Q1. Create a webpage using java and jsp that prints your name to the screen, print your name in Tahoma font, print a definition list with 5 items, Create links to five different pages, etc.

Create a JSP file (e.g., index.jsp) in the "jsp" folder with the following content:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Web Page</title>
  <style>
    body {
      font-family: Tahoma, Arial, sans-serif;
    }
  </style>
</head>
<body>
  <h1>Welcome to My Web Page</h1>
  <p>My name is Your Name</p>

  <h2>Definition List</h2>
  <dl>
    <dt>Item 1</dt>
    <dd>Definition for Item 1</dd>
    <dt>Item 2</dt>
    <dd>Definition for Item 2</dd>
    <dt>Item 3</dt>
    <dd>Definition for Item 3</dd>
    <dt>Item 4</dt>
    <dd>Definition for Item 4</dd>
    <dt>Item 5</dt>
    <dd>Definition for Item 5</dd>
  </dl>

  <h2>Links to Other Pages</h2>
  <ul>
    <li><a href="page1.jsp">Page 1</a></li>
```

```
        <li><a href="page2.jsp">Page 2</a></li>
        <li><a href="page3.jsp">Page 3</a></li>
        <li><a href="page4.jsp">Page 4</a></li>
        <li><a href="page5.jsp">Page 5</a></li>
    </ul>
</body>
</html>
```

2. simple java Program to demonstrate Swing components.

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

public class SwingDemo {
    public static void main(String[] args) {
        // Create a JFrame (the main window)
        JFrame frame = new JFrame("Swing Demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        frame.setLayout(new FlowLayout());

        // Create a label
        JLabel label = new JLabel("This is a JLabel");
        frame.add(label);

        // Create a text field
        JTextField textField = new JTextField(20);
        frame.add(textField);

        // Create a button
        JButton button = new JButton("Click Me");
        frame.add(button);
```

```
// Create a checkbox
JCheckBox checkBox = new JCheckBox("Check Me");
frame.add(checkBox);

// Create a radio button
JRadioButton radioButton1 = new JRadioButton("Option 1");
JRadioButton radioButton2 = new JRadioButton("Option 2");
ButtonGroup radioGroup = new ButtonGroup();
radioGroup.add(radioButton1);
radioGroup.add(radioButton2);
frame.add(radioButton1);
frame.add(radioButton2);

// Create a combo box
String[] options = {"Option 1", "Option 2", "Option 3"};
JComboBox<String> comboBox = new JComboBox<>(options);
frame.add(comboBox);

// Create a list
String[] listData = {"Item 1", "Item 2", "Item 3", "Item 4"};
JList<String> list = new JList<>(listData);
frame.add(list);

// Create a text area
JTextArea textArea = new JTextArea(5, 20);
frame.add(textArea);

// Create an event listener for the button
button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
            String inputText = textField.getText();
            JOptionPane.showMessageDialog(frame, "You clicked the button!\nInput Text: " + inputText);
          }
        });

        // Display the frame
        frame.setVisible(true);
    }
}
```

4. Write a java program that connects to a database using JDBC and does add, delete and retrieve operations.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DatabaseOperations {
    // JDBC URL, username, and password of MySQL server
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/your_database_name";
    private static final String JDBC_USER = "your_username";
    private static final String JDBC_PASSWORD = "your_password";

    public static void main(String[] args) {
        Connection connection = null;
        try {
            // Establish a connection to the database
            connection = DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASSWORD);

            // Add a new student record
            addStudent(connection, "John Doe", 25);
        // Retrieve and print all student records
        retrieveStudents(connection);

        // Delete a student record
        deleteStudent(connection, 1);

        // Retrieve and print the updated list of student records
        retrieveStudents(connection);
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

private static void addStudent(Connection connection, String name, int age) throws SQLException {
    String insertQuery = "INSERT INTO students (name, age) VALUES (?, ?)";
    try (PreparedStatement preparedStatement = connection.prepareStatement(insertQuery)) {
        preparedStatement.setString(1, name);
        preparedStatement.setInt(2, age);
        preparedStatement.executeUpdate();
        System.out.println("Student added successfully.");
    }
}

private static void retrieveStudents(Connection connection) throws SQLException {
        String selectQuery = "SELECT id, name, age FROM students";
        try (PreparedStatement preparedStatement = connection.prepareStatement(selectQuery);
             ResultSet resultSet = preparedStatement.executeQuery()) {
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                int age = resultSet.getInt("age");
                System.out.println("ID: " + id + ", Name: " + name + ", Age: " + age);
            }
        }
    }

    private static void deleteStudent(Connection connection, int studentId) throws SQLException {
        String deleteQuery = "DELETE FROM students WHERE id = ?";
        try (PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery)) {
            preparedStatement.setInt(1, studentId);
            preparedStatement.executeUpdate();
            System.out.println("Student with ID " + studentId + " deleted successfully.");
        }
    }
}
```

1. Write a client-server program which displays the server machine's date and time on the client machine.

To create a simple client-server program using Java and JSP that displays the server machine's date and time on the client machine, you can follow these steps:

Server-Side (Java) Program:

1. Create a Java web application using a Java servlet to fetch the server's date and time and send it to the client. Here's a simple example:

~~~
   ```java
   import java.io.*;
~~~

```java
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;

public class DateTimeServlet extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");

    // Get the current date and time
    Date serverDateTime = new Date();

    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Server Date and Time</title></head>");
    out.println("<body>");
    out.println("<h1>Server Date and Time</h1>");
    out.println("<p>The server's date and time is: " + serverDateTime + "</p>");
    out.println("</body>");
    out.println("</html>");
  }
}
```

2. Compile and deploy this servlet on a servlet container like Apache Tomcat.

Client-Side (JSP) Program:

1. Create a JSP page (e.g., `client.jsp`) that will call the server-side servlet and display the response.

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

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Client Page</title>
</head>
<body>
  <h1>Client Page</h1>
  <p>This page requests the server's date and time:</p>
  <p><a href="DateTimeServlet">Get Server Date and Time</a></p>
</body>
</html>
```

2. Deploy the JSP page in the same web application as the servlet.

Accessing the Application:

1. Start your servlet container (e.g., Apache Tomcat).

2. Access the client JSP page by navigating to its URL, typically something like `http://localhost:8080/YourWebAppName/client.jsp`.

3. Click the "Get Server Date and Time" link on the client JSP page. This link will call the `DateTimeServlet`, which will retrieve the server's date and time and display it on the client's browser.

This client-server program demonstrates how to use a Java servlet to fetch the server's date and time and display it on a JSP client page.

3. Configure Apache Tomcat and write a hello world jsp page

Step 1: download latest Apache tomcat server

Step 2: create new project in Netbeans and click 'add' in 'server and settings'.

Step 3: Select 'Apache Tomcat' & click Next.

Step 4: Select the location where you installed the server. Set username & password & click save settings.

Hello.jsp

```
<html>
<head>
    <title> Hello </title>
</head>
<body>
    <h1> Hello World </h1>
</body>
</html>
```

web.xml

```
<web-app>
    <Hello-file-list>
        <Hello-file> Hello.jsp </Hello-file>
    </Hello-file-list>
</web-app>
```