

Student ID _____ Student Name _____

Distributed Computing DE Midterm Exam June 25, 2011

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.

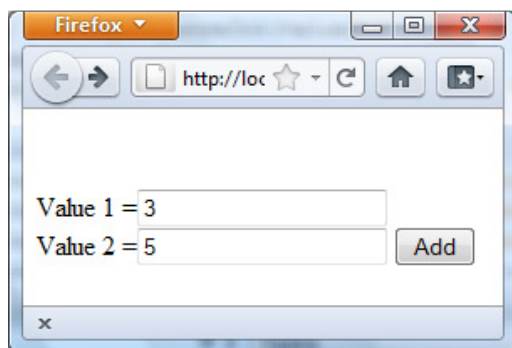
3 blank pages are provided for writing the solutions and/or scratch paper. All 3 pages must be handed in with the exam

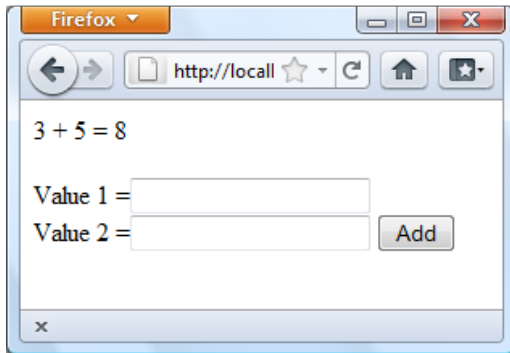
BE VERY CAREFUL WITH THE GIVEN 2 HOURS AND USE YOUR TIME WISELY. THE ALLOTTED TIME IS GIVEN FOR EVERY QUESTION.

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

Question 1: Servlet [15 points][15 minutes]

Write a servlet that shows a form that allows you to enter 2 numbers (integers). If you click the Add button, the servlet should show the result of the calculation, and allows you to add 2 other integers:





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

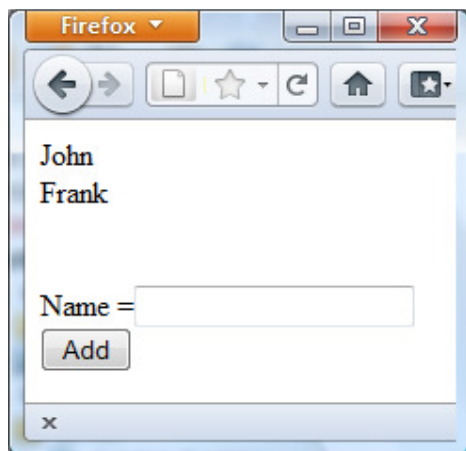
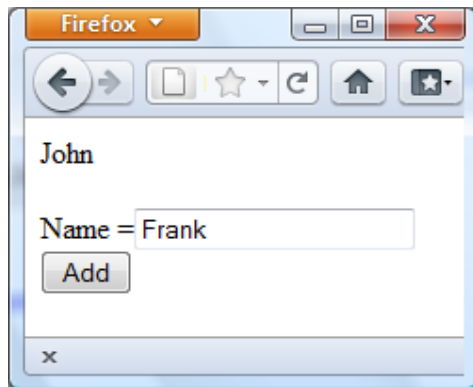
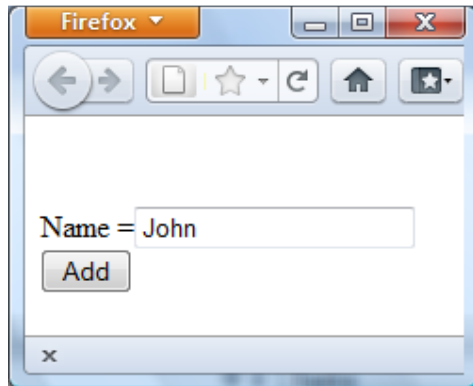
```
public class CalculatorServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        String value1 = (String) request.getParameter("value1");
        String value2 = (String) request.getParameter("value2");

        if (value1 != null || value2 != null) {
            int x = Integer.parseInt(value1);
            int y = Integer.parseInt(value2);
            int result = x + y;
            out.println(value1 + " + " + value2 + " = " + result);
        }
        out.println("<br/><br/>");
        out.println("<form method=GET action=CalculatorServlet>");
        out.println("Value 1 =<input type=text name=value1> <br>");
        out.println("Value 2 =<input type=text name=value2>");
        out.println("<input type=submit value='Add'>");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Question 2: Servlet [20 points][25 minutes]

Write a servlet that allows you to add names to a list of names. The servlet shows the list of all names added to the application. It is important that the names that are being added to the list are only visible to the user that has added the names, and that another user only sees his or her own list of names:



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

```

public class NameServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");

        Collection<String> namelist = null;

        String name = (String) request.getParameter("name");

        if (name != null) {
            HttpSession session = request.getSession();

            namelist = (Collection<String>) session.getAttribute("namelist");
            if (namelist == null) {
                namelist = new ArrayList();
                session.setAttribute("namelist", namelist);
            }
            namelist.add(name);

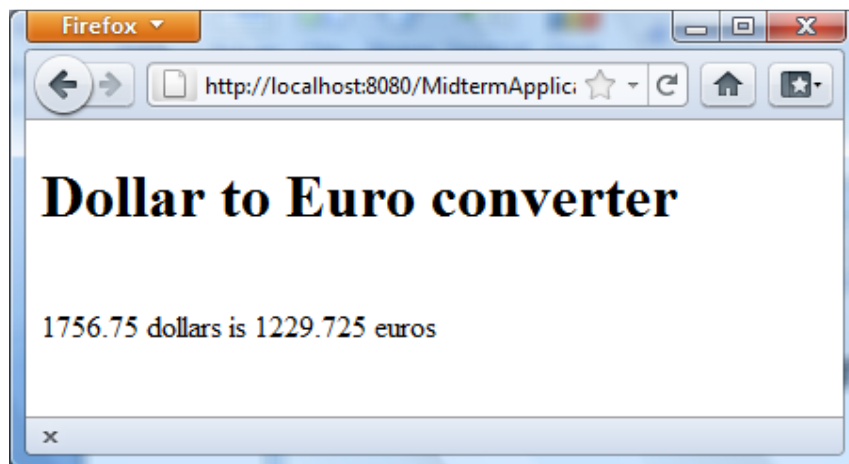
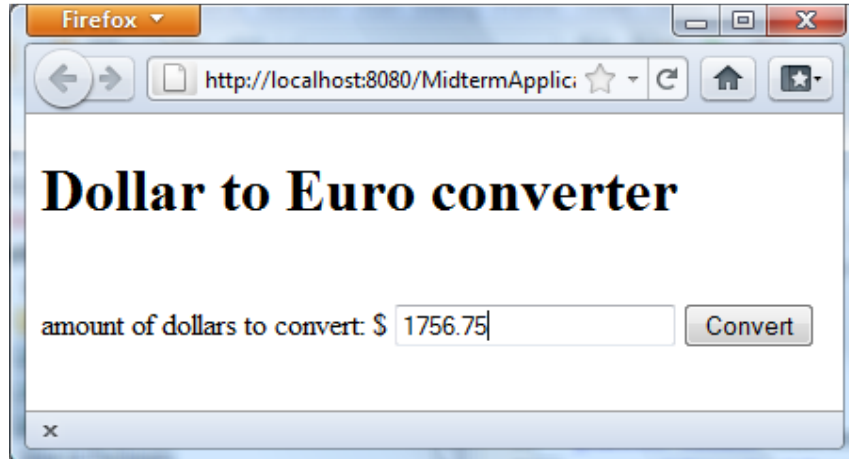
            for (String thename : namelist) {
                out.println(thename);
                out.println("<br/>");
            }
        }

        out.println("<br/><br/>");
        out.println("<form method=POST action=NameServlet>");
        out.println("Name =<input type=text name=name> <br />");
        out.println("<input type=submit value='Add'>");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}

```

Question 3: Model-View-Controller [25 points][30 minutes]

Write the following “Dollar to Euro converter” application in the **Model-View-Controller** style:



The application allows you to convert an entered amount in dollars into the European currency euros. For this application only the partial code of the 2 JSP files are given. Complete the code of the JSP files, and add all code that is necessary for the correct working of this application. For your solution you can use the formula that 1 dollar = 0.7 euro.

input.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!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>Dollar to Euro converter</title>
  </head>
  <body>
    <h1>Dollar to Euro converter</h1>
    <br />

    <form method=GET action=ConvertServlet>
      amount of dollars to convert: $ <input type=text name=amount />
      <input type=submit value=Convert>
    </form>

  </body>
</html>
```

output.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!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>Dollar to Euro converter</title>
  </head>
  <body>
    <h1>Dollar to Euro converter</h1>
    <br />

    <%= request.getAttribute("dollarAmount") %> dollars is
    <%= request.getAttribute("euroAmount") %> euros

  </body>
</html>
```

Your other code for the converter application:

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

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

        String amount = (String) request.getParameter("amount");
        double doubleAmount = Double.parseDouble(amount);

        Converter converter = new Converter();
        double euroAmount = converter.convert(doubleAmount);

        request.setAttribute("euroAmount", euroAmount);
        request.setAttribute("dollarAmount", doubleAmount);
        RequestDispatcher dispatcher = request.getRequestDispatcher("output.jsp");
        dispatcher.forward(request, response);
    }
}

public class Converter {
    public double convert (double amount){
        return 0.7 * amount;
    }
}
```

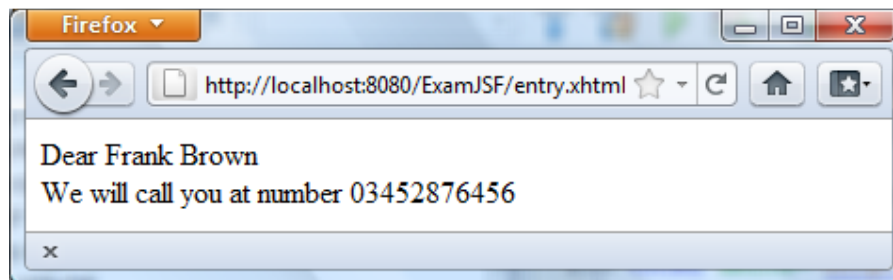
Question 4: JSF [25 points][25 minutes]

Write the following Contact Customer Support application in JSF. The entry page shows the following form:



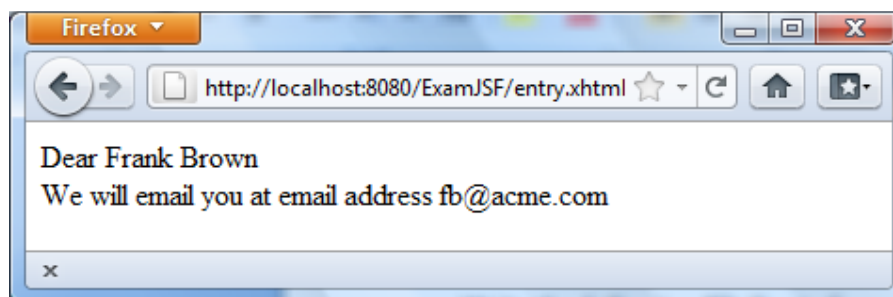
A screenshot of a Firefox browser window displaying a web form titled "Contact Customer Support". The form contains three input fields: "Name" with the value "Frank Brown", "Phone" with the value "03452876456", and "Email" with the value "fb@acme.com". Below the input fields are two buttons: "call me" and "send email". The browser's address bar shows the URL "http://localhost:8080/ExamJSF/entry.xhtml".

If you click the *call me* button, you get the following page:



A screenshot of a Firefox browser window showing a confirmation message. The message reads: "Dear Frank Brown" followed by "We will call you at number 03452876456". The browser's address bar shows the URL "http://localhost:8080/ExamJSF/entry.xhtml".

If you click the *send email* button, you get the following page:



A screenshot of a Firefox browser window showing a confirmation message. The message reads: "Dear Frank Brown" followed by "We will email you at email address fb@acme.com". The browser's address bar shows the URL "http://localhost:8080/ExamJSF/entry.xhtml".

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

entry.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">
  <h:head>
    <title> Contact Customer Support</title>
  </h:head>
  <h:body>
    <h1>Contact Customer Support</h1>
    <h:form>
      Name <h:inputText value="#{customer.name}" /><br/>
      Phone <h:inputText value="#{customer.phone}" /><br/>
      Email <h:inputText value="#{customer.email}" /><br/>
      <h:commandButton value="call me" action="#{customer.callme}" />
      <h:commandButton value="send email" action="#{customer.sendmail}" />
    </h:form>
  </h:body>
</html>
```

call.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">
  <h:head>
    <title>Phone contact</title>
  </h:head>
  <h:body>
    Dear <h:outputText value="#{customer.name}" /><br/>
    We will call you at number <h:outputText value="#{customer.phone}" /><br/>
  </h:body>
</html>
```

email.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">
  <h:head>
    <title>Email contact</title>
  </h:head>
  <h:body>
    Dear <h:outputText value="#{customer.name}" /><br/>
    We will email you at email address <h:outputText value="#{customer.email}"
/><br/>

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

web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.xhtml</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>faces/index.xhtml</welcome-file>
  </welcome-file-list>
</web-app>
```

Customer.java

You don't need to write out getter and setter methods.

@ManagedBean

@RequestScoped

```
public class Customer {  
    private String name;  
    private String phone;  
    private String email;  
  
    public String getEmail() {  
        return email;  
    }  
    public void setEmail(String email) {  
        this.email = email;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public String getPhone() {  
        return phone;  
    }  
    public void setPhone(String phone) {  
        this.phone = phone;  
    }  
    public String callme(){  
        return "call";  
    }  
    public String sendmail(){  
        return "email";  
    }  
}
```

Question 5: JSF [10 points][15 minutes]

- a. What are the different scopes we can apply to a JSF managed bean, and what are the differences between these scopes?

Your answer:

Request scope: Objects in request scope are available in one request-response cycle

Session Scope: Objects in Session scope are available during a session

Application Scope: objects in Application scope are available for all users that use this application

- b. What is the difference between static navigation and dynamic navigation in JSF? Give an example of both.

Your answer:

Static: clicking a particular button always selects a fixed JSF page for rendering the response.

```
<h: commandButton label ="Login" action="welcome"/>
```

Dynamic: If you click a button, a method on the backing bean is called, and this method returns a String. The page that is navigated to depends on the particular String that is returned.

```
<h: commandButton label="Login" action="#{loginController.verifyUser}"/>
```

```
String verifyUser() {  
    if (. ..)  
        return "success";  
    else  
        return "failure"  
}
```

Question 6: SCI [5 points][10 minutes]

Describe how we can relate Model-View-Controller 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 Model-View-Controller and the principles of SCI

Your answer: