

Student ID _____ Student Name _____

Web Applications Architecture and Frameworks DE

Midterm Exam September 6, 2014

PRIVATE AND CONFIDENTIAL

1. Allotted exam duration is 2 hours.
2. Closed 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.

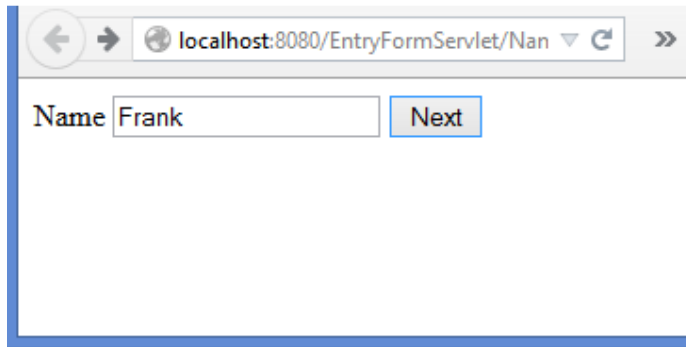
Question 1: [10 points] {10 minutes}

Select true or false for each of the following statements.

| | True or False? |
|---|----------------|
| In a HTTP GET request, the form parameters are added on the URL | true |
| The method response.sendRedirect() will not result in an extra roundtrip between the server and the client. The method requestDispatcher.forward() does result in an extra roundtrip between the server and the client. | false |
| The following servlet is thread save: public class AddServlet extends HttpServlet { int value; protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { // code is omitted String sNewValue = request.getParameter("newvalue"); if (sNewValue != null){ int iNewValue = Integer.parseInt(sNewValue); value = value + iNewValue; } // code is omitted } } | false |
| If you configure your browser such that you don't allow cookies on your computer, you cannot use the HttpSession for storing session state because the session ID is stored in a cookie. | false |
| For every servlet declared in web.xml we can have multiple servlet mappings such that requests with different URL's will be handled by the same servlet. | true |

Question 2: [25 points] {30 minutes}

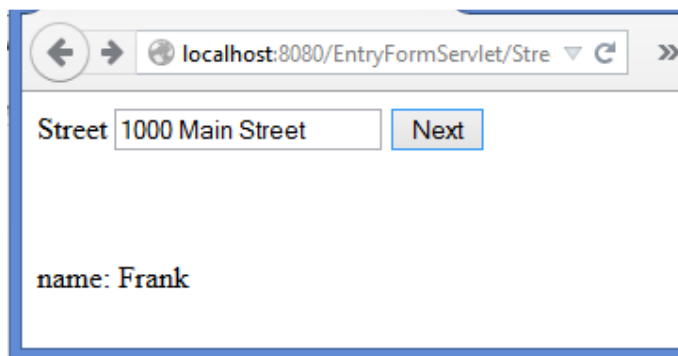
Write the following application with Servlets:



localhost:8080/EntryFormServlet/Nan

Name

The application first asks you to enter your name. Then you click the Next button

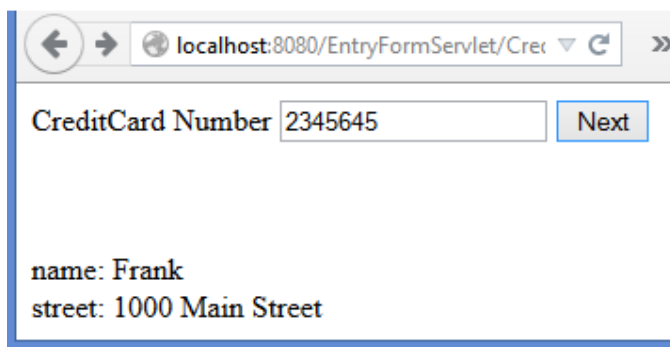


localhost:8080/EntryFormServlet/Stre

Street

name: Frank

The next page shows the entered name, and asks you to enter your street. Then you click the Next button.

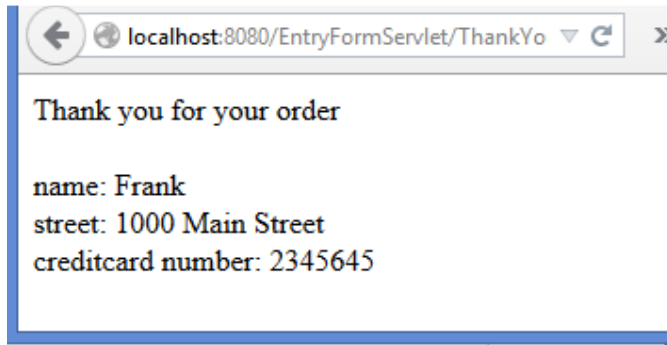


localhost:8080/EntryFormServlet/Cred

CreditCard Number

name: Frank
street: 1000 Main Street

The next page shows the entered name and street, and asks you to enter your credit card number. Then you click the Next button.



The next page shows a thank you message and the entered name, street, and credit card number.

Your implementation should follow the following requirements:

1. You are only allowed to use servlets.
2. You are **not** allowed to use JSP or JSF.
3. Do **not** write getter and setter methods

Complete the partial given code and write all other necessary classes:

```
@WebServlet(urlPatterns = {"/NameServlet"})
public class NameServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>NameServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form method='GET' action='StreetServlet'>");
        out.println("Name <input type='text' name='name' />");
        out.println("<input type='submit' value='Next' />");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

```

@WebServlet(urlPatterns = {"/StreetServlet"})
public class StreetServlet extends HttpServlet {

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String name = request.getParameter("name");
        HttpSession session = request.getSession();
        person = (Person) session.getAttribute("person");
        if (person == null) {
            person = new Person();
            session.setAttribute("person", person);
        }
        person.setName(name);

        out.println("<html>");
        out.println("<head>");
        out.println("<title>StreetServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form method='GET' action='CreditCardServlet'>");
        out.println("Street <input type='text' name='street' />");
        out.println("<input type='submit' value='Next'/>");
        out.println("</form>");
        out.println("<br/><br/>");
        out.println("name: " + person.getName() + "<br/>");
        out.println("</body>");
        out.println("</html>");
    }
}

```

```

@WebServlet(urlPatterns = {"/CreditCardServlet"})
public class CreditCardServlet extends HttpServlet {

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String street = request.getParameter("street");
        HttpSession session = request.getSession();
        person = (Person) session.getAttribute("person");
        if (person == null) {
            person = new Person();
            session.setAttribute("person", person);
        }
        person.setStreet(street);

        out.println("<html>");
        out.println("<head>");
        out.println("<title>StreetServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<form method='GET' action='ThankYouServlet'>");
        out.println("CreditCard Number <input type='text' name='ccnumber' />");
        out.println("<input type='submit' value='Next'/>");
        out.println("</form>");
        out.println("<br/><br/>");
        out.println("name: " + person.getName() + "<br/>");
        out.println("street: " + person.getStreet() + "<br/>");
        out.println("</body>");
        out.println("</html>");
    }
}

```

```

@WebServlet(urlPatterns = {"/ThankYouServlet"})
public class ThankYouServlet extends HttpServlet {

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String creditcardNumber = request.getParameter("ccnumber");
        HttpSession session = request.getSession();
        person = (Person) session.getAttribute("person");
        if (person == null) {
            person = new Person();
            session.setAttribute("person", person);
        }
        person.setCreditcard(creditcardNumber);

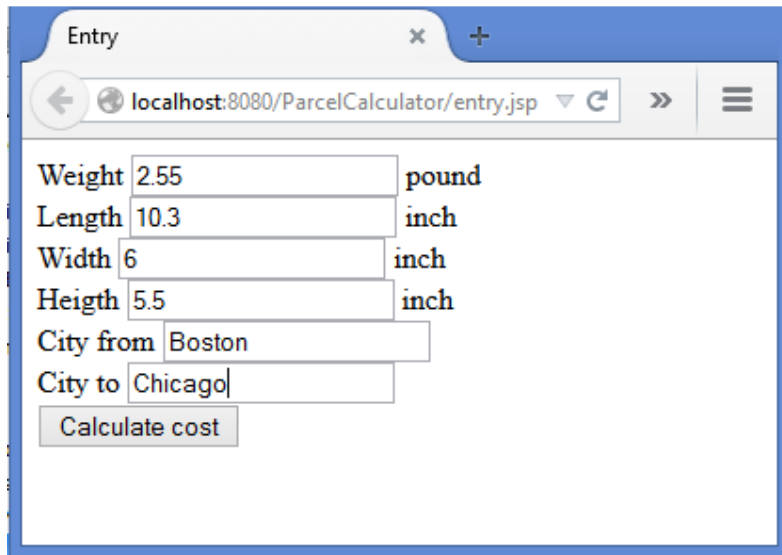
        out.println("<html>");
        out.println("<head>");
        out.println("<title>StreetServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("Thank you for your order");
        out.println("<br/><br/>");
        out.println("name: " + person.getName() + "<br/>");
        out.println("street: " + person.getStreet() + "<br/>");
        out.println("creditcard number: " + person.getCreditcard() + "<br/>");
        out.println("</body>");
        out.println("</html>");
    }
}

public class Person {
    private String name;
    private String street;
    private String creditcard;
    //getter and setter methods are not shown
}

```

Question 3. Servlet [30 points] {40 minutes}

Write a package shipping cost calculator application using Servlets and JSP's according the MVC structure. The application contains 2 pages: entry page and a result page:

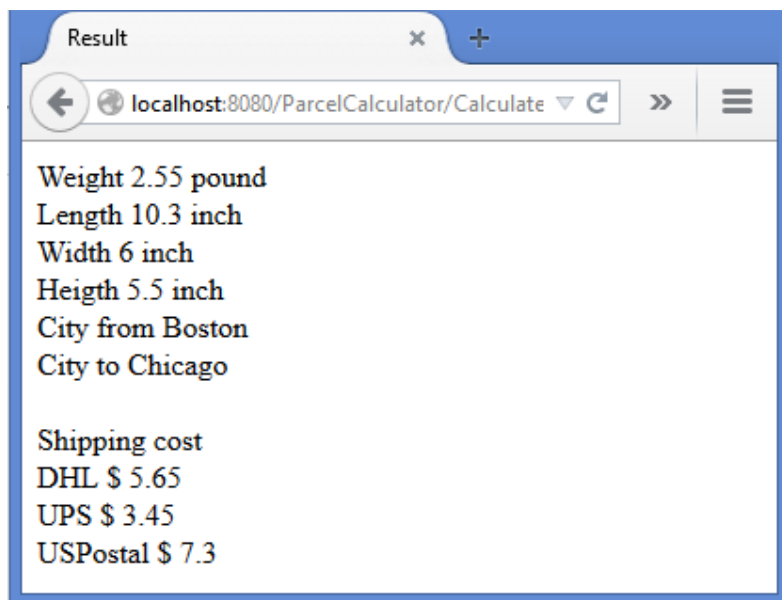


The screenshot shows a web browser window with the title 'Entry'. The address bar displays 'localhost:8080/ParcelCalculator/entry.jsp'. The page contains a form with the following fields and values:

| Field | Value | Unit |
|-----------|---------|-------|
| Weight | 2.55 | pound |
| Length | 10.3 | inch |
| Width | 6 | inch |
| Height | 5.5 | inch |
| City from | Boston | |
| City to | Chicago | |

Below the form is a button labeled 'Calculate cost'.

The entry page allows you to enter certain information about a package. When you click the Calculate cost button the result page is shown.



The screenshot shows a web browser window with the title 'Result'. The address bar displays 'localhost:8080/ParcelCalculator/Calculate'. The page displays the shipping cost calculation results:

Weight 2.55 pound
Length 10.3 inch
Width 6 inch
Height 5.5 inch
City from Boston
City to Chicago

Shipping cost
DHL \$ 5.65
UPS \$ 3.45
USPostal \$ 7.3

The result page shows how much it cost to ship this package with one of 3 different shippers (DHL, UPS and USPostal) .

The following methods are given:

```
public double getDHLShippingCost(double weight, double length, double width,
                                double height, String from, String to){
    // some complex algorithm
    return 5.65;
}
public double getUPSShippingCost(double weight, double length, double width,
                                double height, String from, String to){
    // some complex algorithm
    return 3.45;
}
public double getUSPostalShippingCost(double weight, double length, double width,
                                       double height, String from, String to){
    // some complex algorithm
    return 7.30;
}
```

You don't need to implement the complex algorithm to compute the shipping cost for each of the shippers. In our application we just return a hard coded amount.

Your implementation should follow the following requirements:

1. The application should follow the correct **Model-View-Controller** principles using Servlets, JSP's and Java classes.
 2. You are **not** allowed to use JSF.
 3. It is **not** allowed to write JAVA code in the JSP page (use JSTL and JSP EL).
- Complete the partial given code and write all other necessary classes:

entry.jsp

```
<html>
<head>
  <title>Entry</title>
</head>
<body>
  <form action="CalculateShippingServlet" method="post">
    Weight <input type="text" name="weight" size="20"> pound<br>
    Length <input type="text" name="length" size="20"> inch<br>
    Width <input type="text" name="width" size="20"> inch<br>
    Height <input type="text" name="height" size="20"> inch<br>
    City from <input type="text" name="from" size="20"><br>
    City to <input type="text" name="to" size="20"><br>
    <input type="submit" value="Calculate cost" />
  </form>
</body>
</html>
```

result.jsp

```
<html>
  <head>
    <title>Result</title>
  </head>
  <body>
    Weight ${param.weight} pound<br />
    Length ${param.length} inch<br/>
    Width ${param.width} inch<br/>
    Height ${param.height} inch<br/>
    City from ${param.from} <br/>
    City to ${param.to} <br/><br/>
    Shipping cost<br/>
    DHL $ ${dhl} <br/>
    UPS $ ${ups} <br/>
    USPostal $ ${uspostal} <br/>
  </body>
</html>
```

CalculateShippingServlet.java

```
@WebServlet(name = "CalculateShippingServlet", urlPatterns =
{"/CalculateShippingServlet"})
public class CalculateShippingServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String sweighth = request.getParameter("weight");
        double weight = Double.valueOf(sweighth);
        String slength = request.getParameter("length");
        double length = Double.valueOf(slength);
        String swidth = request.getParameter("width");
        double width = Double.valueOf(swidth);
        String sheigth = request.getParameter("height");
        double heigth = Double.valueOf(sheigth);
        String from = request.getParameter("from");
        String to = request.getParameter("to");

        ShippingCalculator shCalculator= new ShippingCalculator();
        double dhl = shCalculator.getDHLShippingCost(weight, length, width, weight,
from, to);
        double ups = shCalculator.getUPSShippingCost(weight, length, width, weight,
from, to);
        double uspostal = shCalculator.getUSPostalShippingCost(weight, length, width,
weight, from, to);

        RequestDispatcher view =request.getRequestDispatcher("result.jsp");
        request.setAttribute("dhl", dhl);
```

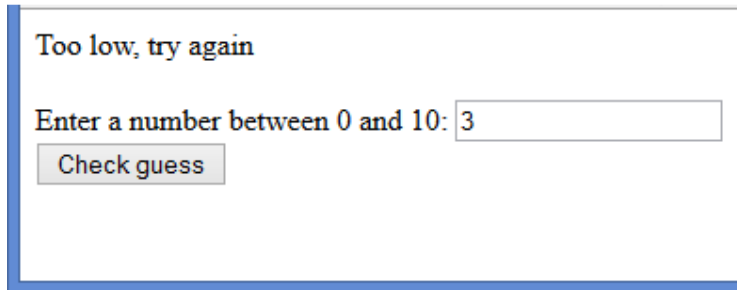
```
        request.setAttribute("ups", ups);
        request.setAttribute("uspostal", uspostal);
        view.forward(request, response);
    }
}
```

ShippingCalculator.java

```
public class ShippingCalculator {
    public double getDHLShippingCost(double weight, double length, double width,
double height, String from, String to){
        // some complex algorithm
        return 5.65;
    }
    public double getUPSShippingCost(double weight, double length, double width,
double height, String from, String to){
        // some complex algorithm
        return 3.45;
    }
    public double getUSPostalShippingCost(double weight, double length, double width,
double height, String from, String to){
        // some complex algorithm
        return 7.30;
    }
}
```

Question 4. JSF [30 points] {30 minutes}

Write the following guessnumber application using JSF. The number that you need to guess is hard coded in the JSF application. In this case the number to be guessed is 6.



Too low, try again

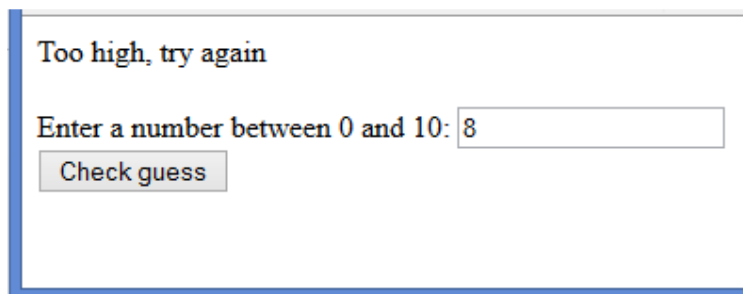
Enter a number between 0 and 10:

When you enter 3, the application shows that your guess is too low. The history of all guesses done so far with this application will be written in the console (using **System.out.println()**) of the application server every time we click the Check guess button:

INFO: History of guesses:

INFO: 3

Then we try a new guess of 8:



Too high, try again

Enter a number between 0 and 10:

In the console we see now:

INFO: History of guesses:

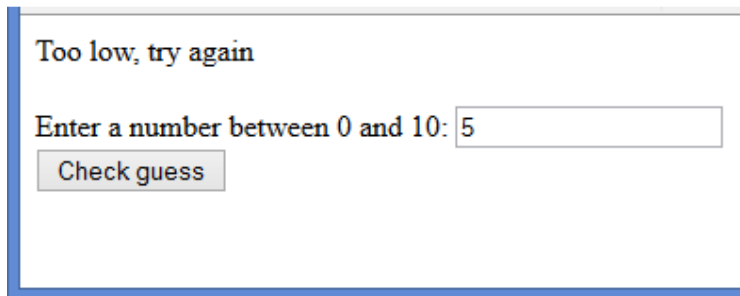
INFO: 3

INFO: History of guesses:

INFO: 3

INFO: 8

Then we try a new guess of 5:



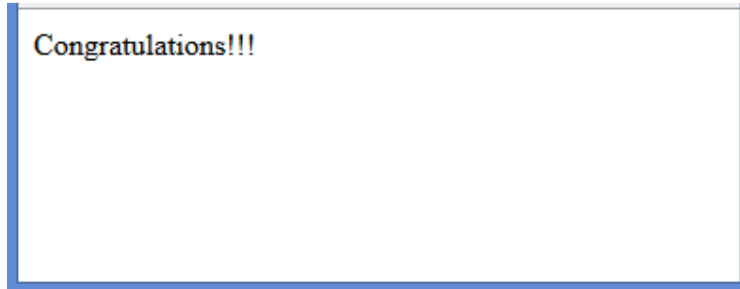
Too low, try again

Enter a number between 0 and 10:

In the console we see now:

```
INFO: History of guesses:  
INFO: 3  
INFO: History of guesses:  
INFO: 3  
INFO: 8  
INFO: History of guesses:  
INFO: 3  
INFO: 8  
INFO: 5
```

Then we try a new guess of 6:



Congratulations!!!

In the console we see now:

```
INFO: History of guesses:  
INFO: 3  
INFO: History of guesses:  
INFO: 3  
INFO: 8  
INFO: History of guesses:  
INFO: 3  
INFO: 8  
INFO: 5  
INFO: History of guesses:  
INFO: 3  
INFO: 8  
INFO: 5  
INFO: 6
```

Your implementation should follow the following requirements:

1. The application should follow the correct **Model-View-Controller** principles using JSF.
2. You are **not** allowed to use HTML, JSP's or servlets.
3. You do not need to validate the input.
4. For this JSF application we will use annotation based configuration, so there is no faces-config.xml file. **Add the necessary annotations to the code.**
5. **Do NOT write getter and setter methods!**

Complete the partial given code and write all other necessary classes:

guess.xhtml :

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Guess</title>
  </h:head>
  <h:body>
    <h:form>
      <h:outputText value="#{guessBean.result}"/><br /><br />
      Enter a number between 0 and 10: <h:inputText value="#{guessBean.guess}"/><br/>
      <h:commandButton value="Check guess"
        action="#{guessBean.check}" />
      <br /><br />
    </h:form>
  </h:body>
</html>
```

success.xhtml :

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Success</title>
  </h:head>
  <h:body>
    Congratulations!!! <br/>
  </h:body>
</html>
```

```

@Named
@RequestScoped
public class GuessBean {
    int guess = 0;
    int toBeGuessed = 6;
    String result = "";

    @Inject
    GuessHistory history;

    public String check(){
        history.add(guess);
        System.out.println("History of guesses:");
        for (Integer x : history.getGuesslist()){
            System.out.println(x);
        }
        if (guess == toBeGuessed){
            return "success";
        }
        if (guess < toBeGuessed){
            result = "Too low, try again";
        }
        if (guess > toBeGuessed){
            result = "Too high, try again";
        }
        return null;
    }

    //getter and setter methods are not shown
}

```

```
@SessionScoped
public class GuessHistory implements Serializable{
    private List<Integer> guesslist = new ArrayList<Integer>();

    public void add(int x){
        guesslist.add(x);
    }

    public List<Integer> getGuesslist() {
        return guesslist;
    }
}
```


Question 5. SCI [5 points] {10 minutes}

Describe how we can relate the concept of **scope** 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 question depend on how well you explain the relationship between **scope** and the principles of SCI.