DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BHARATHIAR UNIVERSITY COIMBATORE – 641 046

NAME OF THE CANDIDATE : _____

REGISTER NO. : _____

M.Sc.[COMPUTER SCIENCE]

SEMESTER – I

ADAVANCED JAVA PROGRAMMING LAB

24CS1C3

SEPTEMBER – 2023

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BHARATHIAR UNIVERSITY COIMBATORE – 641 046

CERTIFICATE

This is to certify that the bonafide record work **ADVANCED JAVA PROGRAMMING LAB-24CS1C3** was done and submitted by Mr. / Ms. _____with the Register Number _____ in partial fulfillment of the requirements for the Degree, Master of Science in Computer Science at Department of Computer Science, Bharathiar University, Coimbatore – 641 046, during the period November 2023.

Submitted for Practical Examination on _____

_____ _____

Staff - In - Charge Head of the Department

_____ _____

Internal Examiner External Examiner

INDEX

1. IMPLEMENTING CALCULATOR USING SWING COMPONENTS

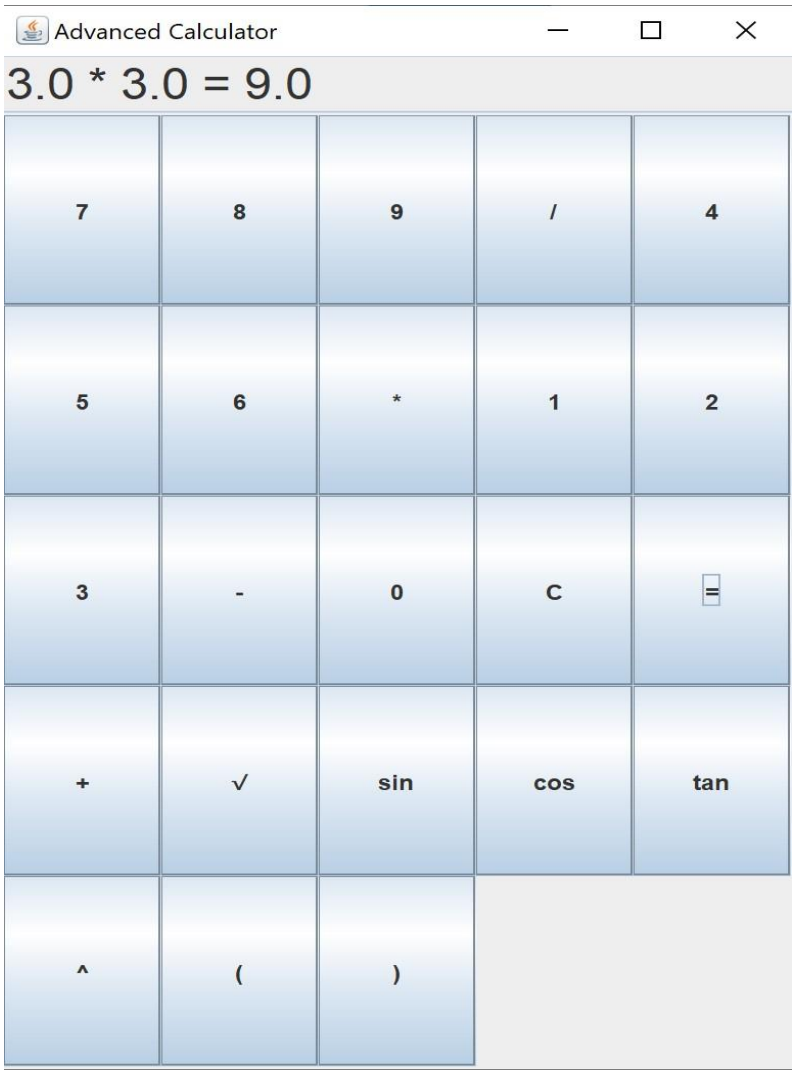
Source Code:

```
package com.mycompany.advancedcalculator;

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

public class AdvancedCalculator extends JFrame implements ActionListener {
    private JTextField display;
    private String operator;
    private double num1, num2;

    public AdvancedCalculator() {
        // Create the frame
        setTitle("Advanced Calculator");
        setSize(400, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Create the display
        display = new JTextField();
        display.setEditable(false);
        display.setFont(new Font("Arial", Font.PLAIN, 24));
        add(display, BorderLayout.NORTH);

        // Create the buttons
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(5, 4));

        String[] buttons = {
```

```
        "7", "8", "9", "/",
        "4", "5", "6", "*",
        "1", "2", "3", "-",
        "0", "C", "=", "+",
        "√", "sin", "cos", "tan",
        "^", "(", ")"
    };

    for (String text : buttons) {
        JButton button = new JButton(text);
        button.addActionListener(this);
        buttonPanel.add(button);
    }

    add(buttonPanel, BorderLayout.CENTER);
}

@Override
public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();

    if (command.charAt(0) >= '0' && command.charAt(0) <= '9') {
        display.setText(display.getText() + command);
    } else if (command.equals("C")) {
        display.setText("");
        operator = "";
    } else if (command.equals("=")) {
        num2 = Double.parseDouble(display.getText());
        display.setText(calculate(num1, num2, operator));
        operator = "";
    } else if (command.equals("√")) {
        num1 = Double.parseDouble(display.getText());
        display.setText(Double.toString(Math.sqrt(num1)));
    } else if (command.equals("sin")) {
```

```
        num1 = Double.parseDouble(display.getText());
        display.setText(Double.toString(Math.sin(Math.toRadians(num1))));
    } else if (command.equals("cos")) {
        num1 = Double.parseDouble(display.getText());
        display.setText(Double.toString(Math.cos(Math.toRadians(num1))));
    } else if (command.equals("tan")) {
        num1 = Double.parseDouble(display.getText());
        display.setText(Double.toString(Math.tan(Math.toRadians(num1))));
    } else {
        if (!operator.isEmpty()) return; // Prevent multiple operators
        num1 = Double.parseDouble(display.getText());
        operator = command;
        display.setText("");
    }
}

private String calculate(double num1, double num2, String operator) {
    return switch (operator) {
        case "+" -> Double.toString(num1 + num2);
        case "-" -> Double.toString(num1 - num2);
        case "*" -> Double.toString(num1 * num2);
        case "/" -> num2 != 0 ? Double.toString(num1 / num2) : "Error";
        case "^" -> Double.toString(Math.pow(num1, num2));
        default -> "Error";
    };
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        AdvancedCalculator calculator = new AdvancedCalculator();
        calculator.setVisible(true);
    });
} }
```

Output:



Result:

2. DEVELOPING REGISTRATION FORM USING SWING COMPONENT

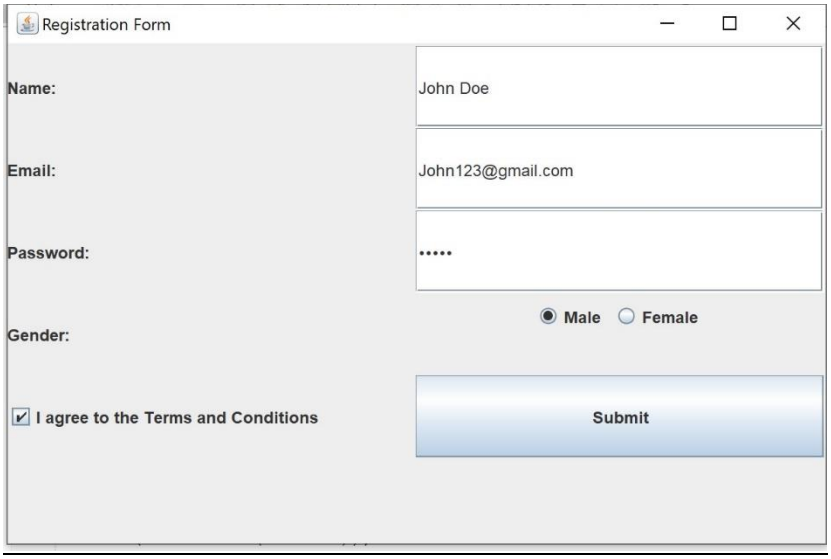
Source Code:

```
package com.mycompany.registrationform;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class RegistrationForm extends JFrame implements ActionListener {
    private JTextField nameField, emailField, passwordField;
    private JRadioButton maleRadio, femaleRadio;
    private JCheckBox termsCheckBox;
    private JButton submitButton;

    public RegistrationForm() {
        // Create the frame
        setTitle("Registration Form");
        setSize(400, 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new GridLayout(6, 2));

        // Name Field
        add(new JLabel("Name:"));
        nameField = new JTextField();
        add(nameField);

        // Email Field
        add(new JLabel("Email:"));
        emailField = new JTextField();
        add(emailField);
```

```
    // Password Field
    add(new JLabel("Password:"));
    passwordField = new JPasswordField();
    add(passwordField);

    // Gender Selection
    add(new JLabel("Gender:"));
    JPanel genderPanel = new JPanel();
    maleRadio = new JRadioButton("Male");
    femaleRadio = new JRadioButton("Female");
    ButtonGroup genderGroup = new ButtonGroup();
    genderGroup.add(maleRadio);
    genderGroup.add(femaleRadio);
    genderPanel.add(maleRadio);
    genderPanel.add(femaleRadio);
    add(genderPanel);

    // Terms and Conditions Checkbox
    termsCheckBox = new JCheckBox("I agree to the Terms and Conditions");
    add(termsCheckBox);

    // Submit Button
    submitButton = new JButton("Submit");
    submitButton.addActionListener(this);
    add(submitButton);
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == submitButton) {
        String name = nameField.getText();
        String email = emailField.getText();
        String password = passwordField.getText(); // Use getPassword() for JPasswordField
```

```
            String gender = maleRadio.isSelected() ? "Male" : "Female";
            boolean termsAccepted = termsCheckBox.isSelected();

            if (termsAccepted) {
                JOptionPane.showMessageDialog(this, "Registration Successful!\n" +
                        "Name: " + name + "\nEmail: " + email + "\nGender: " + gender);
            } else {
                JOptionPane.showMessageDialog(this, "You must agree to the terms and conditions.");
            }
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            RegistrationForm form = new RegistrationForm();
            form.setVisible(true);
        });
    }
}
```

Output:



Result:

3. CALCULATION OF FACTORIAL USING RMI

Source Code:

```
package com.mycompany.factorialrmi;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.util.Scanner;

// Remote Interface
interface FactorialService extends Remote {
    long factorial(int n) throws RemoteException;
}

// Implementation of Remote Interface
class FactorialServiceImpl extends UnicastRemoteObject implements FactorialService {
    protected FactorialServiceImpl() throws RemoteException {
        super();
    }

    @Override
    public long factorial(int n) throws RemoteException {
        if (n < 0) throw new IllegalArgumentException("Number must be non-negative");
        return (n == 0) ? 1 : n * factorial(n - 1);
    }
}

// RMI Server and Client in a Single Program
public class FactorialRMI {
    public static void main(String[] args) {
```

```
try {
        // Start the RMI registry
        LocateRegistry.createRegistry(1099);

        // Create and bind the remote object
        FactorialServiceImpl factorialService = new FactorialServiceImpl();
        Naming.rebind("FactorialService", factorialService);

        System.out.println("Factorial Service is running...");

        // Create a Scanner object for user input
        Scanner scanner = new Scanner(System.in);

        // Get a number from the user
        System.out.print("Enter a number to calculate its factorial: ");
        int number = scanner.nextInt();

        // Lookup the remote object and call the remote method
        FactorialService service = (FactorialService) Naming.lookup("FactorialService");
        long result = service.factorial(number);

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

        scanner.close();
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
}
```

Output:

```
com.mycompany.factorialrmi.FactorialRMI > main > try >
Output - Run (FactorialRMI)
skip non existing resourceDirectory C:\Users\harim\OneDrive\Documents\NetBeansProjects\FactorialRMI\src\main\resources

--- compiler:3.13.0:compile (default-compile) @ FactorialRMI ---
Recompiling the module because of changed source code.
Compiling 1 source file with javac [debug release 23] to target\classes

--- exec:3.1.0:exec (default-cli) @ FactorialRMI ---
Factorial Service is running...
Enter a number to calculate its factorial: 5
Factorial of 5 is 120
```

Result:

4. FINDING EVEN NUMBER USING RMI

Source Code:

```
package com.mycompany.evenoddrmiprogram;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.util.Scanner;

// Remote Interface
interface EvenOddService extends Remote {
    boolean isEven(int number) throws RemoteException;
}

// Implementation of Remote Interface
class EvenOddServiceImpl extends UnicastRemoteObject implements EvenOddService {
    protected EvenOddServiceImpl() throws RemoteException {
        super();
    }

    @Override
    public boolean isEven(int number) throws RemoteException {
        return number % 2 == 0;
    }
}

// RMI Server and Client in a Single Program
public class EvenOddRMIProgram {
    public static void main(String[] args) {
        try {
            // Start the RMI registry
```

```
            LocateRegistry.createRegistry(1099);

            // Create and bind the remote object
            EvenOddServiceImpl evenOddService = new EvenOddServiceImpl();
            Naming.rebind("EvenOddService", evenOddService);

            System.out.println("EvenOdd Service is running...");

            // Create a Scanner object for user input
            Scanner scanner = new Scanner(System.in);

            // Get a number from the user
            System.out.print("Enter a number to check if it is even or odd: ");
            int number = scanner.nextInt();

            // Lookup the remote object and call the remote method
            EvenOddService service = (EvenOddService) Naming.lookup("EvenOddService");
            boolean isEven = service.isEven(number);

            // Print the result
            if (isEven) {
                System.out.println(number + " is even.");
            } else {
                System.out.println(number + " is odd.");
            }

            scanner.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Output - Run (EvenOddRMIProgram)
skip non existing resourceDirectory C:\Users\harim\OneDrive\Documents\NetBeansProjects\EvenOddRMIProgram\src\main\resources

--- compiler:3.13.0:compile (default-compile) @ EvenOddRMIProgram ---
Recompiling the module because of changed source code.
Compiling 1 source file with javac [debug release 23] to target\classes

--- exec:3.1.0:exec (default-cli) @ EvenOddRMIProgram ---
EvenOdd Service is running...
Enter a number to check if it is even or odd: 2
2 is even.
```

Result:

5. EMPLOYEE DETAILS- JDBC

Source Code:

```
package jdbc;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Jdbc {
    public static void main(String[] args) {
        // JDBC URL for UCanAccess
        String jdbcUrl = "jdbc:ucanaccess://C:\\Users\\24CS\\Documents\\employees.accdb";

        // JDBC variables for opening, managing, and closing the connection
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;

        try {
            // Establishing the connection
            connection = DriverManager.getConnection(jdbcUrl);

            // Creating a Statement object to execute the query
            statement = connection.createStatement();

            // Executing a query
            String sql = "SELECT id, name, position FROM employees";
            resultSet = statement.executeQuery(sql);

            // Processing the results
```

```
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                String position = resultSet.getString("position");

                System.out.printf("ID: %d, Name: %s, Position: %s%n", id, name, position);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // Closing the resources
            try {
                if (resultSet != null) resultSet.close();
                if (statement != null) statement.close();
                if (connection != null) connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output:

Result:

6. MANIPULATION OF STUDENT DETAILS- JDBC

Source Code:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class StudentManagement {
    // JDBC URL for UCanAccess
    private static final String JDBC_URL = "jdbc:ucanaccess://C:\\Users\\24CS\\Documents\\students.accdb";

    public static void main(String[] args) {
        // JDBC variables for opening, managing, and closing the connection
        Connection connection = null;
        Statement statement = null;
        Scanner scanner = new Scanner(System.in);

        try {
            // Establishing the connection
            connection = DriverManager.getConnection(JDBC_URL);
            statement = connection.createStatement();

            // Menu for user interaction
            while (true) {
                System.out.println("Choose an option:");
                System.out.println("1. Add Student");
                System.out.println("2. Update Student");
                System.out.println("3. Delete Student");
                System.out.println("4. View All Students");
                System.out.println("5. Exit");
```

```
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
            case 1: // Add Student
                System.out.print("Enter student name: ");
                String name = scanner.nextLine();
                System.out.print("Enter student age: ");
                int age = scanner.nextInt();
                String insertSQL = String.format("INSERT INTO students (name, age) VALUES ('%s', %d)", name,
age);
                statement.executeUpdate(insertSQL);
                System.out.println("Student added successfully.");
                break;

            case 2: // Update Student
                System.out.print("Enter student ID to update: ");
                int updateId = scanner.nextInt();
                scanner.nextLine(); // Consume newline
                System.out.print("Enter new name: ");
                String newName = scanner.nextLine();
                System.out.print("Enter new age: ");
                int newAge = scanner.nextInt();
                String updateSQL = String.format("UPDATE students SET name='%s', age=%d WHERE id=%d",
newName, newAge, updateId);
                statement.executeUpdate(updateSQL);
                System.out.println("Student updated successfully.");
                break;

            case 3: // Delete Student
                System.out.print("Enter student ID to delete: ");
                int deleteId = scanner.nextInt();
                String deleteSQL = String.format("DELETE FROM students WHERE id=%d", deleteId);
```

```
                statement.executeUpdate(deleteSQL);
                System.out.println("Student deleted successfully.");
                break;

            case 4: // View All Students
                String selectSQL = "SELECT id, name, age FROM students";
                ResultSet resultSet = statement.executeQuery(selectSQL);
                System.out.println("ID\tName\tAge");
                while (resultSet.next()) {
                    int id = resultSet.getInt("id");
                    String studentName = resultSet.getString("name");
                    int studentAge = resultSet.getInt("age");
                    System.out.printf("%d\t%s\t%d%n", id, studentName, studentAge);
                }
                break;

            case 5: // Exit
                System.out.println("Exiting...");
                scanner.close();
                return;

            default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    // Closing the resources
    try {
        if (statement != null) statement.close();
        if (connection != null) connection.close();
```

```
  } catch (SQLException e) {
            e.printStackTrace();
        }
      }
    }
  }
```

Output:

Result:

7.Employee Details using XML:

Source Code:

```
package com.mycompany.mavenproject4;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class EmployeeDetailsForm {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Get employee details from user
        System.out.print("Enter Employee ID: ");
        String id = scanner.nextLine();

        System.out.print("Enter Employee Name: ");
        String name = scanner.nextLine();

        System.out.print("Enter Employee Position: ");
        String position = scanner.nextLine();

        System.out.print("Enter Employee Department: ");
        String department = scanner.nextLine();

        // Create XML file
        String xmlContent = createXml(id, name, position, department);
        saveToFile("employee.xml", xmlContent);
```

```
    // Generate HTML file
    String htmlContent = createHtml(id, name, position, department);
    saveToFile("employee.html", htmlContent);

    // Open HTML file in the default browser
    openInBrowser("employee.html");

    scanner.close();
}

private static String createXml(String id, String name, String position, String department) {
    return "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n" +
        "<employees>\n" +
        "    <employee>\n" +
        "        <id>" + id + "</id>\n" +
        "        <name>" + name + "</name>\n" +
        "        <position>" + position + "</position>\n" +
        "        <department>" + department + "</department>\n" +
        "    </employee>\n" +
        "</employees>";
}

private static String createHtml(String id, String name, String position, String department) {
    return "<html>\n" +
        "<head><title>Employee Details</title></head>\n" +
        "<body>\n" +
        "<h1>Employee Details</h1>\n" +
        "<p>ID: " + id + "</p>\n" +
        "<p>Name: " + name + "</p>\n" +
        "<p>Position: " + position + "</p>\n" +
        "<p>Department: " + department + "</p>\n" +
        "</body>\n" +
        "</html>";
}
```

```
    private static void saveToFile(String fileName, String content) {
        try (FileWriter fileWriter = new FileWriter(new File(fileName))) {
            fileWriter.write(content);
            System.out.println(fileName + " created successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void openInBrowser(String fileName) {
        try {
            File htmlFile = new File(fileName);
            if (htmlFile.exists()) {
                String filePath = htmlFile.getAbsolutePath();
                // Open the HTML file in the default web browser
                if (System.getProperty("os.name").toLowerCase().contains("win")) {
                    Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + filePath);
                } else if (System.getProperty("os.name").toLowerCase().contains("mac")) {
                    Runtime.getRuntime().exec("open " + filePath);
                } else {
                    Runtime.getRuntime().exec("xdg-open " + filePath);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Employee.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- employee.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<employees>
  <employee>
    <id>1</id>
    <name>John Doe</name>
    <position>Software Engineer</position>
    <department>IT</department>
  </employee>
</employees>
```

Output:

```
Output - Run (EmployeeDetailsForm)
--- exec:3.1.0:exec (default-cli) @ mavenproject4 ---
Enter Employee ID: 001
Enter Employee Name: John Doe
Enter Employee Position: Team Leader
Enter Employee Department: Research and Development
employee.xml created successfully.
employee.html created successfully.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  39.239 s
```

Employee Details

ID: 001

Name: John Doe

Position: Team Leader

Department: Research and Development

Result:

8.Cylinder Shape using HTML

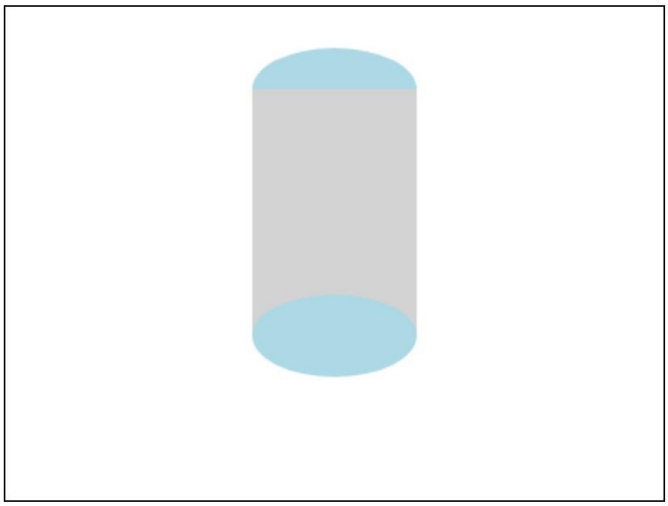
Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Cylinder Drawing</title>
   <style>
      canvas {
         border: 1px solid #000;
      }
   </style>
</head>
<body>
   <h1>Draw a Cylinder</h1>
   <canvas id="cylinderCanvas" width="400" height="300"></canvas>

   <script>
      const canvas = document.getElementById('cylinderCanvas');
      const ctx = canvas.getContext('2d');

      function drawCylinder(x, y, radius, height) {
         // Draw the top ellipse
         ctx.beginPath();
         ctx.ellipse(x, y, radius, radius / 2, 0, 0, Math.PI * 2);
         ctx.fillStyle = 'lightblue';
         ctx.fill();
         ctx.closePath();
```

```
        // Draw the sides
        ctx.fillStyle = 'lightgray';
        ctx.fillRect(x - radius, y, radius * 2, height);

        // Draw the bottom ellipse
        ctx.beginPath();
        ctx.ellipse(x, y + height, radius, radius / 2, 0, 0, Math.PI * 2);
        ctx.fillStyle = 'lightblue';
        ctx.fill();
        ctx.closePath();
    }

    // Draw the cylinder at specified coordinates
    drawCylinder(200, 50, 50, 150);
  </script>
</body>
</html>
```

Output:

Draw a Cylinder



Result:

9.Multiplication Table using Javascript

Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Multiplication Table</title>
  <style>
    body {
      display: flex;
      flex-direction: column;
      align-items: center;
      margin: 0;
      padding: 20px;
      font-family: Arial, sans-serif;
    }
    table {
      border-collapse: collapse;
      margin-top: 20px;
    }
    table, th, td {
      border: 1px solid #333;
      padding: 10px;
      text-align: center;
    }
    th {
      background-color: #f0f0f0;
    }
  </style>
</head>
<body>
```

```
<h1>Multiplication Table</h1>
<label for="number">Choose a number:</label>
<input type="number" id="number" value="1" min="1" max="20">
<button id="generate">Generate Table</button>

<table id="multiplicationTable">
  <thead>
    <tr>
      <th>Multiplier</th>
      <th>Result</th>
    </tr>
  </thead>
  <tbody>
    <!-- Table rows will be inserted here -->
  </tbody>
</table>

<script>
  document.getElementById('generate').addEventListener('click', function() {
    const number = parseInt(document.getElementById('number').value);
    const tableBody = document.querySelector('#multiplicationTable tbody');
    tableBody.innerHTML = ''; // Clear previous results

    for (let i = 1; i <= 10; i++) {
      const row = document.createElement('tr');
      const multiplierCell = document.createElement('td');
      const resultCell = document.createElement('td');

      multiplierCell.textContent = `${number} x ${i}`;
      resultCell.textContent = number * i;

      row.appendChild(multiplierCell);
      row.appendChild(resultCell);
      tableBody.appendChild(row);
```

```
            }
        });
    </script>
</body>
</html>
```

Output:

Multiplication Table

Choose a number:

1

Generate Table

Multiplier	Result
1 x 1	1
1 x 2	2
1 x 3	3
1 x 4	4
1 x 5	5
1 x 6	6
1 x 7	7
1 x 8	8
1 x 9	9
1 x 10	10

Result:

10.Javascript to sort array of numbers

Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Sort Array of Numbers</title>
</head>
<body>
   <h1>Sort Array of Numbers</h1>
   <input type="text" id="numberArray" placeholder="Enter numbers separated by commas">
   <button id="sortAscending">Sort Ascending</button>
   <button id="sortDescending">Sort Descending</button>

   <h2>Sorted Array:</h2>
   <p id="result"></p>

   <script>
      document.getElementById('sortAscending').addEventListener('click', function() {
         const input = document.getElementById('numberArray').value;
         const numberArray = input.split(',').map(num => parseFloat(num.trim()));
         const sortedArray = numberArray.sort((a, b) => a - b);
         document.getElementById('result').textContent = sortedArray.join(', ');
      });

      document.getElementById('sortDescending').addEventListener('click', function() {
         const input = document.getElementById('numberArray').value;
         const numberArray = input.split(',').map(num => parseFloat(num.trim()));
         const sortedArray = numberArray.sort((a, b) => b - a);
         document.getElementById('result').textContent = sortedArray.join(', ');
      });
```

```
    </script>
</body>
</html>
```

Output:

Sort Array of Numbers

3,1,4,5,9,2 | Sort Ascending | Sort Descending

Sorted Array:

9, 5, 4, 3, 2, 1

Result:

11.Java Servlet -Handling input using GET Method

Source Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Form Handling</title>
</head>
<body>
   <h1>Form Handling Example</h1>

   <form action="FormServlet" method="get">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" required><br><br>
      <input type="submit" value="Submit (GET)">
   </form>

   <form action="FormServlet" method="post">
      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required><br><br>
      <input type="submit" value="Submit (POST)">
   </form>
</body>
</html>
```

NewServlet.java

```
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(name = "FormServlet", urlPatterns = {"/FormServlet"}) public class FormServlet extends HttpServlet {


@Override

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

processRequest(request, response);

}


@Override

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

processRequest(request, response);

}


private void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException { response.setContentType("text/html;charset=UTF-8");


try (PrintWriter out = response.getWriter()) { String name = request.getParameter("name"); String email = request.getParameter("email");


out.println("<html>"); out.println("<head>");

out.println("<title>Form Handling Results</title>"); out.println("</head>");

out.println("<body>");

out.println("<h1>Form Handling Results</h1>");


if (name != null) {

out.println("<p>Name: " + name + "</p>");

}
```

```
if (email != null) {
out.println("<p>Email: " + email + "</p>");
}

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

Output:

Enter Your Name:

Nandhini Submit

Input Received via GET Method

Hello, Nandhini!

Result:

12.JAVA SERVLET USING HTML FORM TO ACCEPT DATA, GET AND POST METHODS

Source Code:

```
<!DOCTYPE html>
<html>
<head>
<title>Form Handling</title>
</head>
<body>
<h1>Form Handling Example</h1>
<form action="FormServlet" method="get">
<label for="name">Name:</label>
<input type="text" id="name" name="name"><br><br>
<input type="submit" value="Submit (GET)">
</form>
<form action="FormServlet" method="post">
<label for="email">Email:</label>
<input type="text" id="email" name="email"><br><br>
<input type="submit" value="Submit (POST)">
</form>
</body>
</html>
```

Newservlet.java

```
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(name = "FormServlet", urlPatterns = {"/FormServlet"}) public class FormServlet extends HttpServlet {


@Override

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

processRequest(request, response);

}


@Override

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

processRequest(request, response);

}


private void processRequest(HttpServletRequest request, HttpServletResponse response)


throws ServletException, IOException { response.setContentType("text/html;charset=UTF-8");


try (PrintWriter out = response.getWriter()) { String name = request.getParameter("name"); String email = request.getParameter("email");


out.println("<html>"); out.println("<head>");

out.println("<title>Form Handling Results</title>"); out.println("</head>");

out.println("<body>");

out.println("<h1>Form Handling Results</h1>");


if (name != null) {

out.println("<p>Name: " + name + "</p>");

}


if (email != null) {

out.println("<p>Email: " + email + "</p>");

}
```

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

Output:

Form Servlet Demo

Name: Nandhini
Email: nandhini@gmail.com
Submit

Form Data Received

Name: Nandhini

Email: nandhini@gmail.com

Result:

13.Java Servlet to display Information

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(name = "InfoServlet", urlPatterns = {"/InfoServlet"})
public class InfoServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");

        try (PrintWriter out = response.getWriter()) {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Information Display</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Information Display</h1>");
            out.println("<p>Welcome to the Information Servlet!</p>");
```

```
            out.println("<h2>Details:</h2>");
            out.println("<ul>");
            out.println("<li>Name: John Doe</li>");
            out.println("<li>Email: johndoe@example.com</li>");
            out.println("<li>Position: Software Engineer</li>");
            out.println("</ul>");
            out.println("</body>");
            out.println("</html>");
        }
    }
}
```

Output:

Information Display

Welcome to the Information Page!

Details:

Name: John Doe

Email: johndoe@example.com

Position: Software Engineer

Result:

14.JSP – Generating Factorial Number

Source Code:

```
<html>
<body>
<form action="Factorial.jsp">
Enter a value for n: <input type="text" name="val">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

- - - -

```
<html>
<body>
<%!
long n, result; String str;

long fact(long n) { if(n==0)
return 1; else
return n*fact(n-1);
}
%>
<%
str = request.getParameter("val"); n = Long.parseLong(str);
result = fact(n);
%>
<b>Factorial value: </b> <%= result %>
```

```
</body>
</html>
```

Output:

Result:

15.Deploying and Testing the Sample Web Application using JSP

Source Code:

Output:

Result: