UNIVERSITI MALAYSIA TERENGGANU

CSM3023 WEB-BASED APPLICATION DEVELOPMENT (K1)

BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH HONORS

LAB 3

SEMESTER II 2023/2024

Prepared for:

DR. MOHAMAD NOR HASSAN

Prepared by:

LUQMAN HAKIM BIN AZIZ

(S66292)

Week 2

JSP: Scriplet, Page Directive & Include Directive

Web Programming 2

Name: Luqman Hakim BiN Aziz
Matric #: S66292
Semester: II 2023/2024
Lab: MP1
Demonstrator: Sir Arizal

Lecturers
PUSAT PENGAJIAN INFORMATIK DAN MATEMATIK GUNAAN (PPIMG), UNIVERSITI MALAYSIA TERENGGANU (UMT)

Revision History

Revision Date	Previous Revision Date	Summary of Changes	Changes Marked
		First Issue	Mohamad Nor Hassan
		Second Issue	Dr Rabiei Mamat Dr Faizah Aplop Dr Fouad Ts Dr Rosmayati Mohemad Fakhrul Adli Mohd Zaki
21/02/2019		Addition of Revision History, Table of Contents, Formatting Cover Page	Fakhrul Adli Mohd Zaki

Table of Contents

Arahan:

Manual makmal ini adalah untuk kegunaan pelajar-pelajar Pusat Pengajian Informatik dan Matematik Gunaan (PPIMG), Universiti Malaysia Terengganu (UMT) sahaja. Tidak dibenarkan mencetak dan mengedar manual ini tanpa kebenaran rasmi daripada penulis.

Sila ikuti langkah demi langkah sebagaimana yang dinyatakan di dalam manual. Tandakan (√) setiap langkah yang telah selesai dibuat dan tulis kesimpulan bagi setiap aktiviti yang telah selesai dijalankan.

Instruction:

This laboratory manual is for use by the students of the School of Informatics and Applied Mathematics (PPIMG), Universiti Malaysia Terengganu (UMT) only. It is not permissible to print and distribute this manual without the official authorisation of the author.

Please follow step by step as described in the manual. Tick (√) each step completed and write the conclusions for each completed activity.

Task 1: Passing Data from Main JSP's Page to Other JSP's Page

Objective : To demonstrate the use of *request.getParameter ("fieldName")* for passing input from one JSP'page to another JSP's page.

Problem : i. Create a page *memberRegister.jsp*.

Description ii. Page *memberRegister.jsp* consists of two (2) inputs;

- IC No (Must be in pre-formatted XXXXXXXXXX)
- Name

3. In *memberRegister.jsp*, include two (2) buttons;

   Submit and *Cancel* button.

4. Create a page *memberProcessing.jsp*.

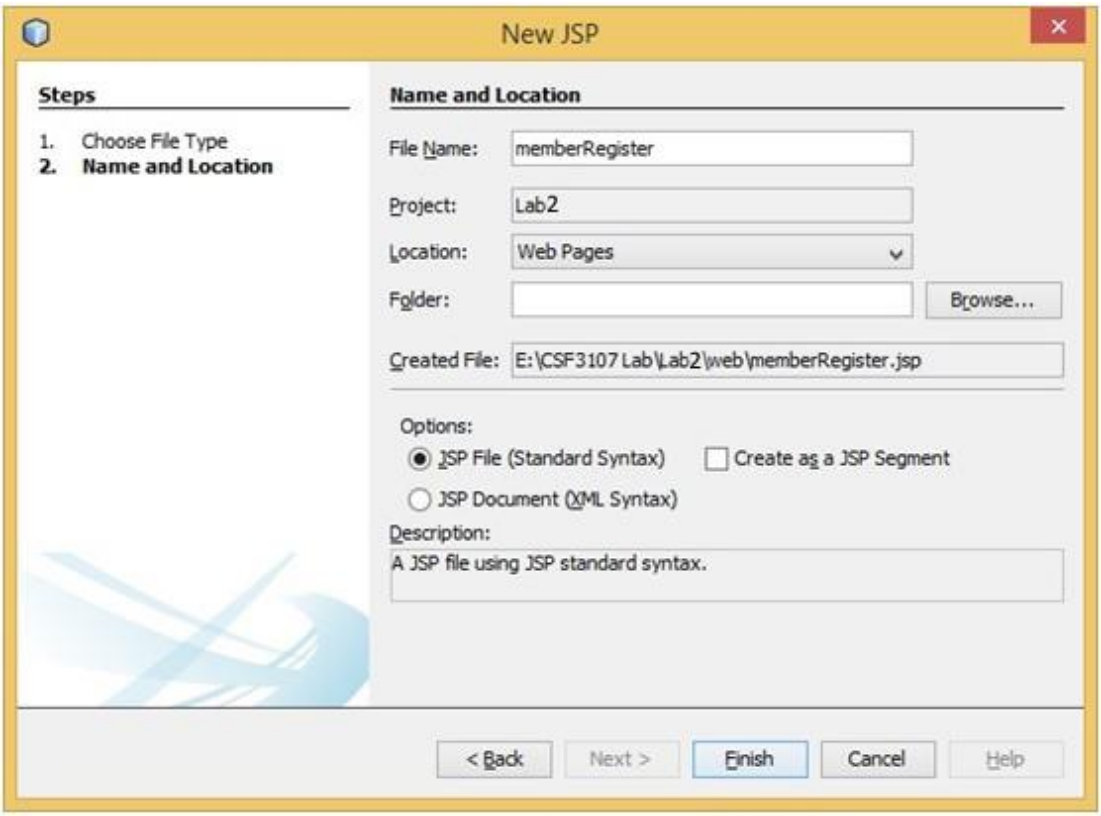
5. When user click *Submit* button, process the request and display the input key-in in *memberProcessing.jsp* page.

Estimated time : 30 minutes

1. Create new Project namely *Lab2*.

2. To create a JSP's page, right click *Lab2* -> *New* -> *JSP*.

3. Key-in File Name: *memberRegister*.



4. Click *Finish* button.

5. Source code for *memberRegister.jsp* will appear.

6. Write a HTML's markup to produce HTML's form

```
<body>
    <h1>Passing data from main JSP's page to other JSP's page </h1>
      <form id="memberFrm" action="memberProcessing.jsp" method="post" onsubmit="return checkICNo()">
            <fieldset>
              <legend>Member Registration</legend>
                    <label for="invoiceno">Ic No *</label>
                    <input type="text" id="icno" name="my_icno" size="15" placeholder="E.g. 921012101245"><br/>

                    <label for="name">Name</label>
                    <input type="text" id="name" name="my_name" size="45" placeholder="Key-in your name"><br/>

                    <p><input type="submit" id="btnSubmit" value="Submit"/>
                       <input type="reset" id="btnCancel" value="Cancel"/>
                    </p>
            </fieldset>
      </form>
</body>
```

7. Save and compile *memberRegister.jsp* file.

8. Run the *memberRegister.jsp* file and you should get the interface as below:

9. Repeat step 1 and step 2.

10. Key-in File Name: *memberProcessing*.

11. Click *Finish* button.

12. Source code for *memberProcessing.jsp* will appear.

13. Write a HTML code.

```
8   <!DOCTYPE html>
9   <html>
10      <head>
11          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12          <title>Lab 6 - Task 6</title>
13      </head>
14      <body>
15          <h1>Passing data from main JSP's page to other JSP's page </h1>
16
17      </body>
18  </html>
```

14. Add additional HTML's tag and Java Scriplet to retrieve the value from main's form.

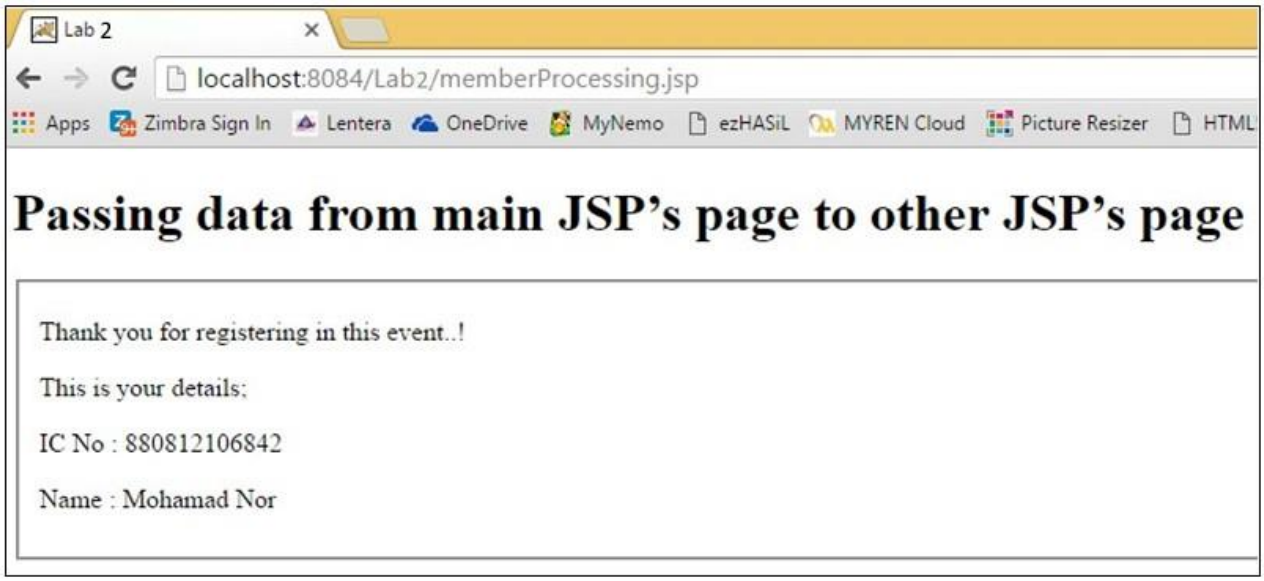
```
16      <fieldset>
17          <%
18              //Define variables...
19              String myIC  = null;
20              String myName  = null;
21
22              //Use request.getParameter() method to retrieve data from main's form...
23              myIC = request.getParameter("my_icno");
24              myName = request.getParameter("my_name");
25          %>
26
27          <!-- Display the output... -->
28          <p>Thank you for registering in this event..!</p>
29          <p>This is your details;</p>
30          <p>IC No : <%=myIC%></p>
31          <p>Name : <%=myName%></p>
32      </fieldset>
```

15. Compile *memberProcessing.jsp* file.

16. Run the *memberRegister.jsp* file and fill-up the input.

17. Click *Submit* button to send the request.

18. These inputs will be sent to *memberProcessing.jsp* page and produce the following page.



Reflection

1. How do you want to submit specific information from one form to next form?

- Using this code <form id="memberFrm" action="memberProcessing.jsp" method="post" onsubmit="return checkICNo()"> . So, the specific information (IC number and name) submitted from the first form is passed to the next form (second JSP page) through the HTTP POST request and then dynamically displayed on the second page.

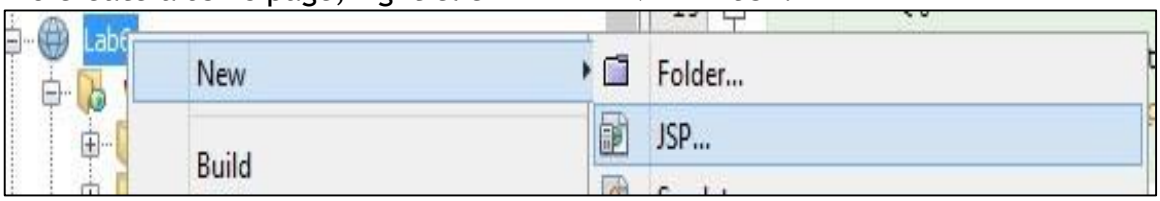
2. What happened if the field name you specify in *request.getParameter("field_name")* in second page is different from the field name you defined in first page?

- If the field name specified in request.getParameter("field_name") in the second page is different from the field name defined in the first page, the getParameter() method will not be able to retrieve the value associated with the specified field name. As a result, the value retrieved will be null.

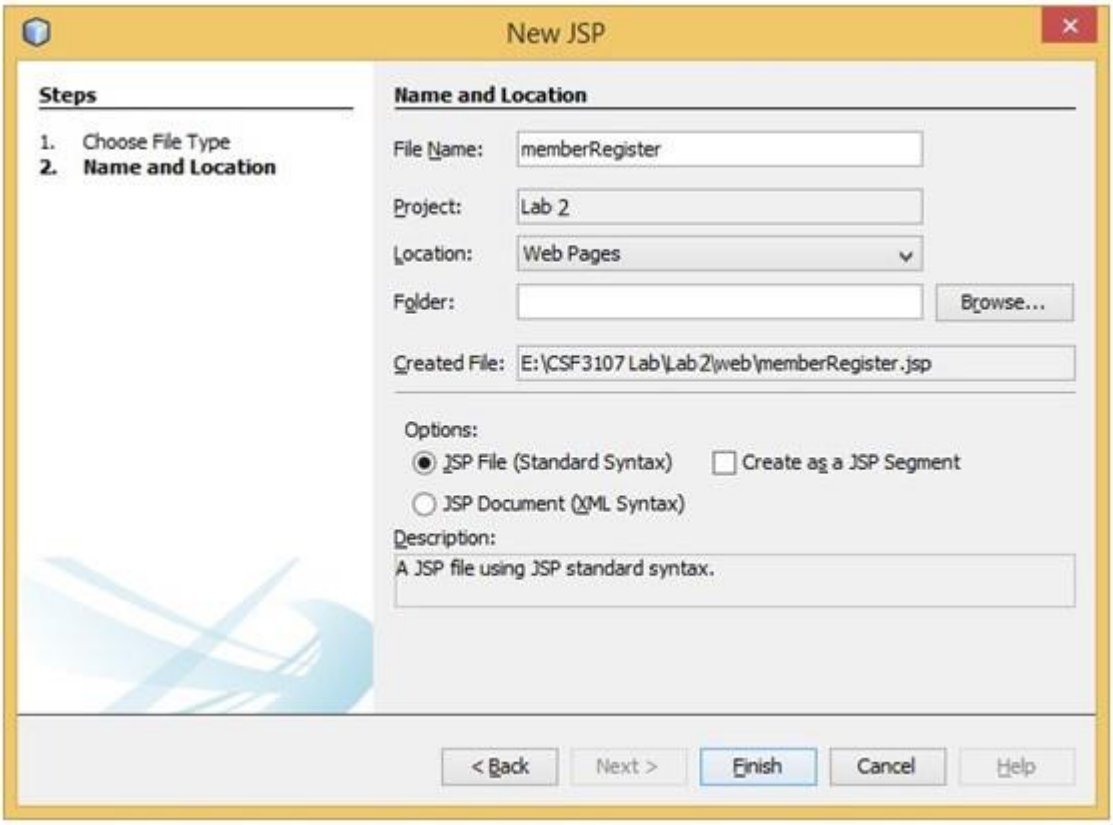
Task 2: Using Mathematics Operations in JSP

Objective	:	To demonstrate the use of *request.getParameter (“Mathematics operations”)* in JSP’s page.
Problem Description	:	i. Create a page *Calculator.jsp* consists of interface represent basic calculator. ii. When user key-in inputs, process the request and display the results direct in JSP page.
Estimated time	:	30 minutes

1. Go to Project *Lab2*.
2. To create a JSP’s page, right click *Lab2 -> New -> JSP*.



3. Key-in File Name: *Calculator*.



4. Click *Finish* button.

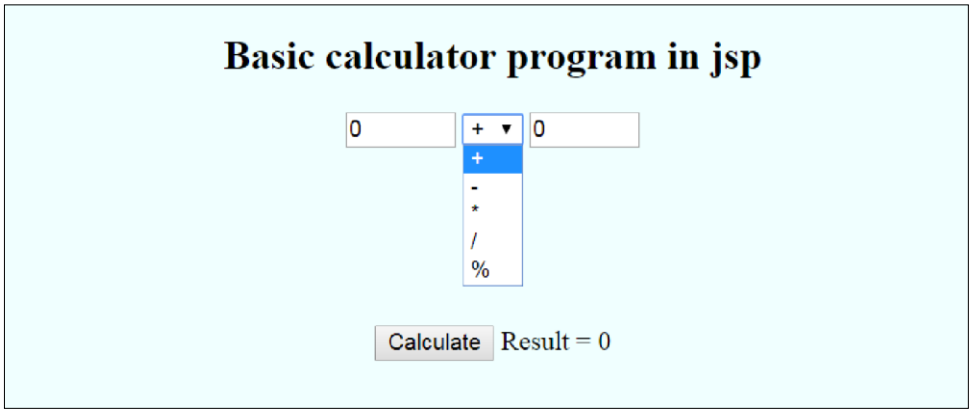
5. Source code for *Calculator.jsp* will appear.

6. Write a HTML's markup to produce HTML's form

```
<body bgcolor= "#a00FFF" text= "gold">



<center>

<h2>Basic calculator program in jsp</h2>
<form method ="get" name ="f1">
<input type ="text" size ="20" name ="operand1" value = "" />

<select name = op size = 1>
<option value = "0" >+</option>
<option value = "1" >-</option>
<option value = "2" >*</option>
<option value = "3" >/</option>
<option value = "4" >%</option>
</select>

<input type ="text" size="20" name ="operand2" value = ""/>
<p><br><br><br><br>
<input type = submit value = Calculate />


</form>

</body>
```

7. Save and compile *Calculator.jsp* file.

8. Run the *Calculator.jsp* file and you should get the interface as below:



9. Add additional HTML's tag and Java scriplet to retrieve the value from users.

```
<%
String num1 = "0",num2 = "0";
int result = 0;
String op = "+";

char opchar = op.charAt(0);
if (request.getParameter("op") != null){
op = request.getParameter("op");
opchar = op.charAt(0);

num1 = request.getParameter("operand1");
num2 = request.getParameter("operand2");

switch(opchar){
case '0': result = Integer.parseInt(num1) + Integer.parseInt(num2);
break;
case '1': result = Integer.parseInt(num1) - Integer.parseInt(num2);
break;
case '2': result = Integer.parseInt(num1) * Integer.parseInt(num2);
break;
case '3': result = Integer.parseInt(num1) / Integer.parseInt(num2);
break;
case '4': result = Integer.parseInt(num1) % Integer.parseInt(num2);
break;
}
}
%>

Result = <%= result + "" %>
```

9. Further, add additional Java Scriplet to HTML's tag as below.

```
<body bgcolor= "#a00FFF" text= "gold">
<center>

<h2>Basic calculator program in jsp</h2>
<form method ="get" name ="f1">
<input type ="text" size ="20" name ="operand1" value = <%= num1 %> />

<select name = op size = 1>
<option value = "0" >+</option>
<option value = "1" >-</option>
<option value = "2" >*</option>
<option value = "3" >/</option>
<option value = "4" >%</option>
</select>

<input type ="text" size="20" name ="operand2" value = <%= num2 %> />
<p>
<input type = submit value = Calculate />

Result = <%= result + "" %>
</form>

</body>
```

10. Compile *Calculator.jsp* file.

11. Run the *Calculator.jsp* file and test the clculator.

Reflection

1. How do you want to submit specific information from one form to next form?

- Add hidden input fields to store the values of operand1, operand2, op, and result.
- Modify the action of the form to point to the next JSP page where you want to submit the data.
- Retrieve the values of the hidden input fields on the next JSP page.

2. What happened if the field name you specify in *request.getParameter("field_name")* in second page is different from the field name you defined in first page?

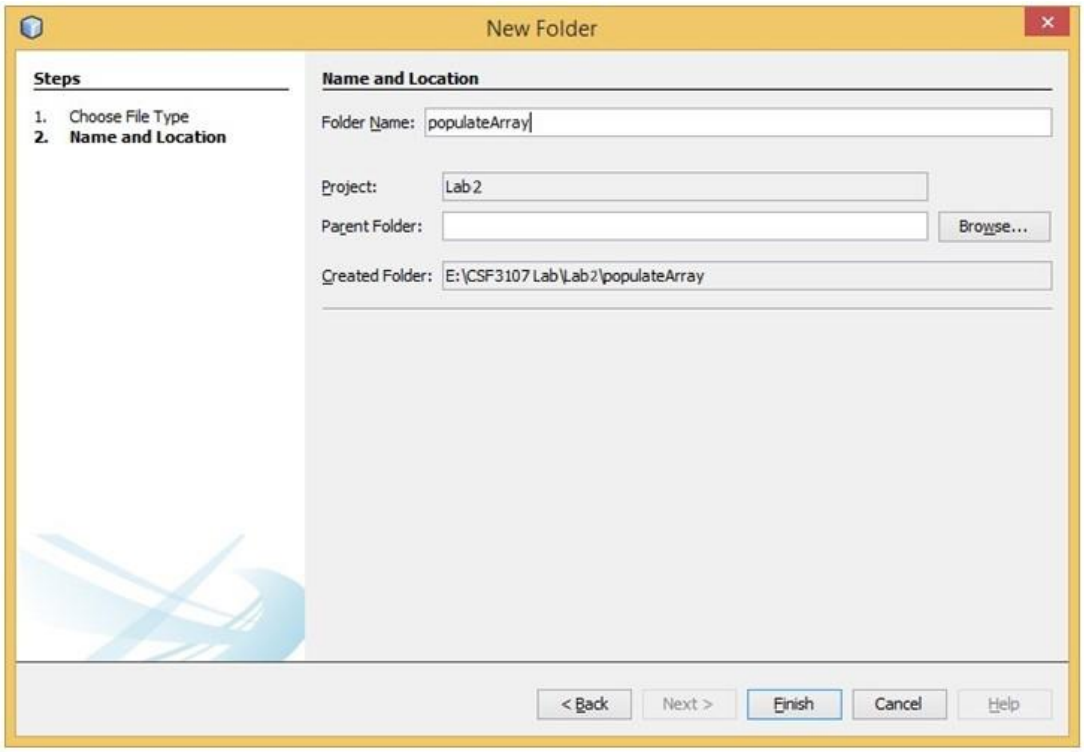
- If the field name you specify in request.getParameter("field_name") in the second page is different from the field name you defined in the first page, the getParameter() method will return null for that field on the second page.

Task 3: Populate an Array Values into HTML's Table

Objective	:	Read Java array and populate it into HTML's table.
Problem Description	:	i. Create 2D array that store sales data. ii. Then, read an array and populate into HTML's table.
Estimated time	:	50 minutes

1. Go to Project *Lab2*.
2. To create a JSP's page, right click *Lab2* -> *New* -> *JSP*.

3. Key-in File Name: *populateArray*.



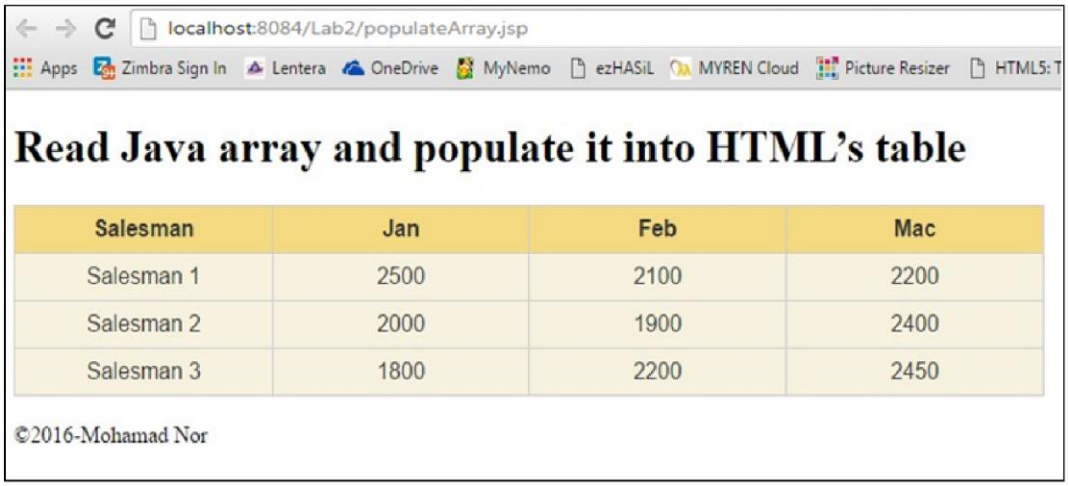
4. Click *Finish* button.

5. Prepare standard HTML's a markup for page *populateArray.jsp*.

6. Write a Java Scriplet and store the following information into an array;

	Jan	Feb	Mac
Salesman 1	2500	2100	2200
Salesman 2	2000	1900	2400
Salesman 3	1800	2200	2450

7. Read the array and populate its value into HTML's table.
8. Save and compile *populateArray.jsp* file.
9. Run the *populateArray.jsp* file and sample of output is shown below:



Reflection

1. Write a sample syntax to declare 2D Java array.

- dataType[][] arrayName = new dataType[m][n];

2. Define a sequence of steps on how you accomplish Task 7.

-
```
<%
List<String[]> salesmans = new ArrayList<>();
salesmans.add(new String[]{"Salesman 1", "2500", "2100", "2200"});
salesmans.add(new String[]{"Salesman 2", "2000", "1900", "2400"});
salesmans.add(new String[]{"Salesman 3", "1800", "2200", "2450"});
%>

<% for (String[] salesman : salesmans) { %>

<tr>
        <td><%= salesman[0] %></td>
        <td><%= salesman[1] %></td>
        <td><%= salesman[2] %></td>
        <td><%= salesman[3] %></td>
    </tr>
  <% } %>
```

3. What is the difference between HTML's page and JSP's page

JSP	HTML
- JSP generated dynamic web pages only. - JSP runs straight on the Web Server and local JVM. - JSP is termed as server-side scripting language.	- Html generated static web pages only. - HTML runs in the Web Browser. - HTML is termed as client-side scripting language.

Task 4: Perform Calculation of Car Loan

Objective : Passing input to next page for further processing.

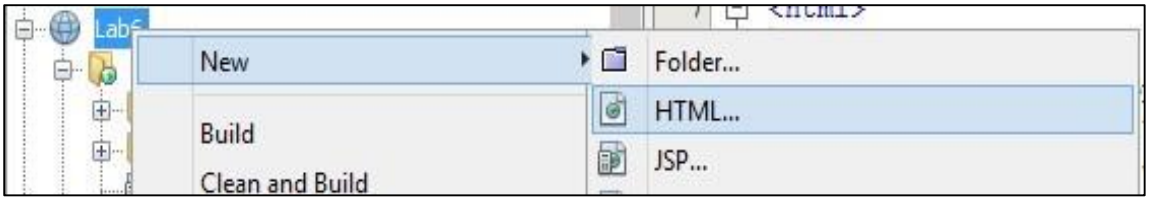
Problem Description : i. Create simple interface in HTML that consists of Loan Amount and Loan Period. (Loan Period < 5 years, interest is 2.8% per year, and > 5 years interest is 4.5% per year).

ii. Submit the form and perform calculation based on user input and, finally, display the result.

Estimated time : 50 minutes

1. Go to Project Lab2.

2. To create a HTML's page, right click Lab2 -> New -> JSP



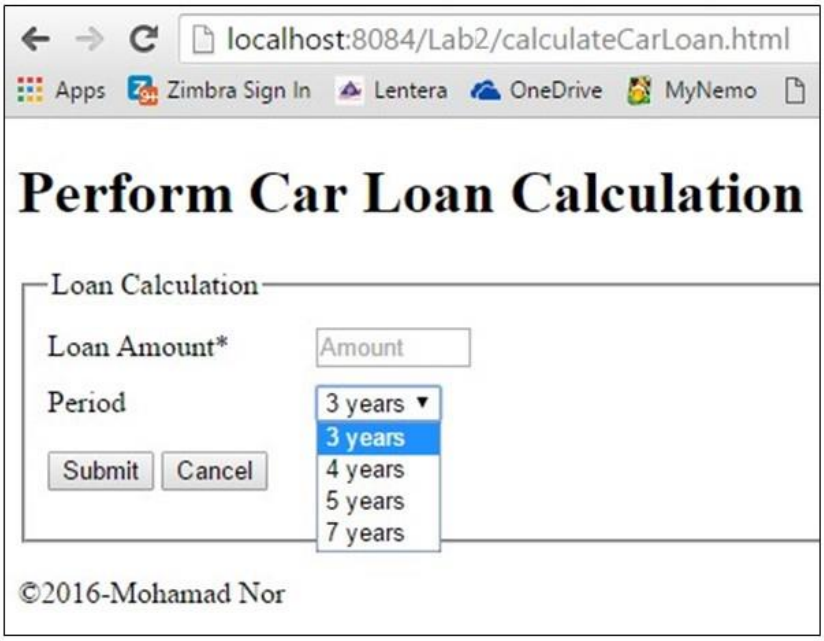
3. Key-in File Name: *calculateCarLoan*

4. Create a standard HTML's markup for form.

5. In your form, create two (2) fields; *Loan Amount* and *Loan Period*.

6. Save *calculateCarLoan.html* and run the file.

7. You will get the following output.



8. Create JSP's file and rename the file as *processCalculateCarLoan*.

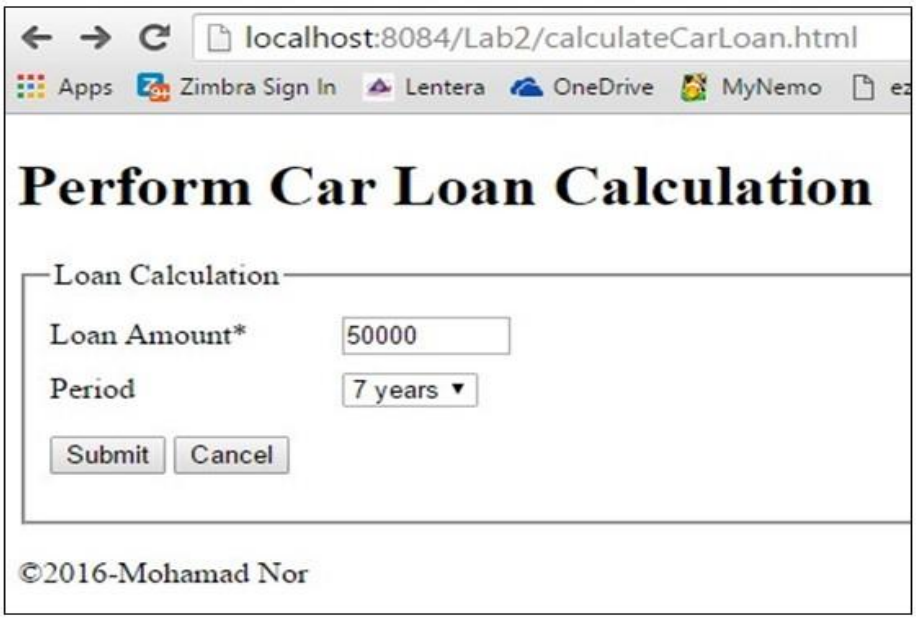
9. Construct the logic for calculating car loan and display the result.

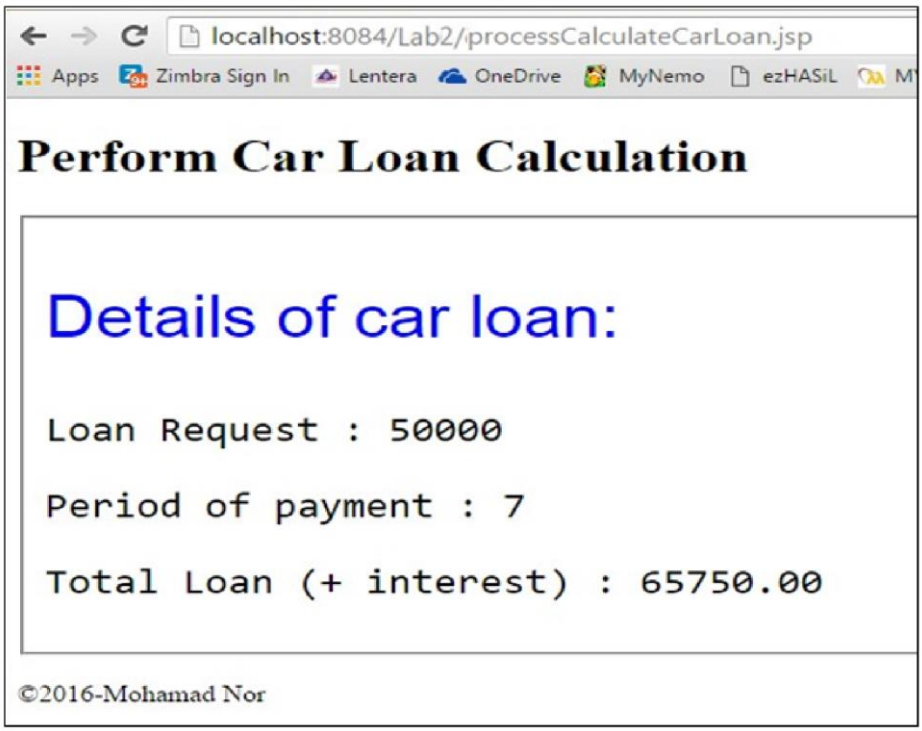
10. Save and compile *processCalculateCarLoan.jsp*.

11. Run *calculateCarLoan.html* file and fill-up the input.

12. Then, submit your result.

13. You should get the following output;





Reflection

1. How you want to retrieve data from previous page?

- To retrieve data from the previous page in a JSP file, you typically use the request.getParameter() method to access the values submitted through the form. In the provided JSP code, this is done for retrieving the loan amount and loan period.

2. Where the construction of logic occur for calculating Total Loan (+ interest) ?

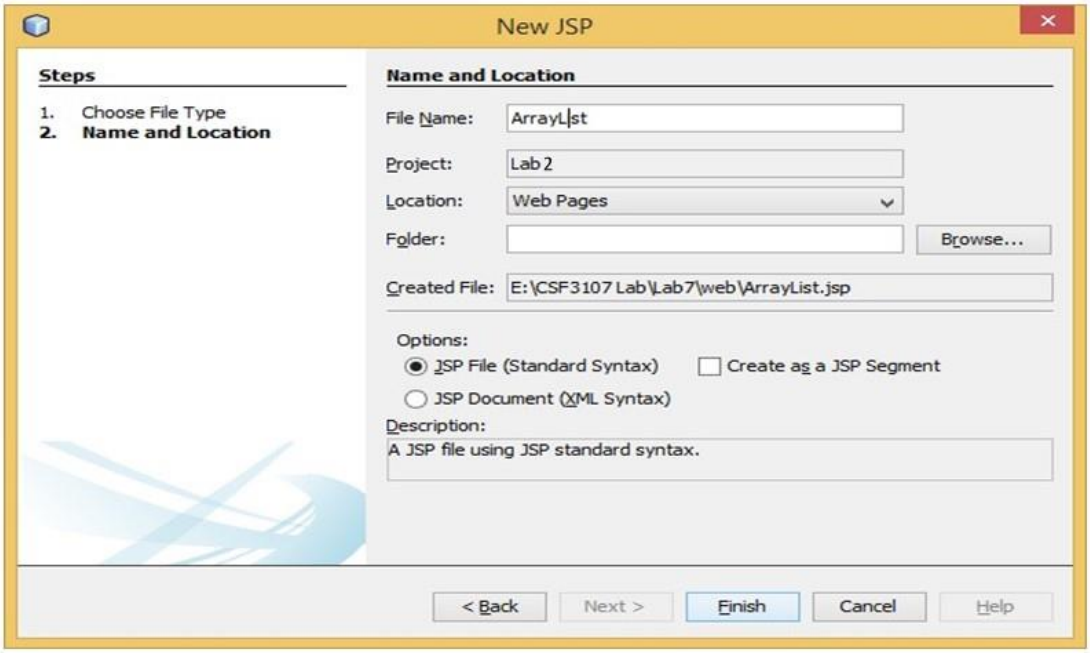
- double totalLoan = monthlyPayment * totalMonths;

Task 5: Using JSP Page Directive to Call Java API

Objective	:	Use JSP page directive elements to call certain Java API.
Problem Description	:	Using Java *ArrayList* object to store data and retrive it via JSP page.
Estimated time	:	20 minutes

1. Create a new JSP's file.

2. Type file name as *ArrayList*.



2. Click *Finish* button.

4. Type title as *Use Java ArrayList*.

5. Type header1 as *Use JSP Page Directive*

```
<%--
    Document   : ArrayList
    Created on : 10-Apr-2016, 09:24:46
    Author     : Mohamad Nor Hassan
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Use Java ArrayList</title>
    </head>
    <body>
        <h1>Use JSP Page Directive</h1>
    </body>
     <br/>
    <footer>&copy2016-Mohamad Nor</footer>
</html>
```

6. In order to use Java *ArrayList*'s object, we need to use JSP Page Directive and import the related API.

```
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <%@page import="java.util.ArrayList"%>
```

7. In order to use Java syntax, create a Java Scriptlet notation.

```
15      <body>
16          <h1>Use JSP Page Directive</h1>
17          <%

19          %>
20      </body>
```

8. Create an object *ArrayList* to store a list of student name.

```
17          <%
18              //Create ArrayList object ...
                ArrayList<String> studentList = new ArrayList<String>();
20          %>
```

9. Add the following name to ArrayList's object.

- Mohamad

  Azam ✓ Peter

  Chong

- Rahimah
  Mansor

- Sri Devi ✓ Ng

  Hue Ween

✓ S. Nagarajan

```
21            //Store student name..
22            studentList.add(0, "Mohamad Azam");
23            studentList.add(1, "Peter Chong");
24            studentList.add(2, "Rahimah Mansor");
25            studentList.add(3, "Sri Devi");
26            studentList.add(4, "Ng Hue Ween");
27            studentList.add(5, "S. Nagarajan");
```

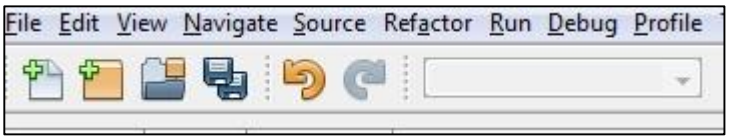
10. Display the number of records for an ArrayList's object.

```
29            //Display the number of records..
30            out.println("<p>The number of records in ArrayList are " +
31                        studentList.size() + "</p>");
```

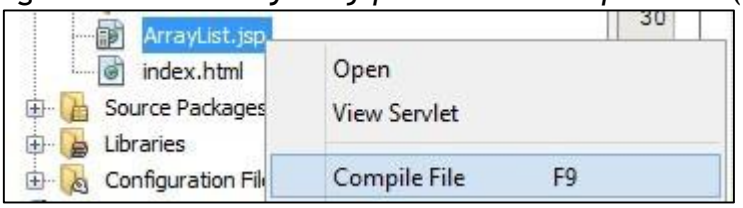
11. Finally, populate the list of students.

```
39            //Populate a list of students...
40            for (int i=0; i < studentList.size(); i++ )
41               out.println("<p>Record " + (i+1) + " is " + studentList.get(i) + "</p>");
```

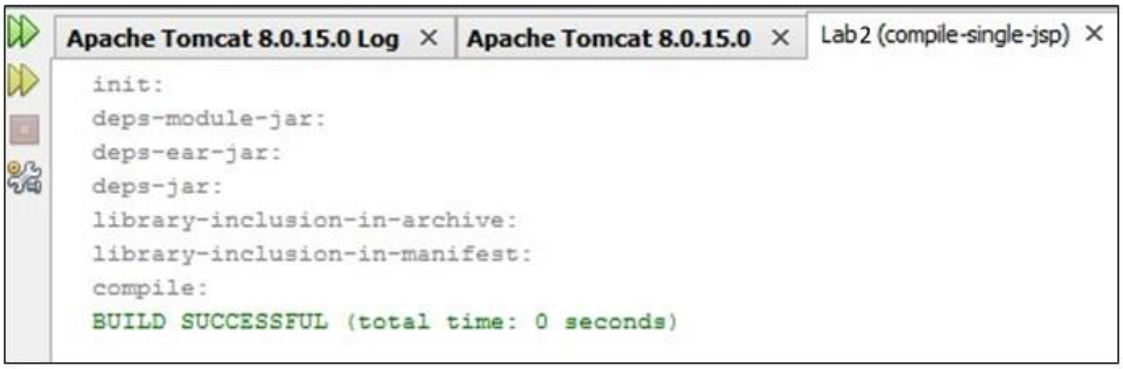
12. Click *SaveAll* icon.



13. Right click file *ArrayList.jsp* and click *Compile File* (F9).

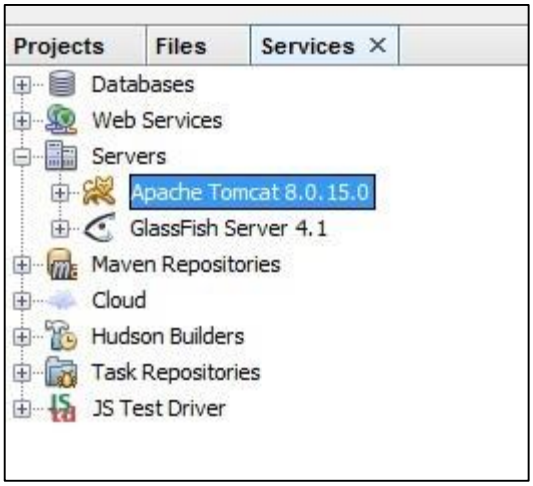


14. You will get notification message the the bottom of Netbeans IDE with the green colour.

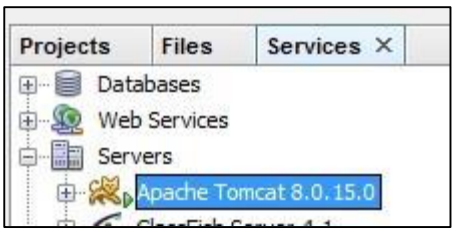


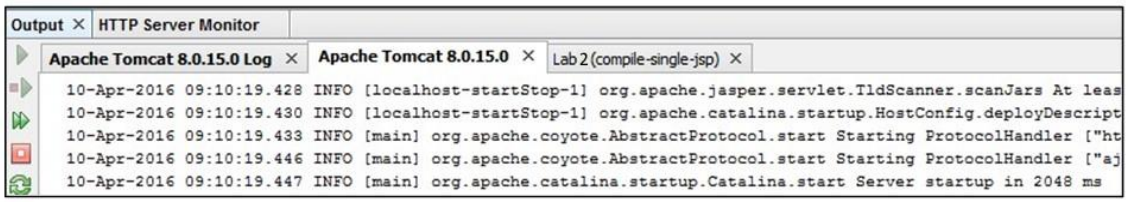
15. Before running any JSP's files for first time upon opening your Netbeans IDE, you need to start your web server (i.e; Apache Tomcat).

16. To perform this, go to *Services -> Servers -> Apache Tomcat*.

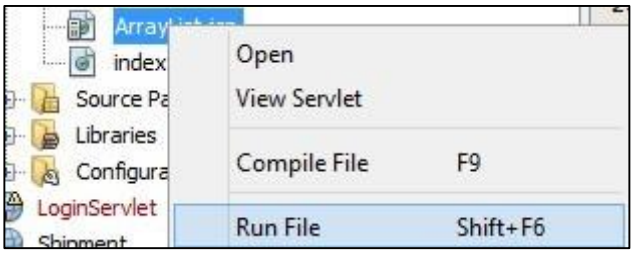


17. You should get the green indicator at *Apache Tomcat*'s icon and *Apache Tomcat* output message with the time taken to start specified time to start *Apache Tomcat* web server.





18. Go *to Project*'s tab. Then right click file *ArrayList.jsp* and click *Run File* (Shift+F6).



19. Output will appear in web browser.

Reflection

1. What you have learnt from this exercise?

- This exercise showcases the utilization of ArrayList in Java, allowing dynamic storage and manipulation of data. Through JSP scriptlets, it demonstrates the integration of Java code within HTML for dynamic web content generation, illustrating basic principles of web development.

2. Write a sample syntax how you want to use java *Math* object in JSP?

```
<%
   double number = 5.5;
   double squareRoot = Math.sqrt(number);
%>

<p>The square root of <%= number %> is <%= squareRoot %>.</p>
```

3. List and write a sample syntax for THREE (3) of JSP page directive.

- page: The page directive is used to define various attributes of the JSP page, such as error handling, content type, language, and import statements. <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
- include: The include directive is used to include the content of another file (JSP or HTML) within the current JSP page during translation time. <%@ include file="header.jsp" %>
- taglib: The taglib directive is used to declare and define custom tag libraries for use in the JSP page. <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

Task 6: Use JSP Include directive for JSP Page

Objective	:	Demonstrate the use of JSP Page Include directive.
Problem Description	:	Create a JSP master page that displays the header, main contents and footer.
Estimated time	:	30 minutes

1. Create a new JSP's file.
2. Type file name as *mainPage*.
3. Create content for *mainPage.jsp* as below.

Using JSP Include directive

Java Server Page (JSP) is a technology for controlling the content or appearance of Web pages through the use of servlets, small programs that are specified in the Web page and run on the Web server to modify the Web page before it is sent to the user who requested it.

4. Create a header file as *headerPage.jsp* and display the following output.

ABC Sdn Bhd

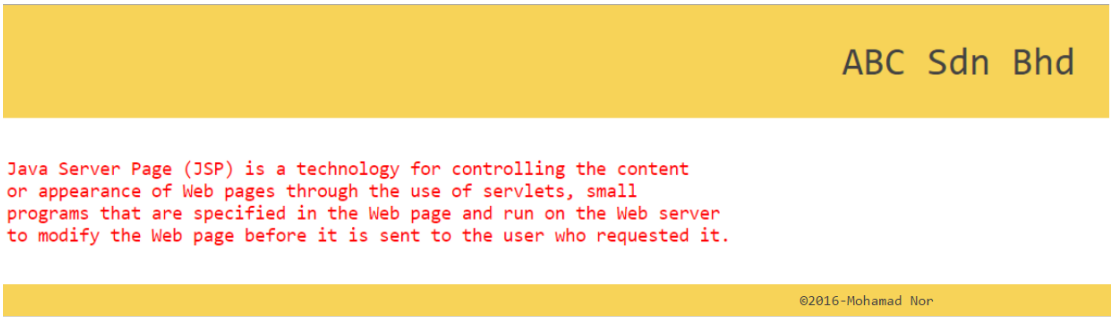
5. Create a header file as *footerPage.jsp* and display the following output

©2016-Mohamad Nor

6. Include your *headerPage.jsp* and *footerPage.jsp* inside your *mainPage.jsp*.
7. Save *mainPage.jsp*

8. Compile and run *mainPage.jsp*.

9. You should het the following output.



Reflection

1. What you have learnt from this exercise?

- From this exercise, I've reinforced the understanding of Java Server Pages (JSP) and its usage in web development. I've learned about the importance of modularizing web pages by using includes, which helps maintain consistency and reusability across the site. Additionally, I've gained practical experience in structuring JSP files to create common elements like headers and footers, enhancing the organization and readability of web projects.

2. Write a syntax how you want to include *common.html* file that located at a directory known as *master*.

- <%@ include file="/master/common.html" %>

Exercise

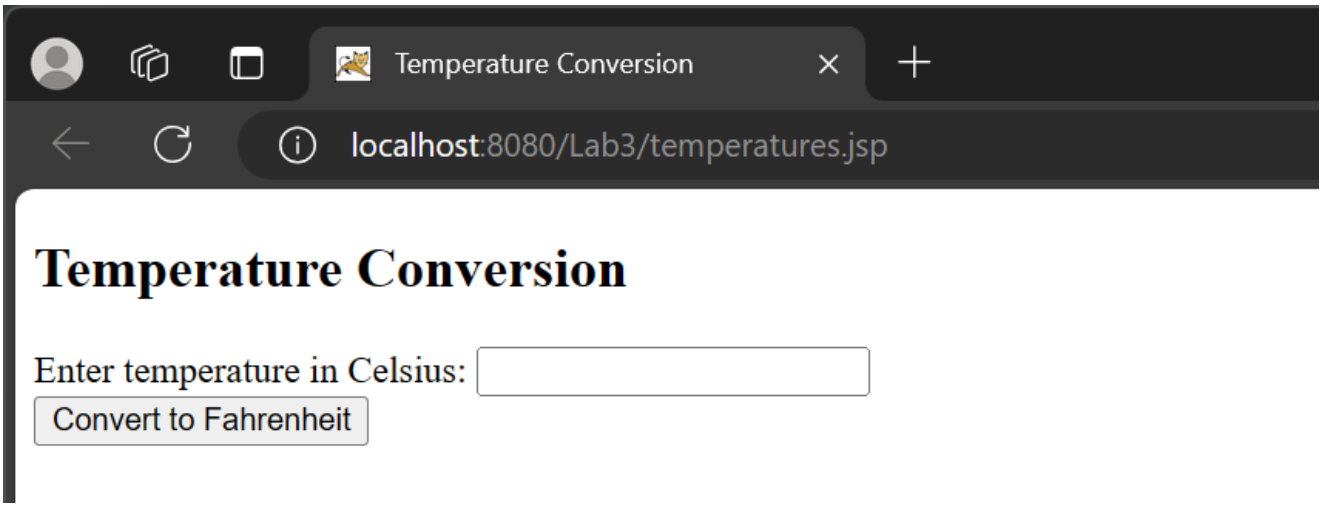
1. Write a JSP's page to convert temperatures to Fahrenheit temperatures and via versa. The formula is given as:

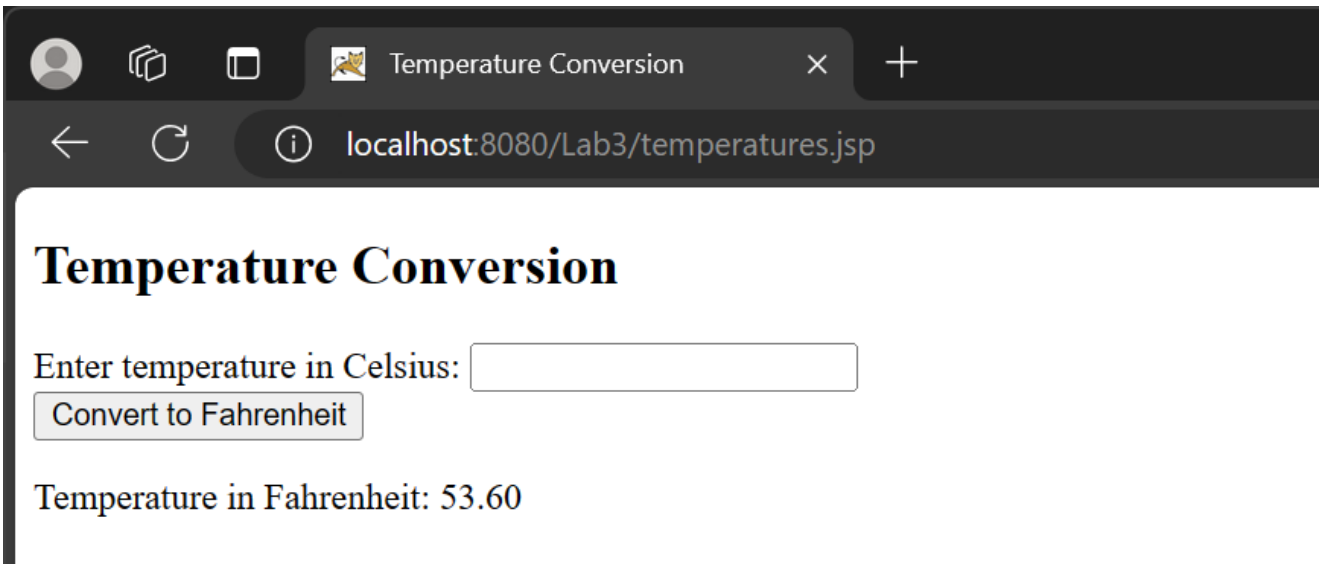
   $$F = (9/5)C + 32$$

   Your program should ask the user to enter a temperature in Celsius, and then display the temperature converted to Fahrenheit.

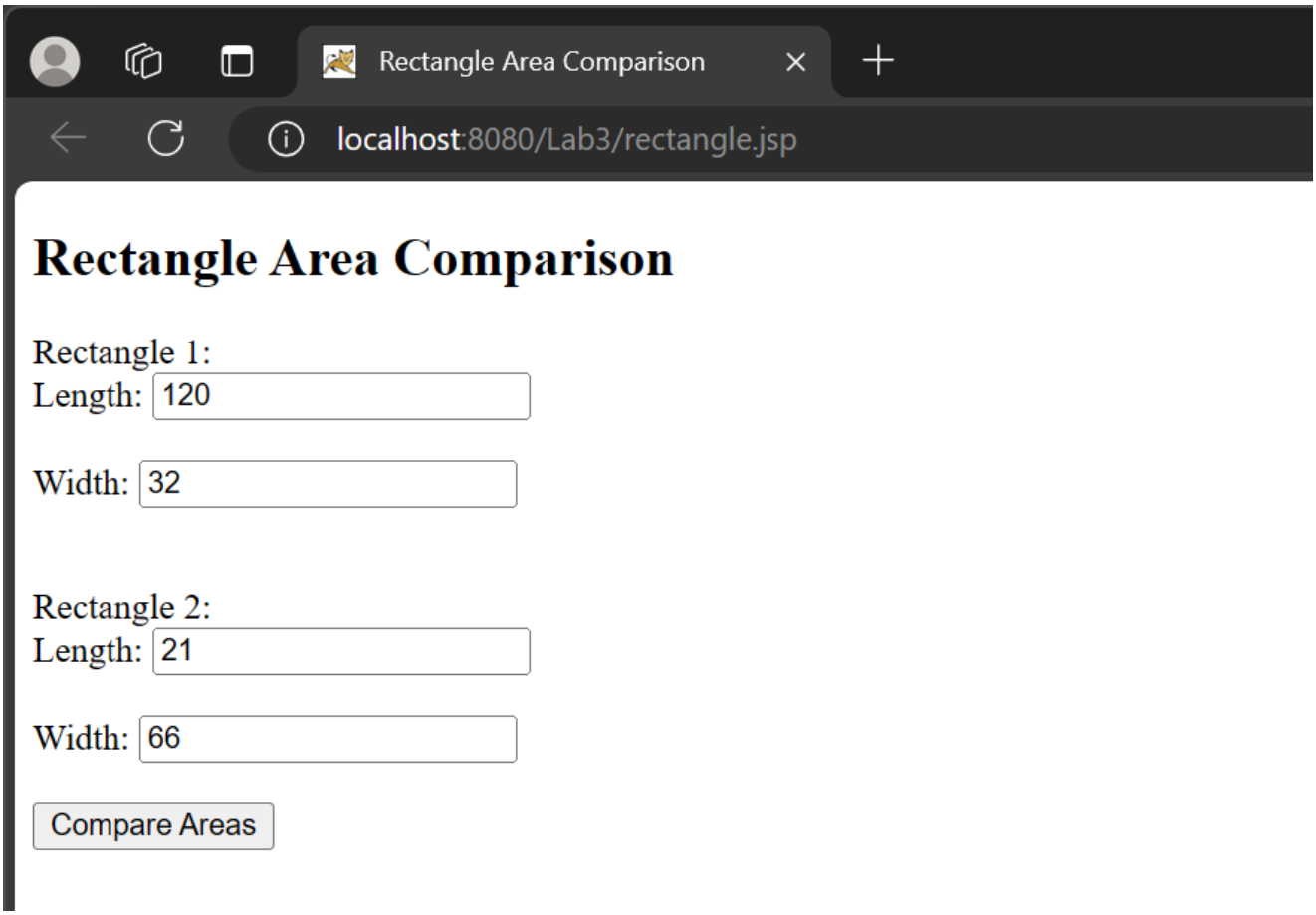
2. Write a JSP's form that asks for the length and width of two rectangles. The program should tell the user which rectangle has the greater area, or if the areas are the same. [Note: All result must be in 2 decimal places].

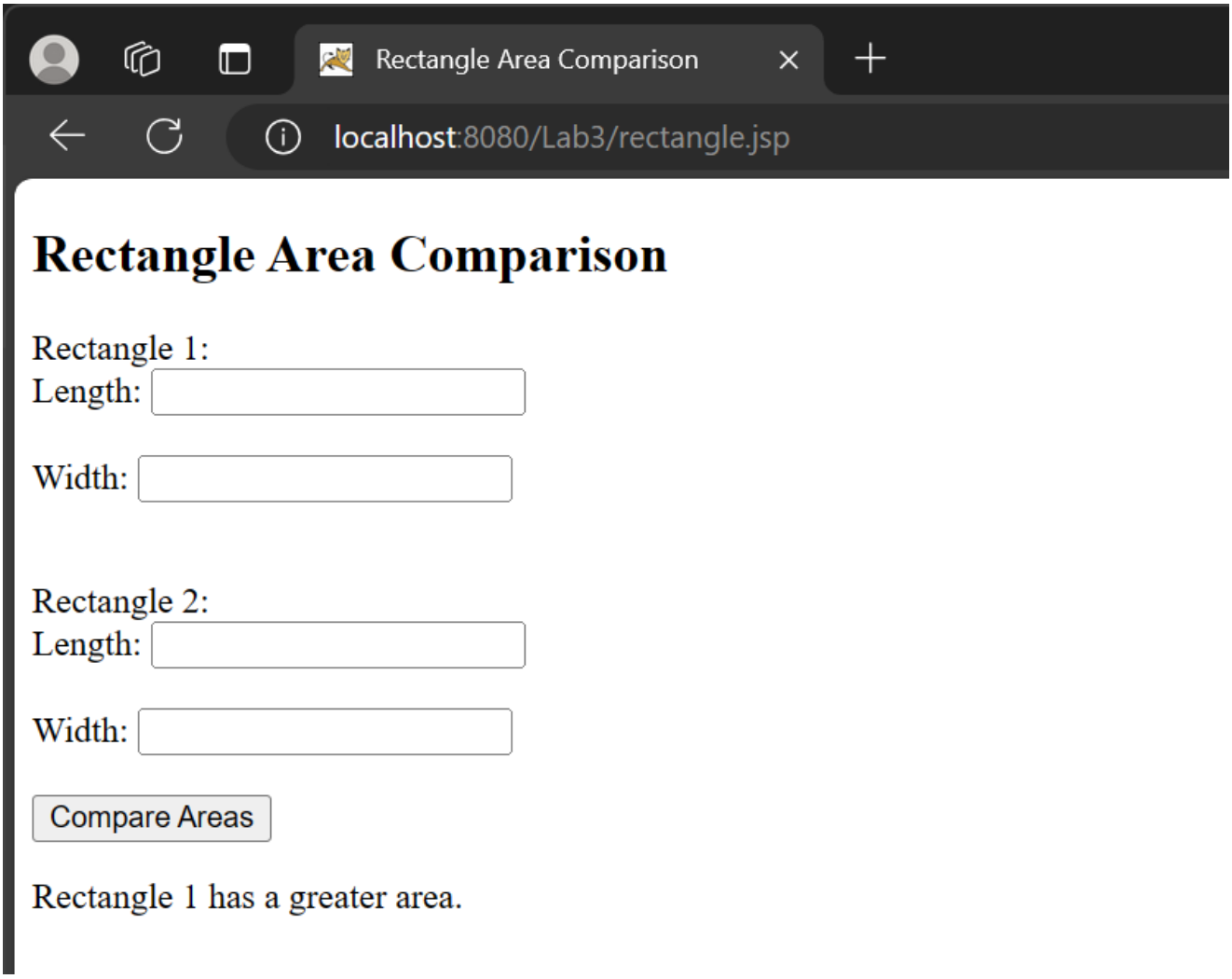
Output temperatures





Output rectangle




Rectangle Area Comparison
localhost:8080/Lab3/rectangle.jsp
Rectangle Area Comparison
Rectangle 1:
Length:
Width:
Rectangle 2:
Length:
Width:
Compare Areas
Rectangle 1 has a greater area.