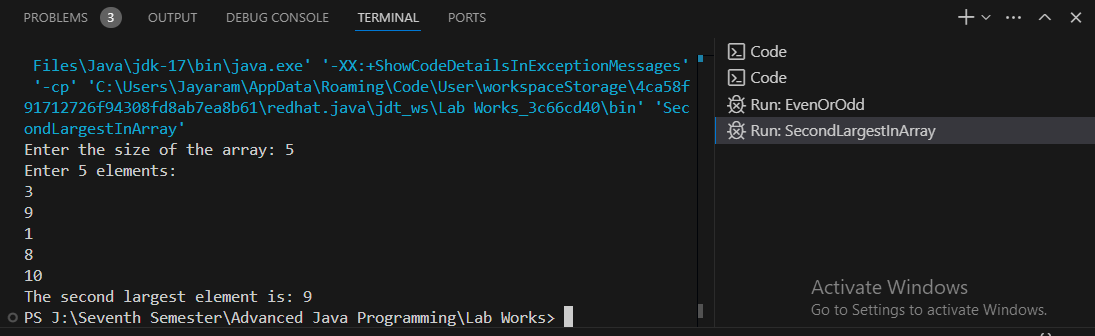
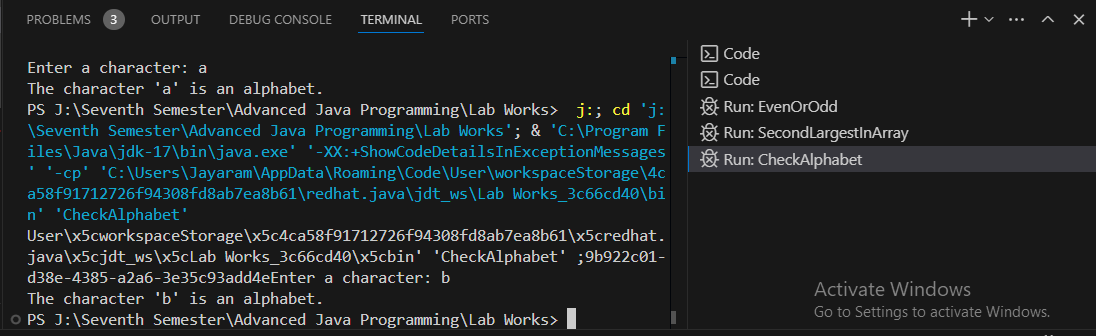
Output of Lab1:

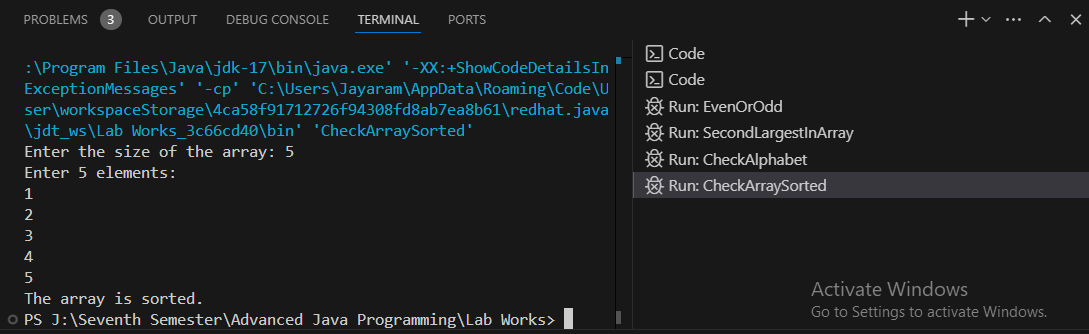
Output of Lab2



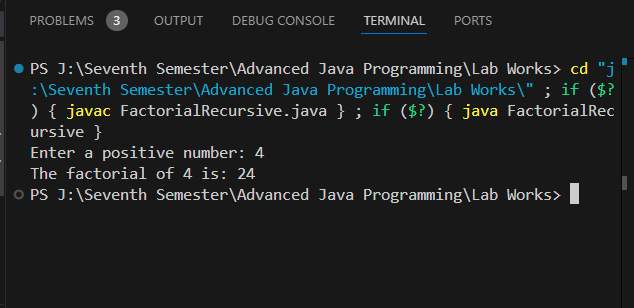
Output of Lab3:



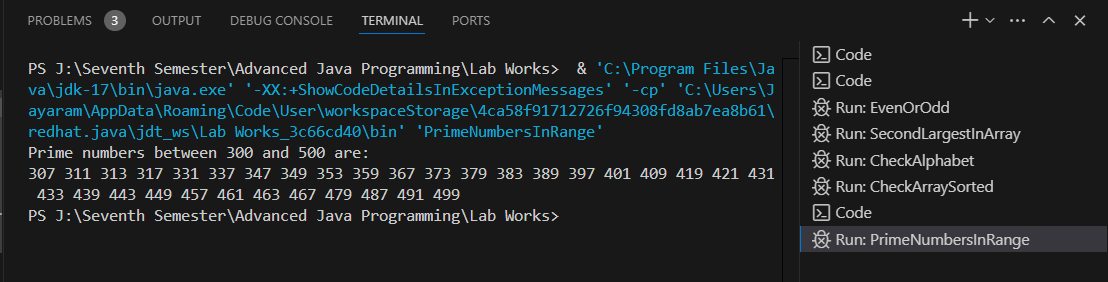
Lab4 :



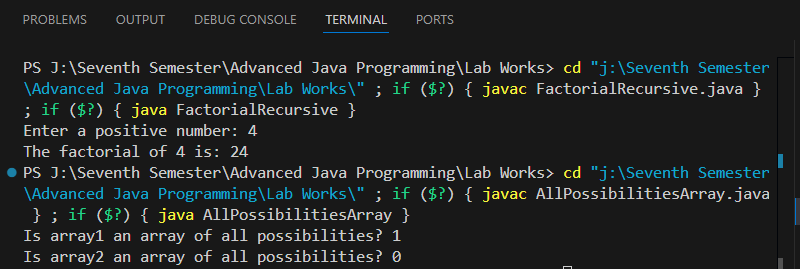
Lab 5:



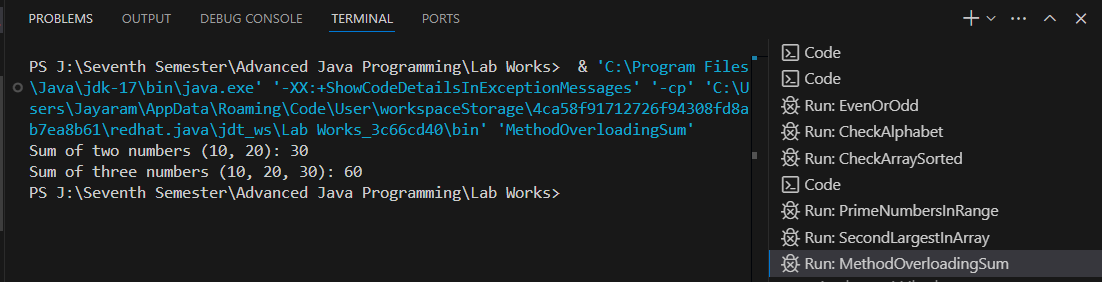
Lab 6:



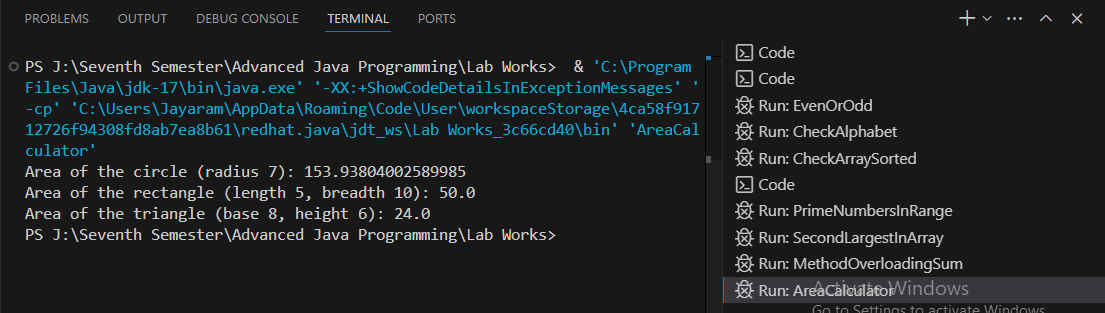
Lab 7:

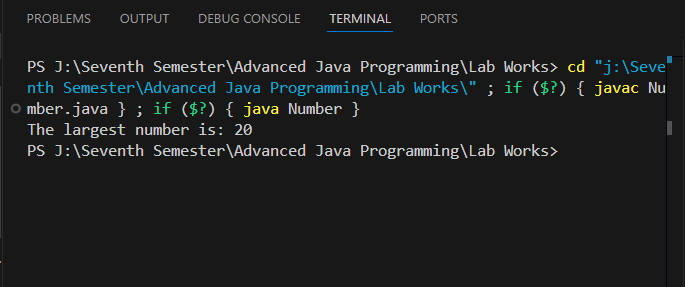


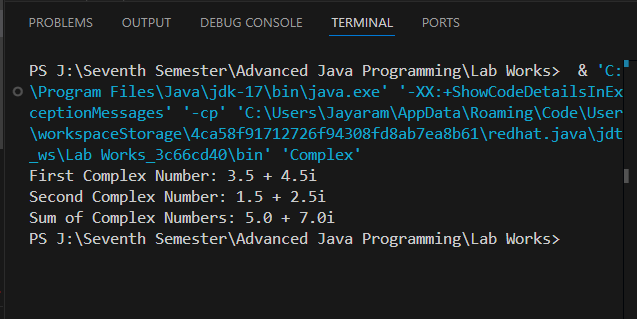
Lab 8:

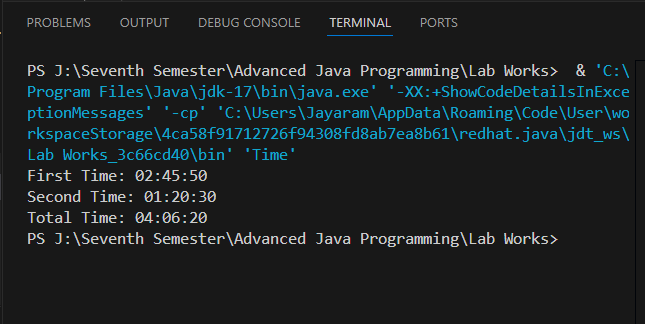


Lab 9:

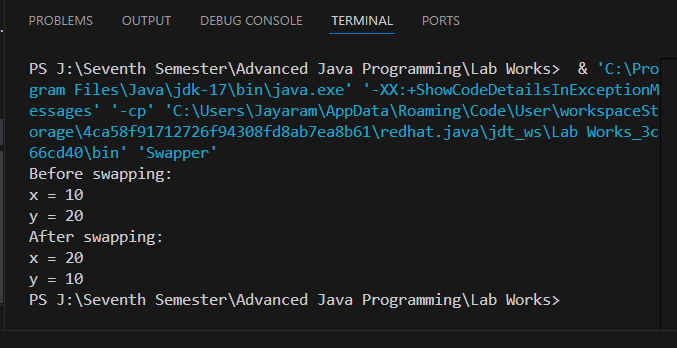


Lab 10:

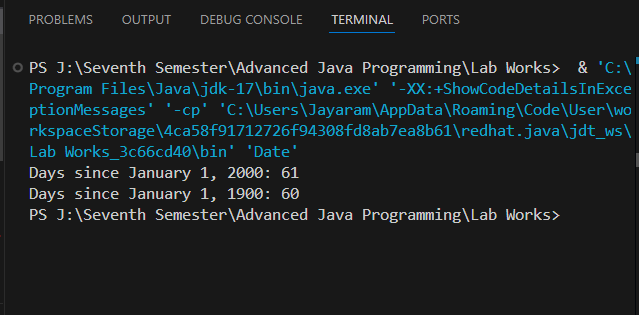
Lab 11:

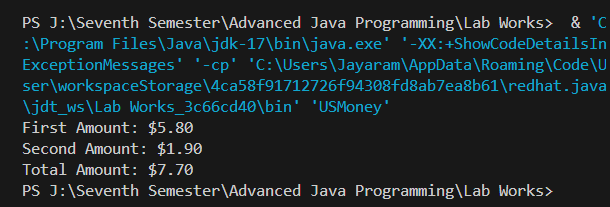
Lab 12: 

Lab 13:

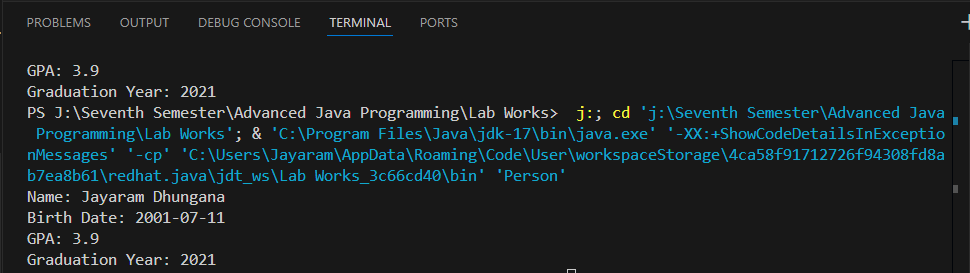


Lab 14:

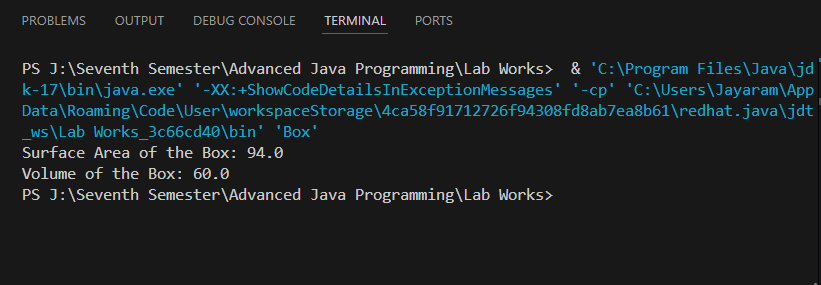


Lab 15:

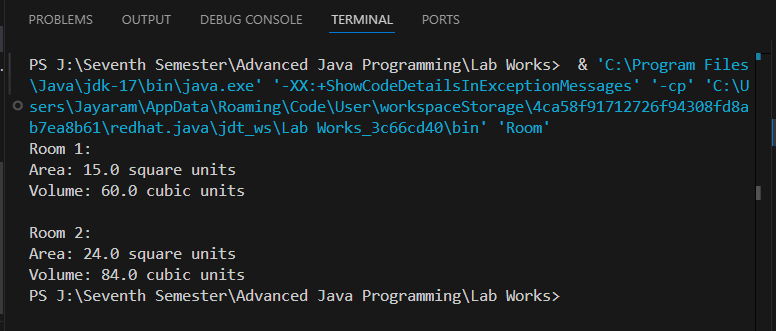
Lab 16:

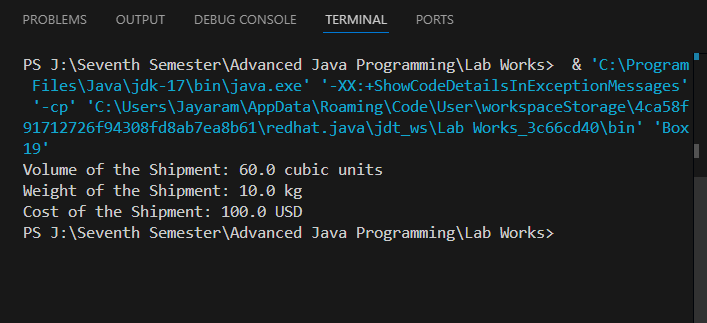


Lab 17

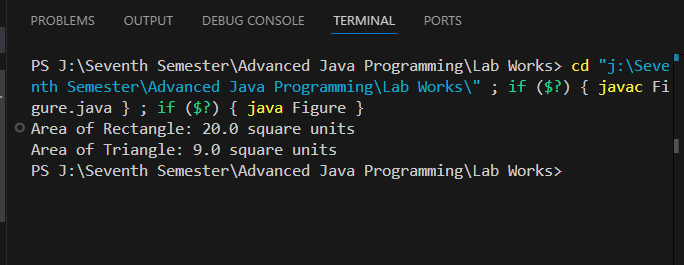


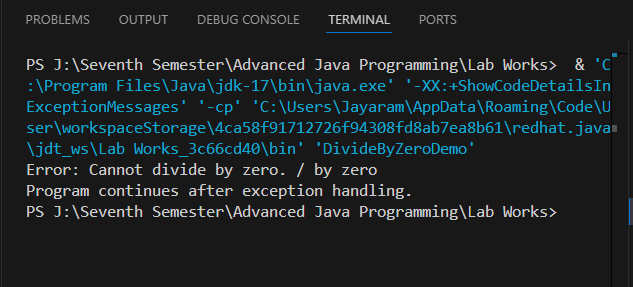
Lab 18:



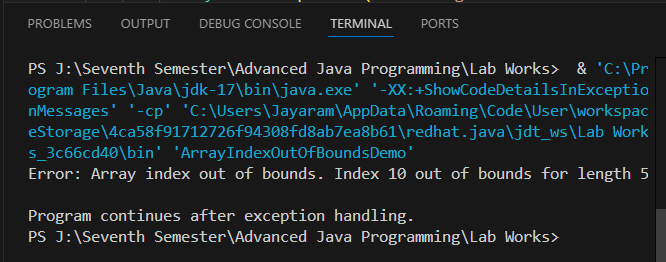
Lab 19:

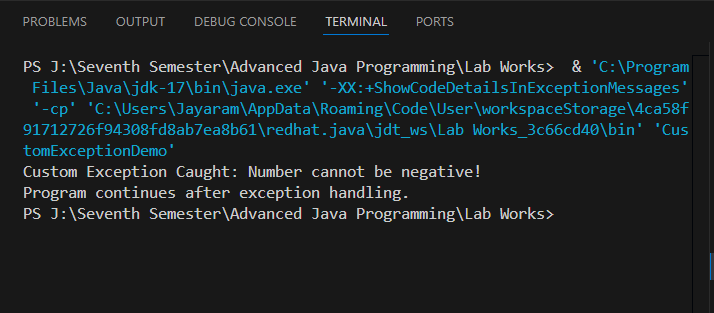
lab 20:



Lab 21

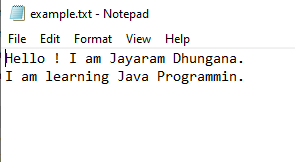
Lab 22:



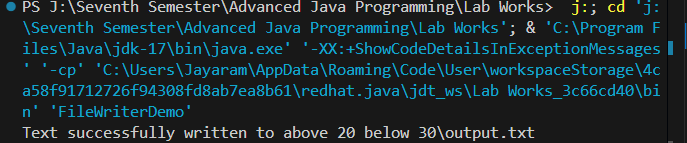
Lab 23:

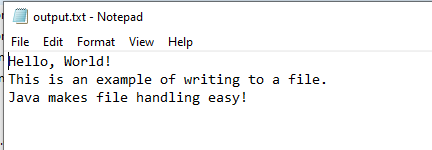
Lab 24:

   
and example.txt is

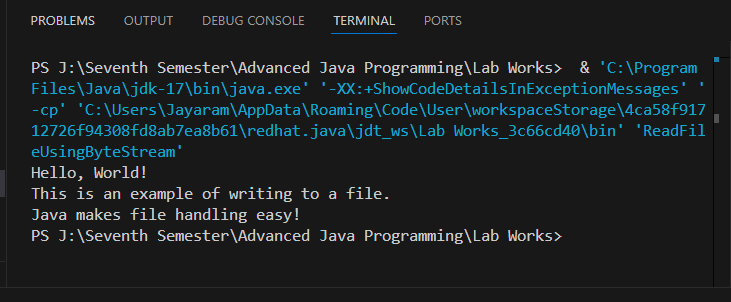


Lab 25:

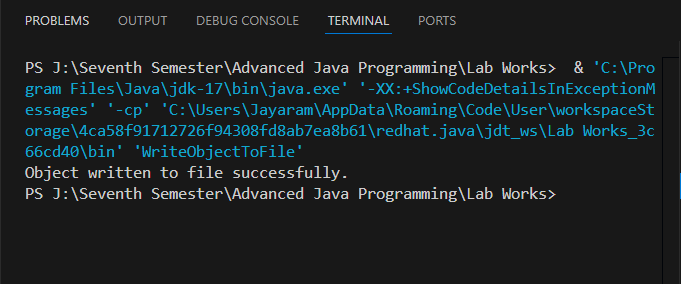


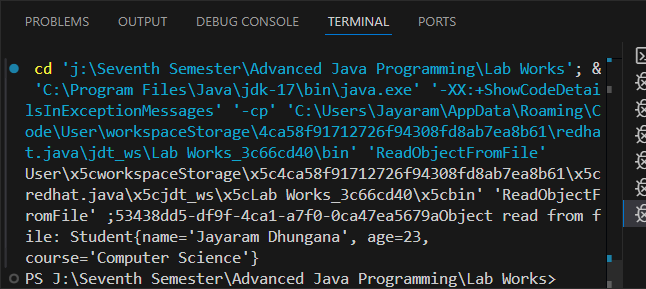
  
Output.txt is

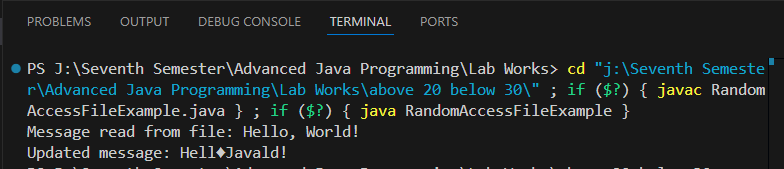
Lab 26:  
Output.txt is same as above.



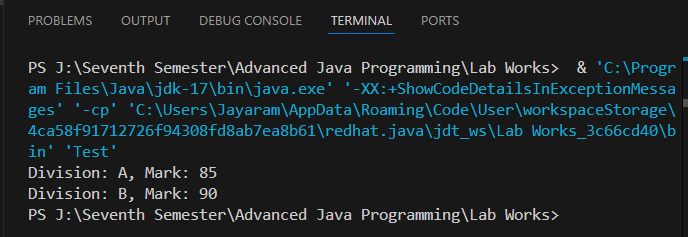
Lab 27:

  
Lab 28:

Lab 29:



Lab 30:



// 31. Write a Java program to create a class Mobile (type, phone\_no).

// Customize the exception such that if the user gives phone\_no having less

// than or greater than 10 digits, then the program has to throw an exception

// with the message "Invalid Phone Number".

class InvalidPhoneNumberException extends Exception {

    public InvalidPhoneNumberException(String message) {

        super(message);

    }

}

class Mobile {

    private String type;

    private String phoneNo;

    public Mobile(String type, String phoneNo) throws InvalidPhoneNumberException {

        if (phoneNo.length() != 10) {

            throw new InvalidPhoneNumberException("Invalid Phone Number: " + phoneNo);

        }

        this.type = type;

        this.phoneNo = phoneNo;

    }

    public String getType() {

        return type;

    }

    public String getPhoneNo() {

        return phoneNo;

    }

    @Override

    public String toString() {

        return "Mobile[type=" + type + ", phoneNo=" + phoneNo + "]";

    }

    public static void main(String[] args) {

        try {

            // Valid phone number

            Mobile mobile1 = new Mobile("Smartphone", "1234567890");

            System.out.println(mobile1);

            // Invalid phone number

            Mobile mobile2 = new Mobile("Feature Phone", "12345");

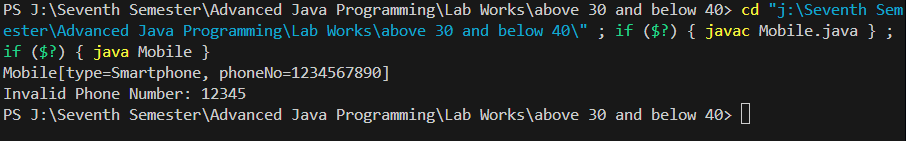
            System.out.println(mobile2);

        } catch (InvalidPhoneNumberException e) {

            System.err.println(e.getMessage());

        }

    }

}

Output:

Lab 32

/\* 32. Create a class named Movie (id, genre). Write the object of Movie class

into a file named “Comedy.dat” having comedy as genre.

\*/

import java.io.\*;

class Movie implements Serializable {

    private int id;

    private String genre;

    public Movie(int id, String genre) {

        this.id = id;

        this.genre = genre;

    }

    public int getId() {

        return id;

    }

    public String getGenre() {

        return genre;

    }

    @Override

    public String toString() {

        return "Movie[id=" + id + ", genre=" + genre + "]";

    }

    public static void main(String[] args) {

        Movie comedyMovie = new Movie(101, "Comedy");

        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("Comedy.dat"))) {

            if (comedyMovie.getGenre().equalsIgnoreCase("Comedy")) {

                oos.writeObject(comedyMovie);

                System.out.println("Movie object written to file: Comedy.dat");

            } else {

                System.out.println("The genre is not comedy, no file written.");

            }

        } catch (IOException e) {

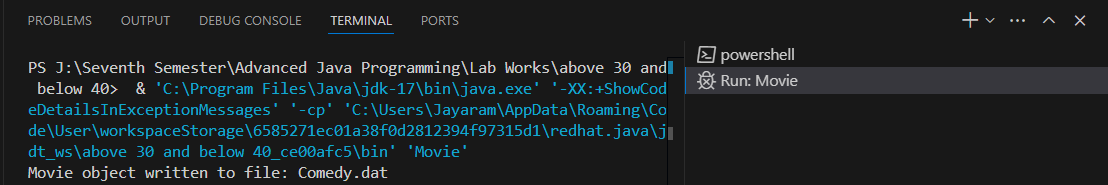
            System.err.println("Error writing to file: " + e.getMessage());

        }

    }

}

Output:



Lab 33:

/\*

33. Write a program to create a class Student with data members roll and name.

Sort the 10 objects of this class on the basis of name.

\*/

import java.util.Arrays;

class Student {

    private int roll;

    private String name;

    // Constructor

    public Student(int roll, String name) {

        this.roll = roll;

        this.name = name;

    }

    // Getters

    public int getRoll() {

        return roll;

    }

    public String getName() {

        return name;

    }

    @Override

    public String toString() {

        return "Student[roll=" + roll + ", name=" + name + "]";

    }

    public static void main(String[] args) {

        // Array of 10 Student objects

        Student[] students = {

            new Student(1, "Jayaram"),

            new Student(2, "kishor"),

            new Student(3, "Srijan"),

            new Student(4, "Sonit"),

            new Student(5, "Chiranjibi"),

            new Student(6, "Sunil"),

            new Student(7, "Dinesh"),

            new Student(8, "Anup"),

            new Student(9, "Anjan"),

            new Student(10, "Sushma")

        };

        // Sorting the array based on name

        Arrays.sort(students, (s1, s2) -> s1.getName().compareTo(s2.getName()));

        // Printing sorted array

        System.out.println("\nSorted Students (by name):");

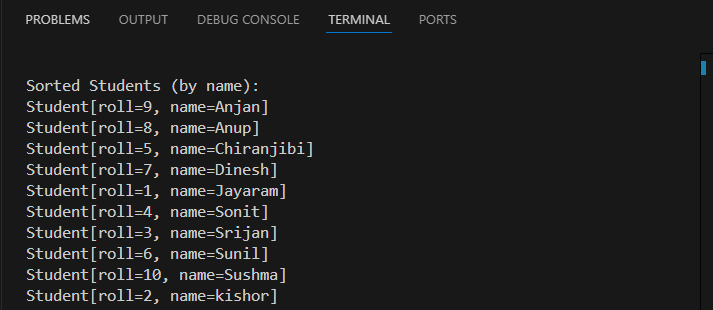
        for (Student student : students) {

            System.out.println(student);

        }

    }

}

Output  


Lab 34:

/\*

34. Create a class named Book with instance variables title and price.

Add a method named setVar to pass parameters for title and price.

Add another method named showVar to display values of these variables.

Now in main(), declare 4 objects of Book and display the records of books that start with “Java”.

\*/

class Book {

    private String title;

    private double price;

    // Method to set the values of title and price

    public void setVar(String title, double price) {

        this.title = title;

        this.price = price;

    }

    // Method to display the values of title and price

    public void showVar() {

        System.out.println("Book[title=" + title + ", price=" + price + "]");

    }

    // Getter for title

    public String getTitle() {

        return title;

    }

    public static void main(String[] args) {

        // Array of 4 Book objects

        Book[] books = new Book[4];

        // Initializing the books

        books[0] = new Book();

        books[0].setVar("Java Programming", 450.0);

        books[1] = new Book();

        books[1].setVar("Python Essentials", 550.0);

        books[2] = new Book();

        books[2].setVar("Java in Depth", 650.0);

        books[3] = new Book();

        books[3].setVar("C++ Basics", 300.0);

        // Displaying books that start with "Java"

        System.out.println("\nBooks that start with 'Java':");

        for (Book book : books) {

            if (book.getTitle().startsWith("Java")) {

                book.showVar();

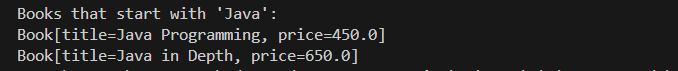
            }

        }

    }

}

Output:



Lab 35:

/\*

35. Create a Shape interface having methods area() and perimeter().

Create two subclasses, Circle and Rectangle that implement the Shape interface.

Create a class Sample with main method and demonstrate the area and perimeters

of both the Shape classes. You need to handle the values of length, breadth, and

radius in respective classes to calculate their area and perimeter.

\*/

// Shape Interface

interface Shape {

    double area();

    double perimeter();

}

// Circle Class

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;

    }

}

// Rectangle Class

class Rectangle implements Shape {

    private double length;

    private double breadth;

    public Rectangle(double length, double breadth) {

        this.length = length;

        this.breadth = breadth;

    }

    @Override

    public double area() {

        return length \* breadth;

    }

    @Override

    public double perimeter() {

        return 2 \* (length + breadth);

    }

}

// Sample Class with Main Method

public class Sample {

    public static void main(String[] args) {

        // Create a Circle object

        Shape circle = new Circle(5.0); // radius = 5.0

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

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

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

        // Create a Rectangle object

        Shape rectangle = new Rectangle(4.0, 6.0); // length = 4.0, breadth = 6.0

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

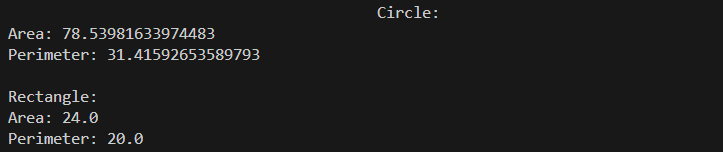
        System.out.println("Area: " + rectangle.area());

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

    }

}

Output:



Lab 36:

/\*

36. Create a class Student with private member variables name and percentage.

Write methods to set, display and return values of private variables in the

Student class. Create 10 different objects of the Student class, set the values,

and display the name of the Student who has the highest percentage in the main

method of another class named StudentDemo.

\*/

class Student {

    private String name;

    private double percentage;

    // Method to set values

    public void setValues(String name, double percentage) {

        this.name = name;

        this.percentage = percentage;

    }

    // Method to display values

    public void displayValues() {

        System.out.println("Student[name=" + name + ", percentage=" + percentage + "]");

    }

    // Method to get name

    public String getName() {

        return name;

    }

    // Method to get percentage

    public double getPercentage() {

        return percentage;

    }

}

public class StudentDemo {

    public static void main(String[] args) {

        // Array to hold 10 Student objects

        Student[] students = new Student[10];

        // Initializing and setting values for 10 students

        students[0] = new Student();

        students[0].setValues("Jayaram", 88.5);

        students[1] = new Student();

        students[1].setValues("Kishor", 91.0);

        students[2] = new Student();

        students[2].setValues("Srijan", 79.3);

        students[3] = new Student();

        students[3].setValues("Nar", 85.0);

        students[4] = new Student();

        students[4].setValues("Sanjay", 93.2);

        students[5] = new Student();

        students[5].setValues("Anup", 68.5);

        students[6] = new Student();

        students[6].setValues("Anjan", 75.0);

        students[7] = new Student();

        students[7].setValues("Sonit", 89.7);

        students[8] = new Student();

        students[8].setValues("Chiranjibi", 94.1);

        students[9] = new Student();

        students[9].setValues("Dinesh", 72.5);

        // Finding the student with the highest percentage

        Student topStudent = students[0];

        for (Student student : students) {

            if (student.getPercentage() > topStudent.getPercentage()) {

                topStudent = student;

            }

        }

        // Displaying the name of the student with the highest percentage

        System.out.println("\nThe student with the highest percentage:");

        topStudent.displayValues();

    }

}

Output:



Lab 37:

/\*

37. Write a program to illustrate the concept of ArrayIndexOutOfBoundsException.

\*/

public class ArrayIndexOutOfBoundsExample {

    public static void main(String[] args) {

        try {

            // Initialize an array with 5 elements

            int[] numbers = {10, 20, 30, 40, 50};

            // Attempt to access an invalid index (out of bounds)

            System.out.println("Accessing element at index 5: " + numbers[5]);

        } catch (ArrayIndexOutOfBoundsException e) {

            // Handle the exception

            System.err.println("Exception caught: " + e);

            System.err.println("You tried to access an invalid index in the array.");

        }

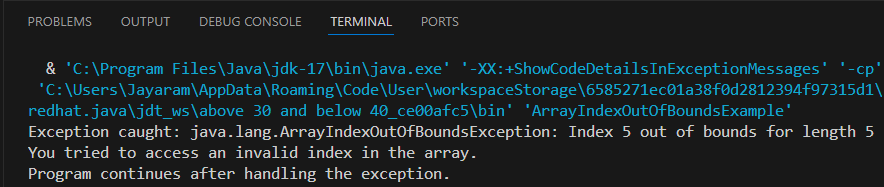
        // Program continues execution

        System.out.println("Program continues after handling the exception.");

    }

}

Output:



Lab 38:

/\*

38. Write a Java program to read data from the file “text.txt”

and write the data into the file “best.txt”.

\*/

import java.io.\*;

public class FileCopyExample {

    public static void main(String[] args) {

        // Specify the input and output file names

        String inputFile = "text.txt";

        String outputFile = "best.txt";

        // Try-with-resources to ensure proper closure of streams

        try (

            BufferedReader reader = new BufferedReader(new FileReader(inputFile));

            BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile))

        ) {

            String line;

            // Read from input file and write to output file line by line

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

                writer.write(line);

                writer.newLine(); // Add a newline after each line

            }

            System.out.println("Data successfully copied from " + inputFile + " to " + outputFile);

        } catch (FileNotFoundException e) {

            System.err.println("Error: The file " + inputFile + " was not found.");

        } catch (IOException e) {

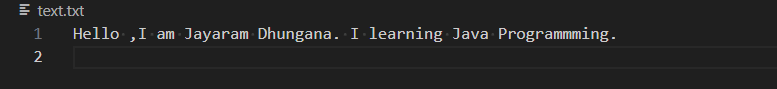
            System.err.println("Error occurred while reading or writing the file: " + e.getMessage());

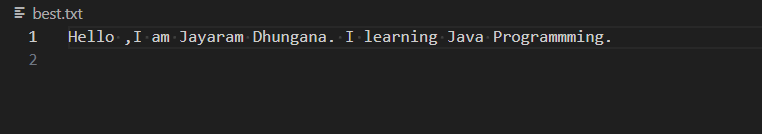
        }

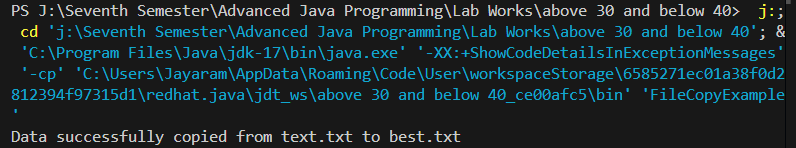
    }

}

text.txt

best.txt

and Output:



Lab 39:

/\*

39. Write a Java program writes line of text to existing file.

 Also read the content of this file and write down on monitor

\*/

import java.io.\*;

public class FileReadWriteExample {

    public static void main(String[] args) {

        String filePath = "example.txt";  // The file to read and write to

        // Writing a line of text to the file

        try (FileWriter writer = new FileWriter(filePath, true); // 'true' for

             BufferedWriter bufferedWriter = new BufferedWriter(writer)) {

            bufferedWriter.write("This is a new line of text.");

            bufferedWriter.newLine();  // To ensure a new line after the text

            System.out.println("Text written to file successfully.");

        } catch (IOException e) {

            System.err.println("Error writing to the file: " + e.getMessage());

        }

        // Reading the content of the file and printing it on the monitor

        try (FileReader reader = new FileReader(filePath);

             BufferedReader bufferedReader = new BufferedReader(reader)) {

            String line;

            System.out.println("\nContent of the file:");

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

                System.out.println(line);

            }

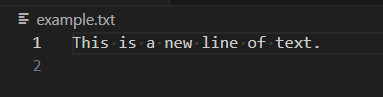
        } catch (IOException e) {

            System.err.println("Error reading from the file: " + e.getMessage());

        }

    }

}

Output:

Lab 40:

/\*40.   Write a Java program reads N names of students and then sort them in ascending order. \*/

import java.util.\*;

public class StudentNameSorter {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Input number of students

        System.out.print("Enter the number of students: ");

        int n = scanner.nextInt();

        scanner.nextLine();  // Consume the leftover newline character

        // Create an array or list to store student names

        List<String> studentNames = new ArrayList<>();

        // Input student names

        System.out.println("Enter the names of the students:");

        for (int i = 0; i < n; i++) {

            String name = scanner.nextLine();

            studentNames.add(name);

        }

        // Sort the names in ascending order

        Collections.sort(studentNames);

        // Output the sorted names

        System.out.println("\nSorted names of students:");

        for (String name : studentNames) {

            System.out.println(name);

        }

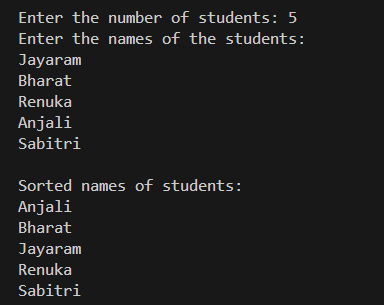
        // Close the scanner

        scanner.close();

    }

}

Output:



41:

/\*41.   Write a Simple GUI program that displays “hello World” in a

text field. The program should display if user clicks a button.

 \*/

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class HelloWorldGUI {

    public static void main(String[] args) {

        // Create the frame for the GUI

        JFrame frame = new JFrame("Hello World GUI");

        // Create a text field to display the message

        JTextField textField = new JTextField(20);

        textField.setEditable(false);  // Make the text field non-editable

        // Create a button that will trigger the action

        JButton button = new JButton("Click Me!");

        // Add an ActionListener to the button

        button.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                textField.setText("Hello World");

            }

        });

        // Set the layout of the frame

        frame.setLayout(new FlowLayout());

        // Add the button and text field to the frame

        frame.add(button);

        frame.add(textField);

        // Set default close operation and frame size

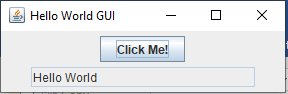
        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(300, 100);

        frame.setVisible(true);

    }

}

Output:

Lab 42:

/\*42.    Write GUI program using Swing components to find sum and

difference of two numbers. Use two text fields for giving input and

 a label for output. The program

should display sum if user presses mouse and difference if user release mouse. \*/

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class SumDifferenceCalculator {

    public static void main(String[] args) {

        // Create the frame for the GUI

        JFrame frame = new JFrame("Sum and Difference Calculator");

        // Create two text fields for user input

        JTextField num1Field = new JTextField(10);

        JTextField num2Field = new JTextField(10);

        // Create a label to display the result

        JLabel resultLabel = new JLabel("Result: ");

        // Create a panel to hold the text fields and label

        JPanel panel = new JPanel();

        panel.setLayout(new FlowLayout());

        panel.add(new JLabel("Enter number 1: "));

        panel.add(num1Field);

        panel.add(new JLabel("Enter number 2: "));

        panel.add(num2Field);

        panel.add(resultLabel);

        // Create a MouseListener to handle mouse pressed and released events

        num1Field.addMouseListener(new MouseAdapter() {

            @Override

            public void mousePressed(MouseEvent e) {

                try {

                    // Get the numbers from text fields

                    double num1 = Double.parseDouble(num1Field.getText());

                    double num2 = Double.parseDouble(num2Field.getText());

                    // Calculate and display the sum

                    double sum = num1 + num2;

                    resultLabel.setText("Sum: " + sum);

                } catch (NumberFormatException ex) {

                    resultLabel.setText("Invalid input!");

                }

            }

            @Override

            public void mouseReleased(MouseEvent e) {

                try {

                    // Get the numbers from text fields

                    double num1 = Double.parseDouble(num1Field.getText());

                    double num2 = Double.parseDouble(num2Field.getText());

                    // Calculate and display the difference

                    double difference = num1 - num2;

                    resultLabel.setText("Difference: " + difference);

                } catch (NumberFormatException ex) {

                    resultLabel.setText("Invalid input!");

                }

            }

        });

        // Set up the frame

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

        frame.setSize(300, 300);

        frame.setLayout(new FlowLayout());

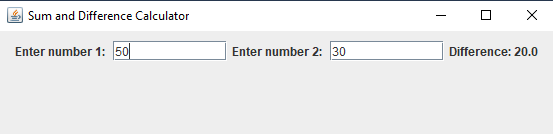
        frame.add(panel);

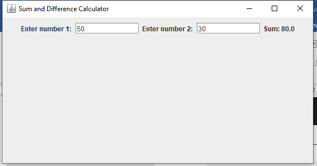
        frame.setVisible(true);

    }

}

Output:





Lab 43:

/\*43.

You are hired by a reputed software company which is going to design an application

for “Movie Rental System”. Your responsibility is to design a schema named MRS and

create a table named Movie(id, Title, Genre, Language, Length). Write a program to

design a GUI to take input for this table and insert the data into table after clicking

OK button

 \*/

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class MovieRentalSystem {

    public static void main(String[] args) {

        // Set up the GUI

        JFrame frame = new JFrame("Movie Rental System");

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

        frame.setSize(400, 300);

        // Create the form labels and text fields

        JLabel idLabel = new JLabel("Movie ID:");

        JTextField idField = new JTextField(15);

        JLabel titleLabel = new JLabel("Title:");

        JTextField titleField = new JTextField(15);

        JLabel genreLabel = new JLabel("Genre:");

        JTextField genreField = new JTextField(15);

        JLabel languageLabel = new JLabel("Language:");

        JTextField languageField = new JTextField(15);

        JLabel lengthLabel = new JLabel("Length (in minutes):");

        JTextField lengthField = new JTextField(15);

        // Create the OK button

        JButton okButton = new JButton("OK");

        // Set the layout of the frame

        frame.setLayout(new GridLayout(6, 2));

        // Add components to the frame

        frame.add(idLabel);

        frame.add(idField);

        frame.add(titleLabel);

        frame.add(titleField);

        frame.add(genreLabel);

        frame.add(genreField);

        frame.add(languageLabel);

        frame.add(languageField);

        frame.add(lengthLabel);

        frame.add(lengthField);

        frame.add(new JLabel());  // Empty label for spacing

        frame.add(okButton);

        // Action listener for the OK button

        okButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Retrieve data from text fields

                String id = idField.getText();

                String title = titleField.getText();

                String genre = genreField.getText();

                String language = languageField.getText();

                String length = lengthField.getText();

                // Check if any field is empty

                if (id.isEmpty() || title.isEmpty() || genre.isEmpty() || language.isEmpty() || length.isEmpty()) {

                    JOptionPane.showMessageDialog(frame, "All fields must be filled!", "Error", JOptionPane.ERROR\_MESSAGE);

                    return;

                }

                // Connect to the database and insert data

                try {

                    // Load the JDBC driver

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

                    // Connect to the database using XAMPP MySQL

                    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MRS", "root", "");

                    // Prepare SQL query to insert the movie details

                    String sql = "INSERT INTO Movie (id, Title, Genre, Language, Length) VALUES (?, ?, ?, ?, ?)";

                    PreparedStatement pstmt = conn.prepareStatement(sql);

                    // Set the values in the prepared statement

                    pstmt.setInt(1, Integer.parseInt(id));

                    pstmt.setString(2, title);

                    pstmt.setString(3, genre);

                    pstmt.setString(4, language);

                    pstmt.setInt(5, Integer.parseInt(length));

                    // Execute the insert statement

                    pstmt.executeUpdate();

                    // Show success message

                    JOptionPane.showMessageDialog(frame, "Movie data inserted successfully!", "Success", JOptionPane.INFORMATION\_MESSAGE);

                    // Clear the text fields

                    idField.setText("");

                    titleField.setText("");

                    genreField.setText("");

                    languageField.setText("");

                    lengthField.setText("");

                    // Close the connection

                    conn.close();

                } catch (Exception ex) {

                    JOptionPane.showMessageDialog(frame, "Error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

                    ex.printStackTrace();

                }

            }

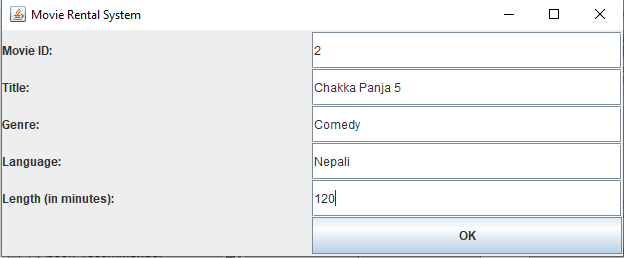
        });

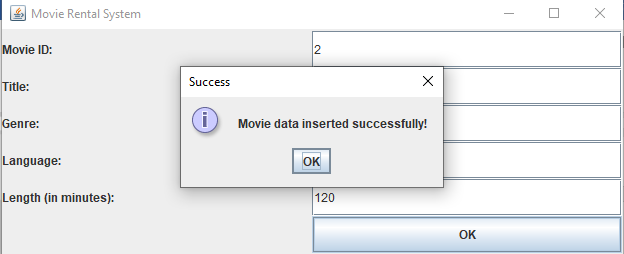
        // Make the frame visible

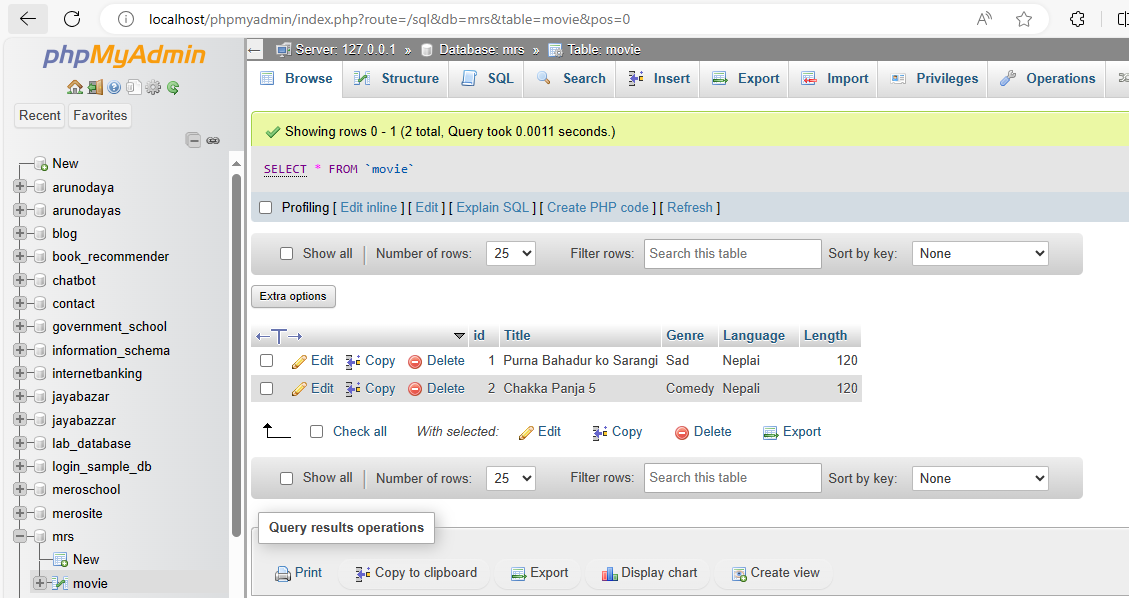
        frame.setVisible(true);

    }

}

Output:





Lab 44:

/\*44.   Write a Java program in awt to create form to enter

 employee information (eid, ename, salary, gender) \*/

import java.awt.\*;

import java.awt.event.\*;

public class EmployeeForm {

    public static void main(String[] args) {

        // Create the Frame

        Frame frame = new Frame("Employee Information Form");

        frame.setSize(400, 400);

        frame.setLayout(null);

        // Labels

        Label labelEid = new Label("Employee ID:");

        labelEid.setBounds(50, 50, 100, 20);

        frame.add(labelEid);

        Label labelEname = new Label("Employee Name:");

        labelEname.setBounds(50, 100, 100, 20);

        frame.add(labelEname);

        Label labelSalary = new Label("Salary:");

        labelSalary.setBounds(50, 150, 100, 20);

        frame.add(labelSalary);

        Label labelGender = new Label("Gender:");

        labelGender.setBounds(50, 200, 100, 20);

        frame.add(labelGender);

        // Text Fields

        TextField textEid = new TextField();

        textEid.setBounds(180, 50, 150, 20);

        frame.add(textEid);

        TextField textEname = new TextField();

        textEname.setBounds(180, 100, 150, 20);

        frame.add(textEname);

        TextField textSalary = new TextField();

        textSalary.setBounds(180, 150, 150, 20);

        frame.add(textSalary);

        // Gender Radio Buttons

        CheckboxGroup genderGroup = new CheckboxGroup();

        Checkbox male = new Checkbox("Male", genderGroup, true);

        male.setBounds(180, 200, 60, 20);

        frame.add(male);

        Checkbox female = new Checkbox("Female", genderGroup, false);

        female.setBounds(250, 200, 70, 20);

        frame.add(female);

        // Submit Button

        Button submitButton = new Button("Submit");

        submitButton.setBounds(50, 250, 80, 30);

        frame.add(submitButton);

        // Reset Button

        Button resetButton = new Button("Reset");

        resetButton.setBounds(150, 250, 80, 30);

        frame.add(resetButton);

        // TextArea to Display Output

        TextArea outputArea = new TextArea();

        outputArea.setBounds(50, 300, 300, 80);

        outputArea.setEditable(false);

        frame.add(outputArea);

        // Action Listener for Submit Button

        submitButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                String eid = textEid.getText();

                String ename = textEname.getText();

                String salary = textSalary.getText();

                String gender = genderGroup.getSelectedCheckbox().getLabel();

                if (eid.isEmpty() || ename.isEmpty() || salary.isEmpty()) {

                    outputArea.setText("Please fill out all fields.");

                } else {

                    outputArea.setText("Employee Details:\n");

                    outputArea.append("ID: " + eid + "\n");

                    outputArea.append("Name: " + ename + "\n");

                    outputArea.append("Salary: " + salary + "\n");

                    outputArea.append("Gender: " + gender + "\n");

                }

            }

        });

        // Action Listener for Reset Button

        resetButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                textEid.setText("");

                textEname.setText("");

                textSalary.setText("");

                genderGroup.setSelectedCheckbox(male);

                outputArea.setText("");

            }

        });

        // Window Closing Event

        frame.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {

                frame.dispose();

            }

        });

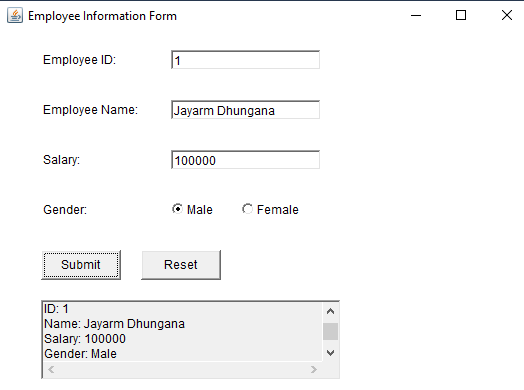
        // Set Frame Visible

        frame.setVisible(true);

    }

}

Output:



Lab 45:

/\*

 \*45. Demonstration of FlowLayout in Java

 \*/

 import java.awt.\*;

 import java.awt.event.\*;

 public class FlowLayoutDemo {

     public static void main(String[] args) {

         // Create a Frame

         Frame frame = new Frame("FlowLayout Example");

         // Set FlowLayout as the layout manager

         frame.setLayout(new FlowLayout());

         // Add some buttons to demonstrate FlowLayout

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

             Button button = new Button("Button " + i);

             frame.add(button);

         }

         // Add a WindowListener to handle window closing

         frame.addWindowListener(new WindowAdapter() {

             public void windowClosing(WindowEvent e) {

                 frame.dispose();

             }

         });

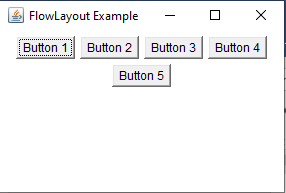
         // Set frame properties

         frame.setSize(300, 200);

         frame.setVisible(true);

     }

 }

Output:

Lab 46:

/\*

 46:Program: Demonstration of GridLayout in Java

 \*/

 import java.awt.\*;

 import java.awt.event.\*;

 public class GridLayoutDemo {

     public static void main(String[] args) {

         // Create a Frame

         Frame frame = new Frame("GridLayout Example");

         // Set GridLayout with 3 rows and 2 columns

         frame.setLayout(new GridLayout(3, 2));

         // Add some buttons to demonstrate GridLayout

         for (int i = 1; i <= 6; i++) {

             Button button = new Button("Button " + i);

             frame.add(button);

         }

         // Add a WindowListener to handle window closing

         frame.addWindowListener(new WindowAdapter() {

             public void windowClosing(WindowEvent e) {

                 frame.dispose();

             }

         });

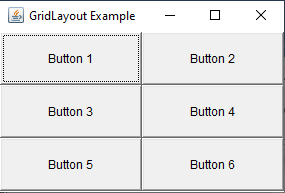
         // Set frame properties

         frame.setSize(300, 200);

         frame.setVisible(true);

     }

 }

Output:

Lab 47:

/\*

 \* 47.Adding Two Numbers Using Swing Components

 \*/

 import javax.swing.\*;

 import java.awt.\*;

 import java.awt.event.ActionEvent;

 import java.awt.event.ActionListener;

 public class AddTwoNumbers {

     public static void main(String[] args) {

         // Create a JFrame

         JFrame frame = new JFrame("Add Two Numbers");

         frame.setSize(400, 300);

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

         frame.setLayout(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns, with gaps

         // Create Labels

         JLabel labelNum1 = new JLabel("Number 1:");

         JLabel labelNum2 = new JLabel("Number 2:");

         JLabel labelResult = new JLabel("Result:");

         // Create Text Fields

         JTextField textNum1 = new JTextField();

         JTextField textNum2 = new JTextField();

         JTextField textResult = new JTextField();

         textResult.setEditable(false); // Output field should not be editable

         // Create Button

         JButton addButton = new JButton("Add");

         // Add Action Listener to the Button

         addButton.addActionListener(new ActionListener() {

             public void actionPerformed(ActionEvent e) {

                 try {

                     // Parse numbers from the text fields

                     double num1 = Double.parseDouble(textNum1.getText());

                     double num2 = Double.parseDouble(textNum2.getText());

                     // Calculate sum

                     double sum = num1 + num2;

                     // Display the result

                     textResult.setText(String.valueOf(sum));

                 } catch (NumberFormatException ex) {

                     // Handle invalid input

                     JOptionPane.showMessageDialog(frame, "Please enter valid numbers", "Error", JOptionPane.ERROR\_MESSAGE);

                 }

             }

         });

         // Add components to the frame

         frame.add(labelNum1);

         frame.add(textNum1);

         frame.add(labelNum2);

         frame.add(textNum2);

         frame.add(labelResult);

         frame.add(textResult);

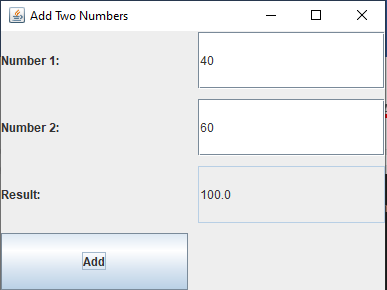
         frame.add(addButton);

         // Set frame visibility

         frame.setVisible(true);

     }

 }



Output:

Lab 48:

/\*

 \* 48.Write a Java program to retrieve and display the records of students who live

 \* in the Kathmandu district. Assume the student table in the database has four

 \*  attributes: ID, name, district,

 \*  and age. Use GUI components to display the data in tabular format.

 \*/

 import javax.swing.\*;

 import javax.swing.table.DefaultTableModel;

 import java.sql.\*;

 public class FetchKathmanduStudentsGUI {

     public static void main(String[] args) {

         // Database credentials

         String url = "jdbc:mysql://localhost:3306/mrs"; // Replace with your database name

         String user = "root"; // Replace with your database username

         String password = ""; // Replace with your database password

         // SQL query to fetch students from Kathmandu district

         String query = "SELECT \* FROM student WHERE district = 'Kathmandu'";

         // Initialize JFrame

         JFrame frame = new JFrame("Students from Kathmandu");

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

         frame.setSize(600, 400);

         // Table to display data

         String[] columnNames = {"ID", "Name", "District", "Age"};

         DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0);

         JTable table = new JTable(tableModel);

         JScrollPane scrollPane = new JScrollPane(table);

         frame.add(scrollPane);

         // JDBC objects

         Connection connection = null;

         Statement statement = null;

         ResultSet resultSet = null;

         try {

             // Load the MySQL JDBC driver

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

             // Establish connection

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

             // Create a statement object to execute the query

             statement = connection.createStatement();

             // Execute the query

             resultSet = statement.executeQuery(query);

             // Populate table model with data from the ResultSet

             while (resultSet.next()) {

                 int id = resultSet.getInt("ID");

                 String name = resultSet.getString("name");

                 String district = resultSet.getString("district");

                 int age = resultSet.getInt("age");

                 // Add row to table model

                 tableModel.addRow(new Object[]{id, name, district, age});

             }

         } catch (ClassNotFoundException e) {

             JOptionPane.showMessageDialog(frame, "JDBC Driver not found. Add the JDBC driver to the classpath.", "Error", JOptionPane.ERROR\_MESSAGE);

         } catch (SQLException e) {

             JOptionPane.showMessageDialog(frame, "SQL Exception: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

         } finally {

             // Close resources

             try {

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

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

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

             } catch (SQLException e) {

                 JOptionPane.showMessageDialog(frame, "Error closing resources: " + e.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

             }

         }

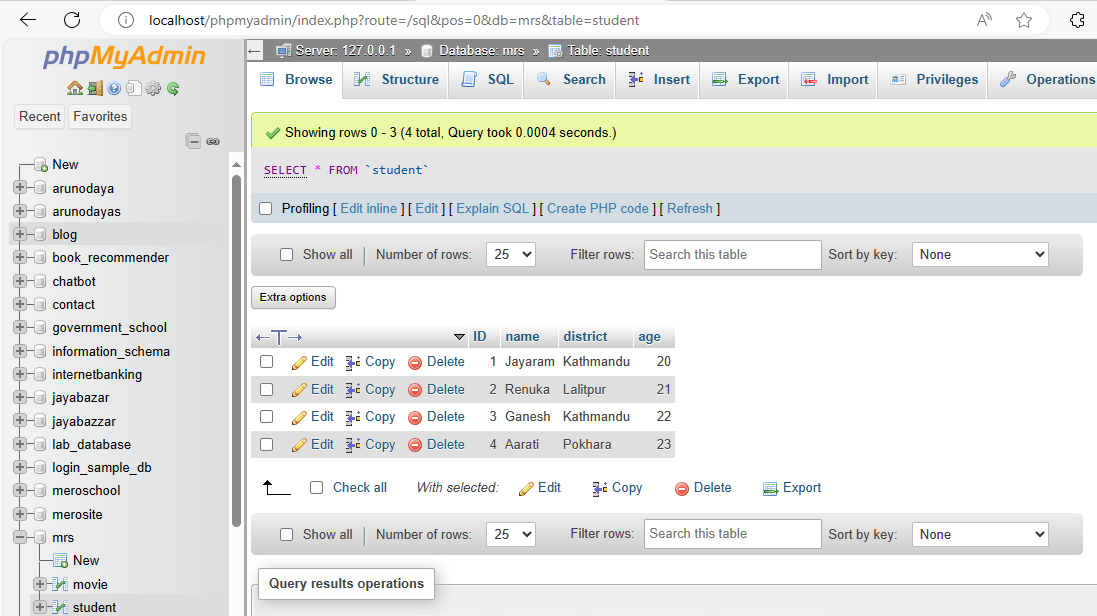
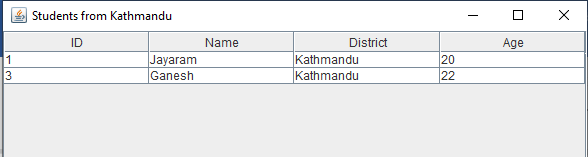
         // Make the frame visible

         frame.setVisible(true);

     }

 }

Databse Table and output are:



Lab 49:

/\*

 \* 49.  Write a Java program to insert one record to database.

 \* Assume your own database and table

 \*/

 import javax.swing.\*;

 import java.awt.\*;

 import java.awt.event.ActionEvent;

 import java.awt.event.ActionListener;

 import java.sql.\*;

 public class InsertRecordGUI {

     public static void main(String[] args) {

         // Database credentials

         String url = "jdbc:mysql://localhost:3306/mrs"; // Replace with your database name

         String user = "root"; // Replace with your database username

         String password = ""; // Replace with your database password

         // Create the JFrame

         JFrame frame = new JFrame("Insert Student Record");

         frame.setSize(400, 300);

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

         frame.setLayout(new GridLayout(5, 2, 10, 10)); // 5 rows, 2 columns with spacing

         // Create Labels and Text Fields

         JLabel labelID = new JLabel("Student ID:");

         JLabel labelName = new JLabel("Name:");

         JLabel labelDistrict = new JLabel("District:");

         JLabel labelAge = new JLabel("Age:");

         JTextField textID = new JTextField();

         JTextField textName = new JTextField();

         JTextField textDistrict = new JTextField();

         JTextField textAge = new JTextField();

         // Create Insert Button

         JButton insertButton = new JButton("Insert Record");

         // Add Action Listener to the Button

         insertButton.addActionListener(new ActionListener() {

             public void actionPerformed(ActionEvent e) {

                 // Read input values

                 int id = 0;

                 String name = textName.getText();

                 String district = textDistrict.getText();

                 int age = 0;

                 try {

                     id = Integer.parseInt(textID.getText());

                     age = Integer.parseInt(textAge.getText());

                 } catch (NumberFormatException ex) {

                     JOptionPane.showMessageDialog(frame, "ID and Age must be numbers.", "Input Error", JOptionPane.ERROR\_MESSAGE);

                     return;

                 }

                 // Database insertion

                 String query = "INSERT INTO student (ID, name, district, age) VALUES (?, ?, ?, ?)";

                 try (Connection connection = DriverManager.getConnection(url, user, password);

                      PreparedStatement preparedStatement = connection.prepareStatement(query)) {

                     // Set query parameters

                     preparedStatement.setInt(1, id);

                     preparedStatement.setString(2, name);

                     preparedStatement.setString(3, district);

                     preparedStatement.setInt(4, age);

                     // Execute the query

                     int rowsInserted = preparedStatement.executeUpdate();

                     if (rowsInserted > 0) {

                         JOptionPane.showMessageDialog(frame, "Record inserted successfully!", "Success", JOptionPane.INFORMATION\_MESSAGE);

                         // Clear input fields

                         textID.setText("");

                         textName.setText("");

                         textDistrict.setText("");

                         textAge.setText("");

                     }

                 } catch (SQLException ex) {

                     JOptionPane.showMessageDialog(frame, "Database Error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

                 }

             }

         });

         // Add components to the frame

         frame.add(labelID);

         frame.add(textID);

         frame.add(labelName);

         frame.add(textName);

         frame.add(labelDistrict);

         frame.add(textDistrict);

         frame.add(labelAge);

         frame.add(textAge);

         frame.add(new JLabel()); // Empty space

         frame.add(insertButton);

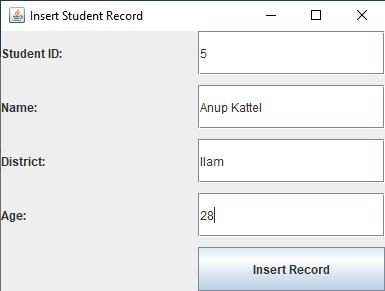
         // Make the frame visible

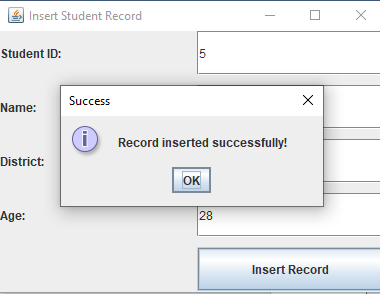
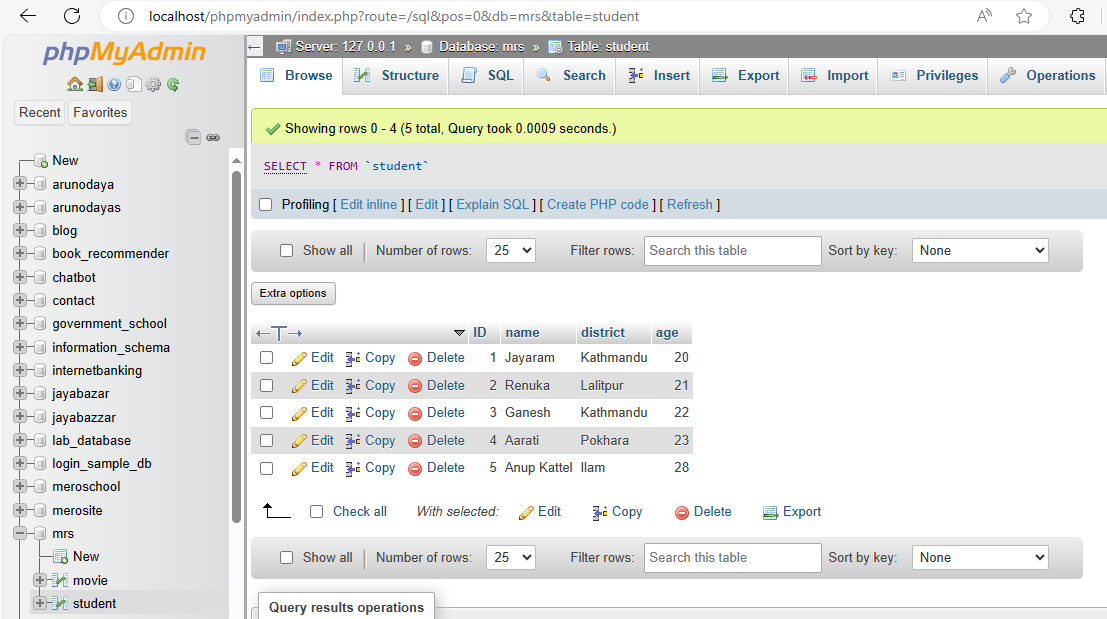
         frame.setVisible(true);

     }

 }

Output:





Lab 50:

/\*

 \*  50.  Write a Java Program to delete a record from database.

 \*  Assume your own database and table

 \*/

 import javax.swing.\*;

 import javax.swing.table.DefaultTableModel;

 import java.awt.\*;

 import java.awt.event.ActionEvent;

 import java.awt.event.ActionListener;

 import java.sql.\*;

 public class ShowAndDeleteRecords {

     public static void main(String[] args) {

         // Database credentials

         String url = "jdbc:mysql://localhost:3306/mrs"; // Replace with your database name

         String user = "root"; // Replace with your database username

         String password = ""; // Replace with your database password

         // Create the JFrame

         JFrame frame = new JFrame("Show and Delete Student Records");

         frame.setSize(600, 400);

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

         frame.setLayout(new BorderLayout());

         // Create table model and JTable

         DefaultTableModel tableModel = new DefaultTableModel(new String[]{"ID", "Name", "District", "Age"}, 0);

         JTable table = new JTable(tableModel);

         JScrollPane scrollPane = new JScrollPane(table);

         // Create Delete Button

         JButton deleteButton = new JButton("Delete Selected Record");

         // Panel for Delete Button

         JPanel buttonPanel = new JPanel();

         buttonPanel.add(deleteButton);

         // Add components to the frame

         frame.add(scrollPane, BorderLayout.CENTER);

         frame.add(buttonPanel, BorderLayout.SOUTH);

         // Load data from the database

         try (Connection connection = DriverManager.getConnection(url, user, password);

              Statement statement = connection.createStatement()) {

             // Execute query to fetch all student records

             String query = "SELECT \* FROM student";

             ResultSet resultSet = statement.executeQuery(query);

             // Populate the table model with data

             while (resultSet.next()) {

                 int id = resultSet.getInt("ID");

                 String name = resultSet.getString("name");

                 String district = resultSet.getString("district");

                 int age = resultSet.getInt("age");

                 tableModel.addRow(new Object[]{id, name, district, age});

             }

         } catch (SQLException ex) {

             JOptionPane.showMessageDialog(frame, "Database Error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

         }

         // Add Action Listener to the Delete Button

         deleteButton.addActionListener(new ActionListener() {

             public void actionPerformed(ActionEvent e) {

                 // Get selected row

                 int selectedRow = table.getSelectedRow();

                 if (selectedRow == -1) {

                     JOptionPane.showMessageDialog(frame, "Please select a record to delete.", "No Selection", JOptionPane.WARNING\_MESSAGE);

                     return;

                 }

                 // Get ID of the selected record

                 int id = (int) tableModel.getValueAt(selectedRow, 0);

                 // Delete the record from the database

                 String deleteQuery = "DELETE FROM student WHERE ID = ?";

                 try (Connection connection = DriverManager.getConnection(url, user, password);

                      PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery)) {

                     // Set query parameter

                     preparedStatement.setInt(1, id);

                     // Execute the query

                     int rowsDeleted = preparedStatement.executeUpdate();

                     if (rowsDeleted > 0) {

                         JOptionPane.showMessageDialog(frame, "Record deleted successfully!", "Success", JOptionPane.INFORMATION\_MESSAGE);

                         tableModel.removeRow(selectedRow); // Remove the row from the table

                     }

                 } catch (SQLException ex) {

                     JOptionPane.showMessageDialog(frame, "Database Error: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

                 }

             }

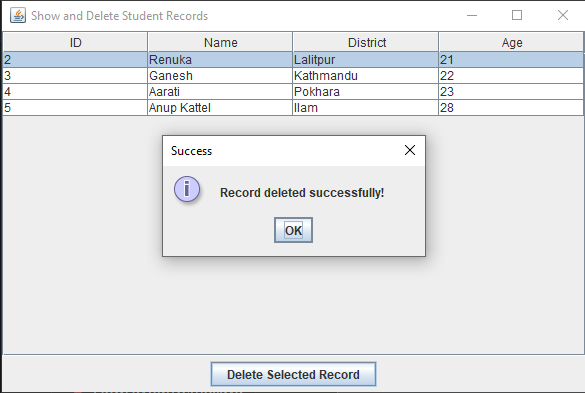
         });

         // Make the frame visible

         frame.setVisible(true);

     }

 }

Output:

51. Create a servlet that displays two text boxes in web browser, reads number entered in first text box, calculates factorial and displays it in second text field.

**Source code**

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet({"/FactorialServlet"})

public class FactorialServletDemo extends HttpServlet {

   public FactorialServletDemo() {

   }

   protected void doPost(HttpServletRequest var1, HttpServletResponse var2) throws ServletException, IOException {

      int var3 = Integer.parseInt(var1.getParameter("number"));

      int var4 = this.calculateFactorial(var3);

      var2.setContentType("text/html");

      PrintWriter var5 = var2.getWriter();

      var5.println("<html><body>");

      var5.println("<form method='post' action='FactorialServlet'>");

      var5.println("Enter a number: <input type='text' name='number' value='" + var3 + "'><br>");

      var5.println("Factorial: <input type='text' name='factorial' value='" + var4 + "' readonly><br>");

      var5.println("<input type='submit' value='Calculate'>");

      var5.println("</form>");

      var5.println("</body></html>");

   }

   private int calculateFactorial(int var1) {

      int var2 = 1;

      for(int var3 = 1; var3 <= var1; ++var3) {

         var2 \*= var3;

      }

      return var2;

   }

}

**Factorial.html**

<!DOCTYPE html>

<html>

<head>

    <title>Factorial Calculator</title>

</head>

<body>

    <form action="FactorialServlet" method="post">

        Enter a number: <input type="text" name="number"><br>

        Factorial: <input type="text" name="factorial" readonly><br>

        <input type="submit" value="Calculate">

    </form>

</body>

</html>

**Web.xml**

/webapps/Factorial/WEB-INF/classes

Copy compiled class file inside it

Accesss: <http://localhost:8080/Factorial/FactorialServlet>

<web-app>

    <servlet>

        <servlet-name>FactorialServletDemo</servlet-name>

        <servlet-class>FactorialServletDemo</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>FactorialServletDemo</servlet-name>

        <url-pattern>/FactorialServlet</url-pattern>

    </servlet-mapping>

</web-app>

**Output:**

52. Write a servlet program that reads two numbers from web browser and finds sum of these two numbers.

**Source code**

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/SumServlet")

public class SumServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)throws ServletException, IOException {

        int number1 = Integer.parseInt(request.getParameter("number1"));

        int number2 = Integer.parseInt(request.getParameter("number2"));

        int sum = number1 + number2;

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

        out.println("<html><body>");

        out.println("<form method='post' action='SumServlet'>");

        out.println("Enter first number: <input type='text' name='number1' value='" + number1 + "'><br>");

        out.println("Enter second number: <input type='text' name='number2' value='" + number2 + "'><br>");

        out.println("Sum: <input type='text' name='sum' value='" + sum + "' readonly><br>");

        out.println("<input type='submit' value='Calculate'>");

        out.println("</form>");

        out.println("</body></html>");

    }

}

**Sum.html**

<!DOCTYPE html>

<html>

<head>

    <title>Sum Calculator</title>

</head>

<body>

    <form action="SumServlet" method="post">

        Enter first number: <input type="text" name="number1"><br>

        Enter second number: <input type="text" name="number2"><br>

        Sum: <input type="text" name="sum" readonly><br>

        <input type="submit" value="Calculate">

    </form>

</body>

</html>

**Web.xml**

/webapps/SumServlet /WEB-INF/classes

Copy compiled class file inside it

Accesss: <http://localhost:8080/SumServlet/FactorialServlet>

<web-app>

    <servlet>

        <servlet-name>SumServlet</servlet-name>

        <servlet-class>SumServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>SumServlet</servlet-name>

        <url-pattern>/SumServlet</url-pattern>

    </servlet-mapping>

</web-app>

**Output:**

53. Write a JSP program display text “Apache Tomcat” 10 times.

**Source code**

#webapp/JSP/displayTomcat.jsp#

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<title>Display Text</title>

</head>

<body>

<%

for (int i = 0; i < 10; i++) {

out.println("Apache Tomcat<br>");

}

%>

</body>

</html>

**Output**

54. How exceptions can be handled in JSP scripts? Explain with suitable JSP script

Ans:>In JSP (JavaServer Pages), exceptions can be handled using the try-catch block within the JSP scriptlets. Additionally, you can use the <error-page> element in the web.xml file to define error pages for specific exceptions or error codes.

**web.xml**

<web-app>

    <error-page>

        <exception-type>java.lang.Exception</exception-type>

        <location>/errorPage.jsp</location>

    </error-page>

</web-app>

**Source code**

**HandleException.jsp**

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

    <!DOCTYPE html>

    <html>

    <head>

        <title>Exception Handling in JSP</title>

    </head>

    <body>

        <h1>Exception Handling Example</h1>

        <% try { int number=Integer.parseInt(request.getParameter("number"));

         int result=10 / number; // This may throw an ArithmeticException

            out.println("Result: " + result);

        } catch (NumberFormatException e) {

            throw new ServletException(" Error: Please enter a valid number.", e); }

            catch (ArithmeticException e) {

            throw new ServletException("Error: Division by zero is not allowed.", e); }

             catch (Exception e) { throw new ServletException("Error: An unexpected error occurred.", e); }

             %>

    </body>

    </html>

**errorPage.jsp**

<%@ page contentType="text/html;charset=UTF-8" language="java" isErrorPage="true" %>

<!DOCTYPE html>

<html>

<head>

<title>Error Page</title>

</head>

<body>

<h1>An error occurred</h1>

<p>Error message: <%= exception.getMessage() %></p>

<p>Exception type: <%= exception.getClass().getName() %></p>

</body>

</html>

**Web.xml**

<web-app>

    <error-page>

        <exception-type>java.lang.Exception</exception-type>

        <location>/errorPage.jsp</location>

    </error-page>

</web-app>

**Output**

55.Write a server and client program by using RMI such that the program finds factorial of a positive number ‘n’.

**Source code**

**FactorialInterfece.java**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface FactorialInterface extends Remote {

int calculateFactorial(int n) throws RemoteException;

}

**FactorialImpl.java**

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class FactorialImpl extends UnicastRemoteObject implements Factorial {

protected FactorialImpl() throws RemoteException {

super();

}

@Override

public int calculateFactorial(int n) throws RemoteException {

int result = 1;

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

result \*= i;

}

return result;

}

}

**FactorialServer.java**

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class FactorialServer {

public static void main(String[] args) {

try {

LocateRegistry.createRegistry(1099);

FactorialImpl factorial = new FactorialImpl();

Naming.rebind("rmi://localhost:1099/FactorialService", factorial);

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

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Factorialclient.java**

import java.rmi.Naming;

public class FactorialClient {

public static void main(String[] args) {

try {

Factorial factorial = (Factorial) Naming.lookup("rmi://localhost:1099/FactorialService");

int number = 5; // Example number

int result = factorial.calculateFactorial(number);

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

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Output:**