Student ID_____**Student Name**_____

Web Applications Architecture and Frameworks DE

Midterm Exam May 18, 2013

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 ALLOTED TIME IS GIVEN FOR EVERY QUESTION.

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

Question 1: [7 points] {10 minutes}

Suppose I have written a servlet with the name **CalculatorServlet**. I want to use the url: **/calc** to call the CalculatorServlet.
Explain clearly what I need to do so that when my URL ends with /calc, the CalculatorServlet is called.

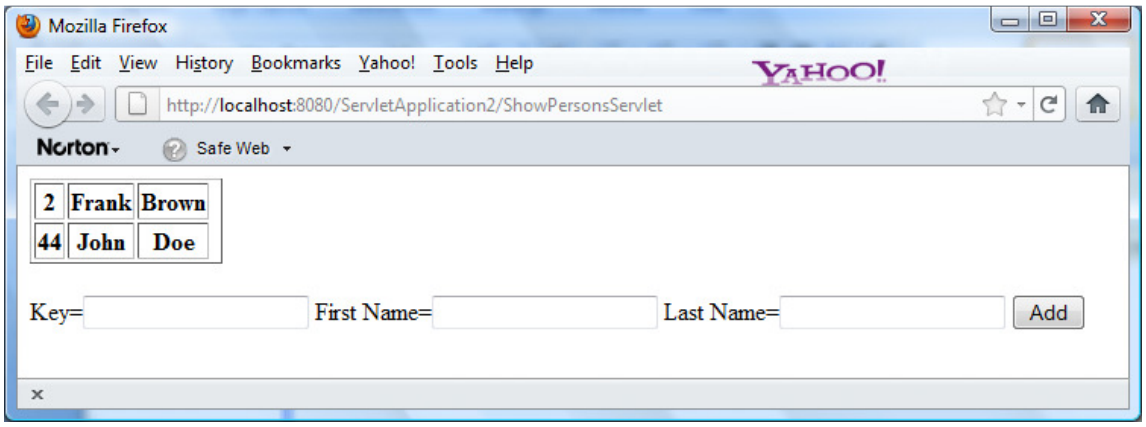
```
<servlet>
   <servlet-name> calculator </servlet-name>
   <servlet-class>CalculatorServlet</servlet-class>
 </servlet>
  <servlet-mapping>
   <servlet-name>calculator</servlet-name>
   <url-pattern>/calc</url-pattern>
  </servlet-mapping>
```

Question 2: [8 points] {10 minutes}

We learned that we can implement navigation in JSF with static navigation and with dynamic navigation. Give an example of each and explain how this works.

Question 3. Servlet [20 points] {20 minutes}

Write a web application using Servlets only (not JSP's) that allows you to add names to a list.



When you fill in the key, firstname and lastname, and click the add button, this person is then added to the list of persons shown on the page.

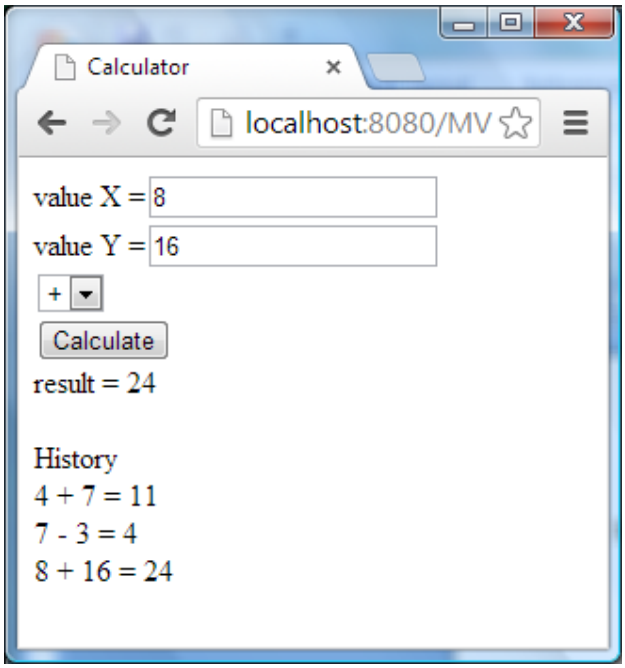
Complete the partial given code. Make sure you add all code that is necessary for the correct working of this application. You are only allowed to use Servlets and Java objects, not JSP's
Complete the partial given code. <u>Do NOT write getter and setter methods!</u>

```
public class Person {

   private String key;
   private String firstname;
   private String lastname;

   public Person(String key, String firstname, String lastname) {
         this.key = key;
         this.firstname = firstname;
         this.lastname = lastname;
  }
}
```

```
public class AddPersonServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
                                        throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      Collection<Person> personlist = new ArrayList<Person>();

      String key = request.getParameter("key");
      String firstname = request.getParameter("firstname");
      String lastname = request.getParameter("lastname");

      if (key != null && firstname != null && lastname != null) {
         HttpSession session = request.getSession();
         personlist = (Collection<Person>) session.getAttribute("list");
         if (personlist == null) {
            personlist = new ArrayList<Person>();
            session.setAttribute("list", personlist);
         }
         personlist.add(new Person(key, firstname, lastname));
      }

     out.println("<html>");
      out.println("<body>");
      out.println("<table  border='1'>");
      for (Person p : personlist) {
         out.println("<tr><th>" + p.getKey() + "</th><th>" + p.getFirstname() + "</th><th>"
+ p.getLastname() + "</th><th>");
      }
      out.println("</table>");
      out.println("<br>");

      out.println("<form method=GET action=AddPersonServlet>");
      out.println("Key=<input type=text name=key>");
      out.println("First Name=<input type=text name=firstname>");
      out.println("Last Name=<input type=text name=lastname>");
      out.println("<input type=submit value='Add'>");

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

   }
}
```

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

Write the following calculator application:



When you enter the x and y value, and then select the operator (can be + or -) and click the Calculate button, the page shows the result of the calculation, and also shows the history of all calculations done so far with this application.

Your implementation should follow the following requirements:

1. The calculator can only **add** and **subtract** 2 numbers
2. The application should follow the correct **Model-View-Controller** principles using Servlets, JSP's and Java classes.
3. You are **not** allowed to use JSF.
4. It is **not** allowed to write JAVA code in the JSP page.
5. You can assume the user only enters integers for the x and y value on the webpage. You do not need to validate the input.
6. You can see only the history of your own calculations and not the history of someone else's calculations.

Complete the partial given code:

calculator.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">


<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Calculator</title>
  </head>
  <body>
    <form method=GET action=CalcServlet>
     value X =<input type=text name=x><br/>
     value Y =<input type=text name=y><br/>
     <select name="operator">
      <option label="+" value="+">+</option>
      <option label="-" value="-">-</option>
     </select>
     <br/>
     <input type=submit value='Calculate'><br/>
     </form>
       result = ${result}
     <br/>
     <br/>
     History
     <br/>
     <c:forEach var="calculation" items="${history}">
       ${calculation}<br/>
     </c:forEach>
  </body>
</html>
```

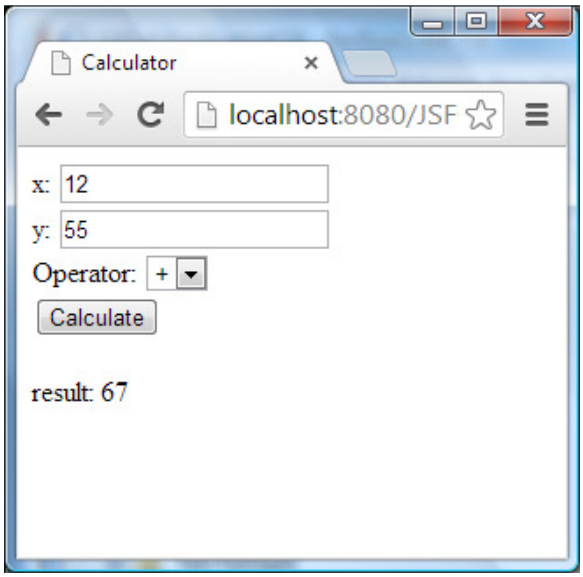
CalcServlet.java

```
public class CalcServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html");

      String strX = request.getParameter("x");
      String strY = request.getParameter("y");
      String operator = request.getParameter("operator");
      int firstnumber = Integer.parseInt(strX);
      int secondnumber = Integer.parseInt(strY);
      int result = 0;
      Calculator calculator = new Calculator();
      result = calculator.calculate(operator, firstnumber, secondnumber);

      request.setAttribute("result", result);

      HttpSession session = request.getSession();
      List<String> historylist = (List<String>) session.getAttribute("history");
      if (historylist == null) {
         historylist = new ArrayList<String>();
         session.setAttribute("history", historylist);
      }
      String calculation = strX +" "+ operator +" "+ strY +" = "+ result;
      historylist.add(calculation);

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

Calculator.java

```
public class Calculator {

   public int add(int x, int y) {
      return x + y;
   }

   public int subtract(int x, int y) {
      return x - y;
   }

   int calculate(String operator, int firstnumber, int secondnumber) {
      if (operator.equals("+")) {
         return add(firstnumber, secondnumber);
      }
      if (operator.equals("-")) {
         return subtract(firstnumber, secondnumber);
      }
      return 0;
   }
}
```

Question 5. JSF [30 points] {40 minutes}
Write the following calculator application using JSF :



When you enter the x and y value, and then select the operator (can be + or -) and click the Calculate button, the page shows the result of the calculation.

The history of all calculations 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 Calculate button:

INFO: History of calculations:
INFO: 3 + 4 = 7
INFO: History of calculations:
INFO: 3 + 4 = 7
INFO: 8 - 4 = 4
INFO: History of calculations:
INFO: 3 + 4 = 7
INFO: 8 - 4 = 4
INFO: 12 + 55 = 67

Your implementation should follow the following requirements:

1. The calculator can only **add** and **subtract** 2 numbers
2. The application should follow the correct **Model-View-Controller** principles using JSF.
3. You are **not** allowed to use HTML, JSP's or servlets.
4. You can assume the user only enters integers for the x and y value on the webpage. You do not need to validate the input.
5. You can see only the history of your own calculations and not the history of someone else's calculations.
6. 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**.
7. **Do NOT write getter and setter methods!**

Complete the partial given code:

calculator.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core">
  <h:head>
    <title>Calculator</title>
  </h:head>
  <h:body>
    <h:form>
      x: <h:inputText value="#{calcManagedBean.x}"/><br/>
      y: <h:inputText value="#{calcManagedBean.y}"/><br/>
      Operator: <h:selectOneMenu value="#{calcManagedBean.operator}">
        <f:selectItem itemLabel="+" itemValue="+"/>
        <f:selectItem itemLabel="-" itemValue="-"/>
      </h:selectOneMenu>
      <br />
      <h:commandButton value="Calculate"
                action="#{calcManagedBean.calculate}" />
      <br /><br />
      result: <h:outputText value="#{calcManagedBean.result}"/>

    </h:form>
  </h:body>
</html>
```

```
@Named
@RequestScoped
public class CalcManagedBean {

   int x;
   int y;
   String operator;
   int result;

   @Inject private Calculator calculator;
   @Inject private CalcHistory calcHistory;

   public String calculate(){
      result = calculator.calculate(operator, x, y);
      calcHistory.add(x+" "+operator+" "+y+" = "+result);
      System.out.println("History of calculations:");
      for (String hcalc : calcHistory.getHistorylist() ){
         System.out.println(hcalc);
      }
      return null;
   }
}

public class Calculator {

   public int add(int x, int y) {
      return x + y;
   }

   public int subtract(int x, int y) {
      return x - y;
   }

   int calculate(String operator, int firstnumber, int secondnumber) {
      if (operator.equals("+")) {
         return add(firstnumber, secondnumber);
      }
      if (operator.equals("-")) {
         return subtract(firstnumber, secondnumber);
      }
      return 0;
   }
}
```

```
@Named
@SessionScoped
public class CalcHistory implements Serializable{
   List<String> historylist= new ArrayList<String>();

   public void add(String calculation){
      historylist.add(calculation);
   }
}
```

Question 6. SCI [5 points] {10 minutes}

Describe how we can relate the concept of **scope** (request, session, etc) 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.