// Initialize ORB

ORB orb = ORB.init(args, null);

// Read object reference from file

java.io.BufferedReader br = new java.io.BufferedReader(new java.io.FileReader("Hello.ref"));

String ior = br.readLine();

br.close();

// Convert string back to object reference

org.omg.CORBA.Object obj = orb.string\_to\_object(ior);

// Narrow it to Hello

Hello helloRef = HelloHelper.narrow(obj);

// Call method

String response = helloRef.sayHello();

System.out.println("Response from server: " + response);

} catch (Exception e) {

e.printStackTrace();

}

}

}

Output:

Compile the IDL:



This generates:

HelloApp/Hello.java

HelloApp/\_HelloStub.java

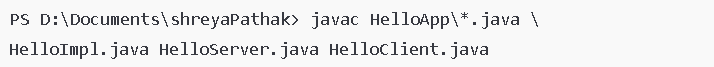
HelloApp/HelloHelper.java

HelloApp/HelloHolder.java

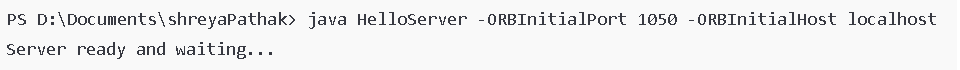
HelloApp/HelloOperations.java

HelloApp/HelloPOA.java

Compile Everything:



Run the Server:



Run the Client:

A close-up of a word

AI-generated content may be incorrect.