

Student ID \_\_\_\_\_ Student Name \_\_\_\_\_

## Distributed Computing DE Midterm Exam

### PRIVATE AND CONFIDENTIAL

1. Allotted exam duration is 2 hours.
2. Closed book/notes or open book/notes.
3. No personal items including electronic devices (cell phones, computers, calculators, PDAs).
4. Cell phones must be turned in to your proctor before beginning exam.
5. No additional papers are allowed. Sufficient blank paper is included in the exam packet.
6. Exams are copyrighted and may not be copied or transferred.
7. Restroom and other personal breaks are not permitted.
8. Total exam including questions and scratch paper must be returned to the proctor.

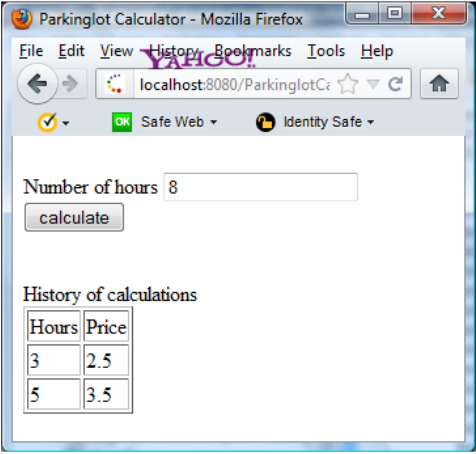
3 blank pages are provided for writing the solutions and/or scratch paper. All 3 pages must be handed in with the exam

**BE VERY CAREFUL WITH THE GIVEN 2 HOURS AND USE YOUR TIME WISELY. THE ALLOTTED TIME IS GIVEN FOR EVERY QUESTION.**

Write your name and student id at the top of this page.

#### Question 1: Servlet [30 points][35 minutes]

Write a simple parkinglot calculation application that works as follows:



Parkinglot Calculator - Mozilla Firefox

File Edit View History Bookmarks Tools Help

localhost:8080/ParkinglotCa

Safe Web Identity Safe

Number of hours 8

calculate

History of calculations

Hours	Price
3	2.5
5	3.5

In the inputfield for “Number of hours” enter the number of hours you have parked, and the application shows you how much you have to pay. The application also shows the history of all calculations that you have entered. The numbers of hours you can enter should only integers, but you don’t need to verify this in your code.

This application **MUST** be implemented with a servlet. Do not use a JSP or JSF page. Complete the partial given code. Make sure you add all the code that is necessary for the correct working of this application. Do not write constructors or getter and setter methods.

```
public class ParkinglotCalculator {  
    public double computePrice (int hours){  
        if (hours == 1) return 1.0;  
        else if (hours < 3 ) return 1.75;  
        else if (hours < 5 ) return 2.5;  
        else if (hours < 8 ) return 3.5;  
        else return 4.0;  
    }  
}
```

```
public class ParkingQuery {  
    private int hours;  
    private double price;  
  
    public ParkingQuery(int hours, double price) {  
        super();  
        this.hours = hours;  
        this.price = price;  
    }  
  
    public int getHours() {  
        return hours;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
}
```

```

public class ParkingCalculatorServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        double price = 0.0;
        int ihours = 0;

        String shours = request.getParameter("hours");
        if (shours != null){
            ihours = Integer.parseInt(shours);
            ParkinglotCalculator calculator = new
                ParkinglotCalculator();
            price = calculator.computePrice(ihours);
        }

        HttpSession session = request.getSession();
        List<ParkingQuery> historylist = (List<ParkingQuery>)
            session.getAttribute("history");
        if (historylist == null){
            historylist = new ArrayList<ParkingQuery>();
            session.setAttribute("history", historylist);
        }
        else{
            historylist.add(new ParkingQuery(ihours, price));
        }

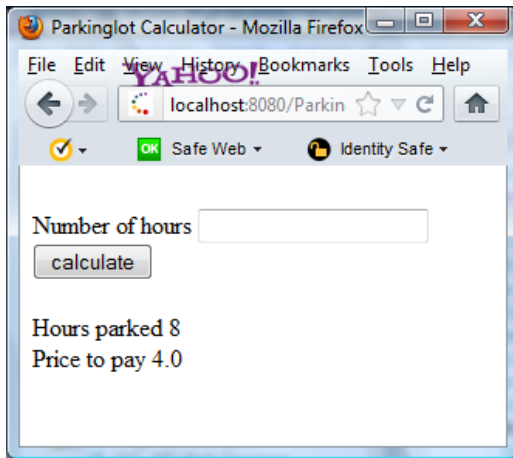
        out.println("");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Parkinglot Calculator</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form name='hours_form' method='get'
            action='ParkingCalculatorServlet'>");
        out.println("<br>");
        out.println("Number of hours  <input type='TEXT' name='hours'>");
        out.println("<br />");
        out.println("<input type='submit' value='calculate' >");
        out.println("</form>");
        out.println("<br />");
        out.println("History of calculations<br />");
        out.println("<table border='1'>");
        out.println("<tr>");
        out.println("<td>Hours</td><td>Price</td>");
        out.println("</tr>");
        for (ParkingQuery query :historylist){
            out.println("<tr>");
            out.println("<td>"+query.getHours()+"</td><td>"+
                query.getPrice()+"</td>");
            out.println("</tr>");
        }
        out.println("</table>");
    }
}

```

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

## Question 2: MVC [25 points][35 minutes]

Now we need to write a simplified version of the parkinglot calculator application, but now we are going to implement it using the MVC pattern.



In the inputfield enter the number of hours you have parked, and the application shows you how much you have to pay. The numbers of hours you can enter should only integers, but you don't need to verify this in your code. This application does NOT show the history of all calculations that you have entered.

This application **MUST** be implemented according the **Model-View-Controller** style using servlets, JSP's and Java classes. Do not use JSF.

Complete the partial given code. Make sure you add all the code that is necessary for the correct working of this application.

```
public class ParkinglotCalculator {  
    public double computePrice (int hours){  
        if (hours == 1) return 1.0;  
        else if (hours < 3 ) return 1.75;  
        else if (hours < 5 ) return 2.5;  
        else if (hours < 8 ) return 3.5;  
        else return 4.0;  
    }  
}
```

```

public class ParkingCalculatorServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String shours = request.getParameter("hours");
        if (shours != null){
            int ihours = Integer.parseInt(shours);
            ParkinglotCalculator calculator = new
                ParkinglotCalculator();
            double price = calculator.computePrice(ihours);
            request.setAttribute("price", price);
            request.setAttribute("hours", ihours);
        }

        RequestDispatcher view
            =request.getRequestDispatcher("parkinglotcalculator.jsp");
        view.forward(request, response);
    }
}

```

#### Parkinglotcalculator.jsp

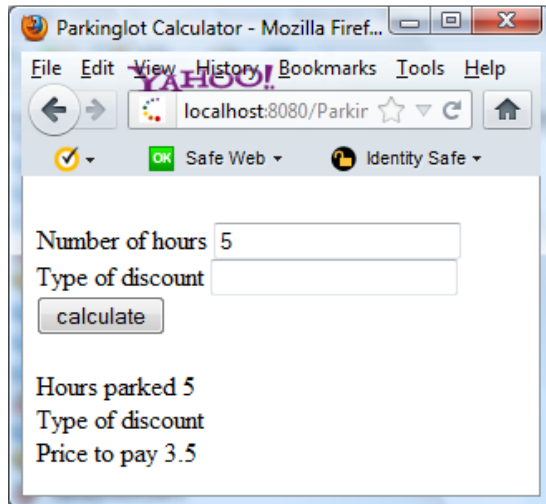
```

<html>
<head>
    <title>Parkinglot Calculator</title>
</head>
<body>
    <form name="hours_form" method="post"
        action="ParkingCalculatorServlet">
        <br>
        Number of hours <input type="TEXT" name="hours">
        <br />
        <input type="submit" value="calculate" >
    </form>
    <br />
    Hours parked ${hours }
    <br />
    Price to pay ${price }
</body>
</html>

```

### Question 3: JSF [30 points][30 minutes]

Write a new parking calculator with JSF that works as follows



Parkinglot Calculator - Mozilla Firef...

File Edit View History Bookmarks Tools Help

localhost:8080/Parkir

Number of hours 5

Type of discount

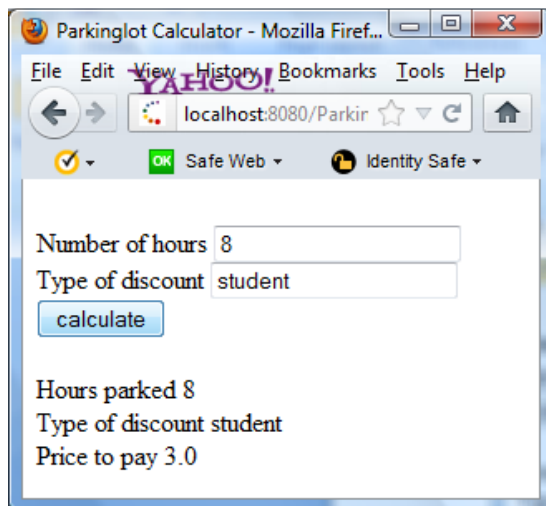
calculate

Hours parked 5

Type of discount

Price to pay 3.5

In the inputfield for “Number of hours” enter the number of hours you have parked, and the application shows you how much you have to pay. The numbers of hours you can enter should only integers, but you don’t need to verify this in your code.



Parkinglot Calculator - Mozilla Firef...

File Edit View History Bookmarks Tools Help

localhost:8080/Parkir

Number of hours 8

Type of discount student

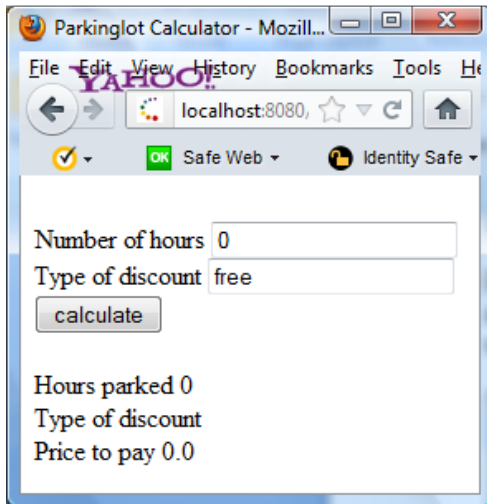
calculate

Hours parked 8

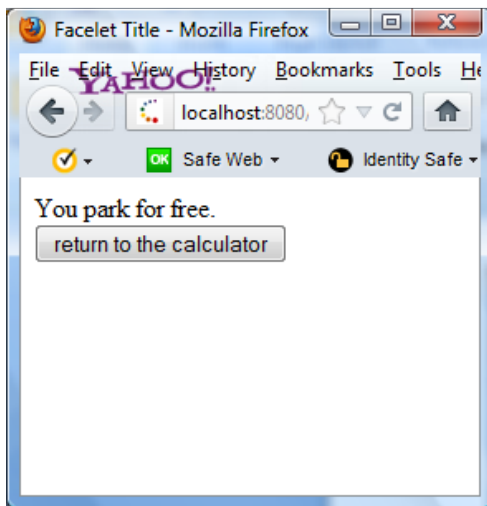
Type of discount student

Price to pay 3.0

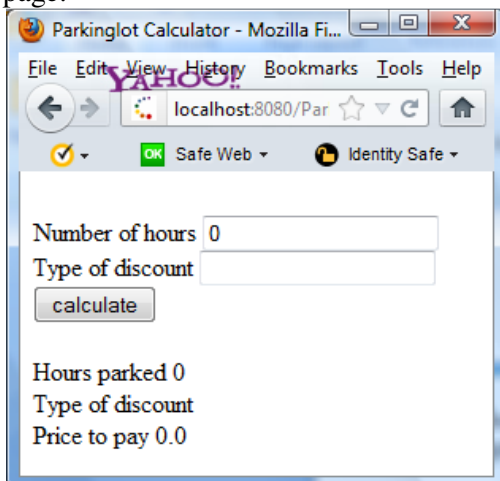
In the inputfield “Type of discount” you can enter either “student” or ”elderly”. Students get 20% discount and erderly get 25% discount.



If you enter 0 for number of hours and 'free' for type of discount and you click the calculate button, then the application shows a new page:



When you click the 'return to the calculator' button, then you navigate back to the calculator page.





Complete the partial given code. Make sure you add all the code that is necessary for the correct working of this application. Do not write getter and setter methods.

```
public class ParkinglotCalculator {  
    public double computePrice(int hours, String discount) {  
        double price;  
        if (hours == 1) {  
            price = 1.0;  
        } else if (hours < 3) {  
            price = 1.75;  
        } else if (hours < 5) {  
            price = 2.5;  
        } else if (hours < 8) {  
            price = 3.5;  
        } else {  
            price = 4.0;  
        }  
  
        if (discount.equals("erderly")) {  
            price = price * 0.8;  
        } else if (discount.equals("student")) {  
            price = price * 0.75;  
        }  
        return price;  
    }  
}
```

parkinglotcalculator.xhtml:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Parkinglot Calculator</title>
  </h:head>
  <h:body>
    <h:form>
      <br/>
      Number of hours <h:inputText value="#{parkingBean.hours}"/>
      <br />
      Type of discount <h:inputText
        value="#{parkingBean.discount}"/>
      <br />
      <h:commandButton value="calculate"
        action="#{parkingBean.calculate}" />

    </h:form>
    <br />
    Hours parked <h:outputText value="#{parkingBean.hours}"/>
    <br />
    Type of discount <h:outputText value="#{parkingBean.discount}"/>
    <br />
    Price to pay <h:outputText value="#{parkingBean.price}"/>
  </h:body>
</html>
```

```

@ManagedBean
@RequestScoped
public class ParkingBean {

    private int hours;
    private String discount;
    private double price;
    private ParkinglotCalculator parkcalculator = new ParkinglotCalculator();

    public String calculate() {
        if ((hours == 0) && (discount.equals("free"))) {
            return "free";
        } else {
            price = parkcalculator.computePrice(hours, discount);
            return "parkingcalculator";
        }
    }
}

```

*free.xhtml:*

```

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        You park for free.
        <h:form>
            <h:commandButton value="return to the calculator"
                            action="parkingcalculator" />
        </h:form>
    </h:body>
</html>

```

**Question 4: JSF [10 points][10 minutes]**

Suppose I have the following calculator class:

```
public class Calculator {  
    private int value=0;  
  
    public int add(int newValue){  
        value=value+newValue;  
        return value;  
    }  
}
```

I want to use this calculator on the web, but I have 2 solutions to choose from: Solution A and Solution B, both given on the next page. Some of the servlet code in both solutions is intentionally omitted, but the main difference between Solution A and Solution B is the place where the Calculator class is instantiated.

Which of the following options is correct?

Only one option is allowed. Circle the correct answer

- a. Only Solution A works correctly
- b. Only Solution B works correctly
- c. Both Solution A and Solution B work correctly
- d. Both Solution A and Solution B do not work correctly

Explain your answer

**Your answer**

**Answer d is correct.**

**Solution A is not thread safe**

**In solution B, a new Calculator is created for every call, so the calculator starts with the value 0 for all requests.**

### **Solution A:**

```
public class CalculatorServlet extends HttpServlet {

    Calculator calc = new Calculator();

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

        // code is ommited

        String sNewValue = request.getParameter("newvalue");
        if (sNewValue != null){
            int iNewValue = Integer.parseInt(sNewValue);
            calcvalue = calc.add(iNewValue);
        }
        // code is ommited
    }
}
```

### **Solution B:**

```
public class CalculatorServlet extends HttpServlet {

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

        // code is ommited

        String sNewValue = request.getParameter("newvalue");
        if (sNewValue != null){
            int iNewValue = Integer.parseInt(sNewValue);
            Calculator calc = new Calculator();
            calcvalue = calc.add(iNewValue);
        }
        // code is ommited
    }
}
```

**Question 5: SCI [5 points][10 minutes]**

Describe how we can relate scope in a web application to the principles of SCI. Your answer should be about half a page, but should not exceed one page (handwritten). The number of points you get for this questions depend on how well you explain the relationship between scope in a web application and the principles of SCI

**Your answer:**