Student ID___________________Student Name____________________________________

Web Applications Architecture and Frameworks DE

Midterm Exam May 10, 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 ALLOTED TIME IS GIVEN FOR EVERY QUESTION.

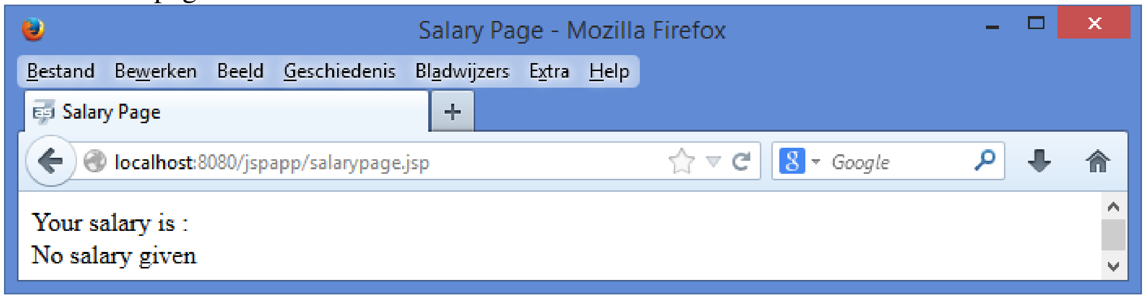
Write your name and student id at the top of this page.

Question 1: [10 points] {10 minutes}
Select true of false for each of the following statements.

	True or False?
The ServletContext is thread safe.	False
The method **response.sendRedirect()** will result in an extra roundtrip between the server and the client. The method **requestDispatcher.forward()** does not result in an extra roundtrip between the server and the client.	True
Every servlet in a web application has access to its own instance of the ServletContext.	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
If our Java code throws an exception, then we can specify in the web.xml file which web page is shown for this particular exception.	True

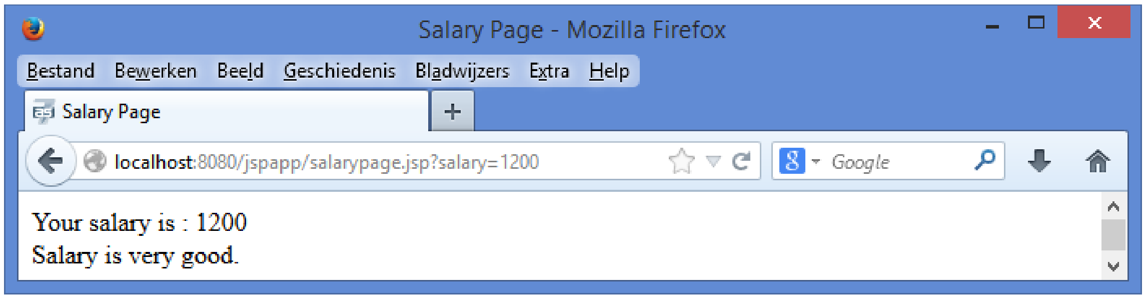
Question 2: [15 points] {10 minutes}

Write a JSP page that works as follows:


Salary Page - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Salary Page
localhost:8080/jspapp/salarypage.jsp

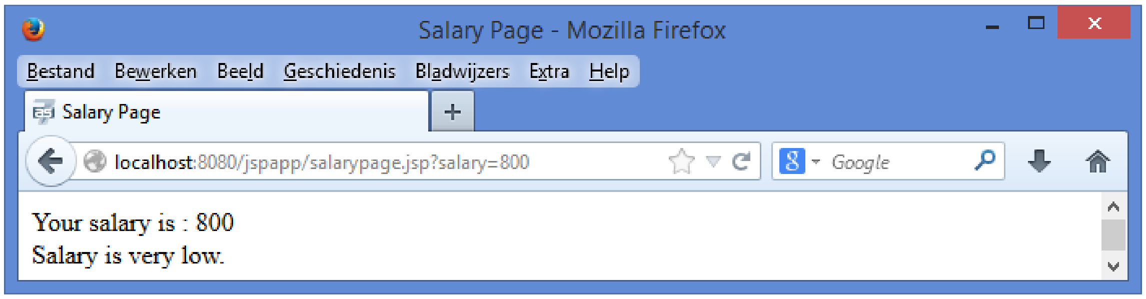
Your salary is :
No salary given

When you do not provide a parameter on the URL with the name salary, the page shows that no salary is given.


Salary Page - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Salary Page
localhost:8080/jspapp/salarypage.jsp?salary=1200

Your salary is : 1200
Salary is very good.

When you provide the parameter salary=1200, the page shows that the salary is very good.


Salary Page - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Salary Page
localhost:8080/jspapp/salarypage.jsp?salary=800

Your salary is : 800
Salary is very low.

When you provide the parameter salary=800, the page shows that the salary is very low.

Your implementation should follow the following requirements:

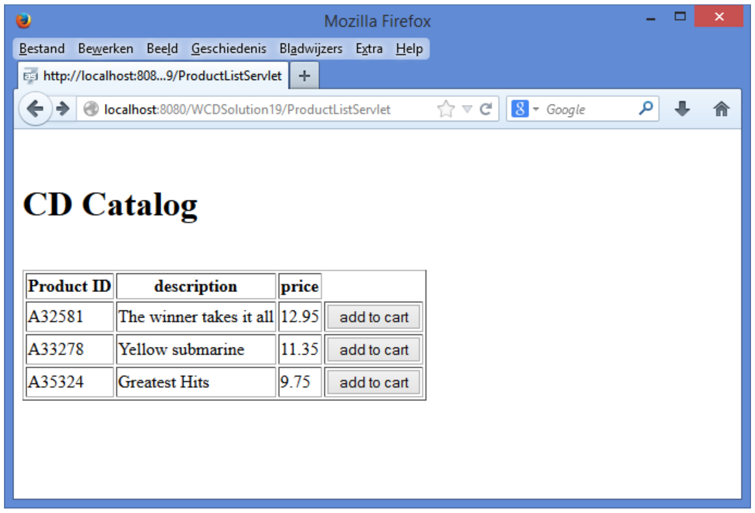
1. The salary is very good if the salary is bigger than 1000 or equal to 1000.
2. The salary is very low if the salary is smaller than 1000.
3. You are **not** allowed to use servlets or JSF.
4. It is **not** allowed to write JAVA code in the JSP page (use JSTL and JSP EL).

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

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Salary Page</title>
  </head>
  <body>
    Your salary is : ${param.salary}<br/>
    <c:choose>
      <c:when test="${param.salary <= 1000}">
        Salary is very low.
      </c:when>
      <c:when test="${param.salary > 1000}">
        Salary is very good.
      </c:when>
      <c:otherwise>
        No salary given
      </c:otherwise>
    </c:choose>
  </body>
</html>
```

Question 3. Servlet [50 points] {60 minutes}

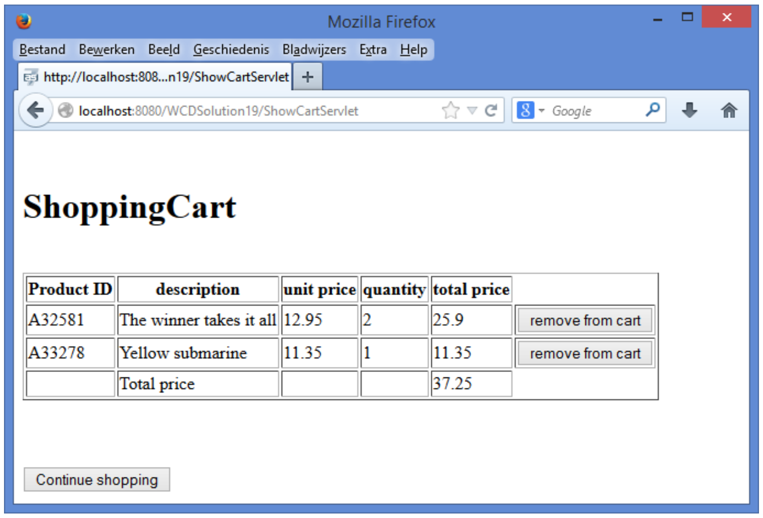
Write a cd webshop application using Servlets and JSP's according the MVC structure. The application contains 2 pages: a cd catalog page and a shoppingcart page:


Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
http://localhost:808...9/ProductListServlet
localhost:8080/WCDSolution19/ProductListServlet
Google

CD Catalog

Product ID	description	price	
A32581	The winner takes it all	12.95	add to cart
A33278	Yellow submarine	11.35	add to cart
A35324	Greatest Hits	9.75	add to cart

The cd catalog page shows a list of CD's and when you click the "**add to cart**" button, the cd is added to the shoppingcart, and the shoppingcart page is shown.


Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
http://localhost:808...n19/ShowCartServlet
localhost:8080/WCDSolution19/ShowCartServlet
Google

ShoppingCart

Product ID	description	unit price	quantity	total price	
A32581	The winner takes it all	12.95	2	25.9	remove from cart
A33278	Yellow submarine	11.35	1	11.35	remove from cart
	Total price			37.25	

Continue shopping

The **shoppingcart page** shows the content of the shoppingcart. When you click the "**remove from cart**" button, and the quantity is 1, then this cd is removed from the cart. When you click the "**remove from cart**" button, and the quantity is higher than 1, then the quantity is decreased with 1. When you click the "**Continue shopping**" button, the **cd catalog page** is shown.

Your implementation should follow the following requirements:

5. The application should follow the correct **Model-View-Controller** principles using Servlets, JSP's and Java classes.
6. You are **not** allowed to use JSF.
7. 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:

showproducts.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.*, domain.*"%>
<%@ 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>
<body>

<BR>
<h1>CD Catalog</h1>
<BR>
<TABLE BORDER=1>
      <TH>Product ID</TH>
      <TH>description</TH>
      <TH>price</TH>
      <c:forEach var="product" items="${pcatalog.productlist}">
            <TR>
                  <TD>${product.itemcode}</TD>
                  <TD>${product.name}</TD>
                  <TD>${product.itemprice}</TD>
                  <TD>
                  <form method='get' action='AddToCartServlet'>
                    <input type='hidden' name='productid'
value='${product.itemcode}' />
                    <input type='submit' value='add to cart' />
            </form>
                  </TD>
            </TR>
      </c:forEach>
</TABLE>
</body>
</html>
```

showcart.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.*,domain.*"%>
<%@ 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>
<body>
<BR>
<h1>ShoppingCart</h1>
<BR>
<TABLE BORDER=1>
      <TH>Product ID</TH>
      <TH>description</TH>
      <TH>unit price</TH>
      <TH>quantity</TH>
      <TH>total price</TH>
      <c:forEach var="cartitem" items="${cart.cartlist}">
      <TR>
            <TD>${cartitem.product.itemcode}</TD>
            <TD>${cartitem.product.name}</TD>
            <TD>${cartitem.product.itemprice}</TD>
            <TD>${cartitem.quantity}</TD>
            <TD>${cartitem.price}</TD>
            <TD>
            <form method='get' action='RemoveFromCartServlet'>
              <input type='hidden' name='productid'
value='${cartitem.product.itemcode}' />
              <input type='submit' value='remove from cart' /></form>
            </TD>
      </TR>
      </c:forEach>
      <TR>
            <TD></TD>
            <TD>Total price</TD>
            <TD></TD>
            <TD></TD>
            <TD>${cart.totalPrice}</TD>
      </TR>
</TABLE>
<br />
<br />
<br />
<form method='get' action='ProductListServlet'>
              <input type='submit' value='Continue shopping' /></form>
</body>
</html>
```

ProductListServlet

```
public class ProductListServlet extends HttpServlet {

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

            HttpSession session = request.getSession(true);
          session.setAttribute("pcatalog", pcatalog);

            RequestDispatcher rd =
request.getRequestDispatcher("showproducts.jsp");
            rd.forward(request, response);
      }
}
```

ShowCartServlet

```
public class ShowCartServlet extends HttpServlet {

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


            HttpSession session = request.getSession(true);

            ShoppingCart cart = (ShoppingCart)
                          session.getAttribute("cart");
            if (cart == null){
                  cart = new ShoppingCart();
                  session.setAttribute("cart", cart);
            }

            // show the cart
            RequestDispatcher rd =
             request.getRequestDispatcher("showcart.jsp");
            rd.forward(request, response);

      }

}
```

AddToCartServlet

```
public class AddToCartServlet extends HttpServlet {

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


            // get the productid
            String productid= request.getParameter("productid");
            //get the session
            HttpSession session = request.getSession(true);
            //get the cart from the session
            ShoppingCart cart = (ShoppingCart)
                  session.getAttribute("cart");
            if (cart == null){
                  cart = new ShoppingCart();
                  session.setAttribute("cart", cart);
            }
            //get the catalog from the session
            ProductCatalog pcatalog = (ProductCatalog)
                  session.getAttribute("pcatalog");
            if (pcatalog == null){
                  pcatalog = new ProductCatalog();
                  session.setAttribute("pcatalog", pcatalog);
            }

            //get product and add to the cart
            Product product = pcatalog.getProduct(productid);
            cart.addToCart(product);
            // show the cart
            response.sendRedirect("ShowCartServlet");
      }

}
```

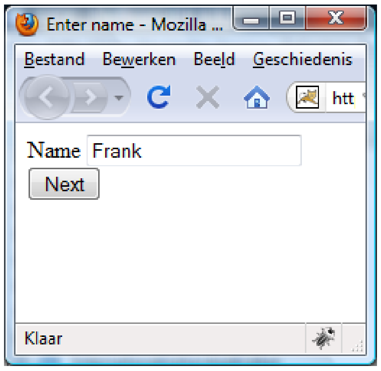
RemoveFromCartServlet

```
public class RemoveFromCartServlet extends HttpServlet {

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

            // get the productid
            String productid= request.getParameter("productid");
            System.out.println("productid="+productid);
            //get the session
            HttpSession session = request.getSession(true);
            //get the cart from the session
            ShoppingCart cart = (ShoppingCart)
                  session.getAttribute("cart");
            if (cart == null){
                  cart = new ShoppingCart();
                  session.setAttribute("cart", cart);
            }
            //remove product from the cart
            cart.removeFromCart(productid);

            // show the cart
            response.sendRedirect("ShowCartServlet");

            }

}
```

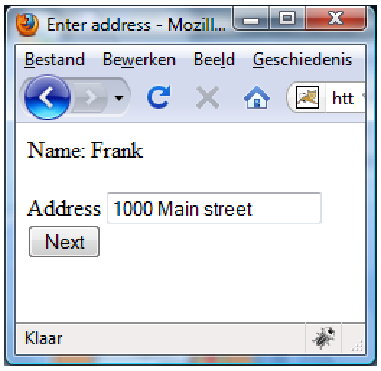
```
public class ProductCatalog {
    private Collection<Product> productlist = new ArrayList<Product>();

      public ProductCatalog(){
        productlist.add(new Product("A32581","The winner takes it
             all",12.95));
        productlist.add(new Product("A33278","Yellow submarine",11.35));
        productlist.add(new Product("A35324","Greatest Hits",9.75));
    }

      public Product getProduct(String productid) {
            Product p = null;
          for (Product product : productlist) {
            if (product.getItemcode().equals(productid)){
                  p= product;
                break;
            }
          }
            return p;
      }
      //constructor, getter and setter methods are not shown
}

public class Product {
    private String itemcode;
    private String name;
    private double itemprice;
    //constructor, getter and setter methods are not shown
}


public class Cartitem {
    private int quantity;
    private Product product;
    private double price=0;
    //constructor, getter and setter methods are not shown
}
```

```
public class ShoppingCart {
    private List cartlist = new LinkedList();
    private double totalPrice= 0;

    public ShoppingCart() {
    }

    public void addToCart(Product product){
        boolean found=false;
        // first check if product is already in the shoppingcart
        Iterator iter = cartlist.iterator();
        while (iter.hasNext()){
            Cartitem cartitem = (Cartitem) iter.next();
            Product prod = cartitem.getProduct();
            if (prod.getItemcode().equals(product.getItemcode())){
                cartitem.setQuantity(cartitem.getQuantity()+1);
                found=true;
                break;
            }
        }
        if (!found){
          cartlist.add(new Cartitem(product,1));
        }
        computeTotalPrice();
    }

    public void computeTotalPrice(){
        totalPrice = 0;
        Iterator iter = cartlist.iterator();
        while (iter.hasNext()){
            Cartitem cartitem = (Cartitem) iter.next();
            totalPrice = totalPrice + (cartitem.getQuantity()*
cartitem.getProduct().getItemprice());
        }
    }

    public void removeFromCart(String itemcode){
        Iterator iter = cartlist.iterator();
        while (iter.hasNext()){
            Cartitem cartitem = (Cartitem) iter.next();
            Product product = cartitem.getProduct();
            if (product.getItemcode().equals(itemcode)){
                  if (cartitem.getQuantity()== 1){
                    iter.remove();
                    computeTotalPrice();
                    break;
                  }
                  else{
                       cartitem.setQuantity(cartitem.getQuantity()-1);
                  }
            }
        }
        computeTotalPrice();
    }
    //constructor, getter and setter methods are not shown
}
```

Question 5. JSF [20 points] {30 minutes}
Write the following application using JSF :

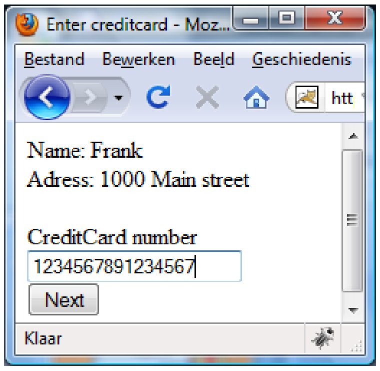


Name Frank
Next


Enter address - Mozill...
Bestand Bewerken Beeld Geschiedenis
htt
Klaar

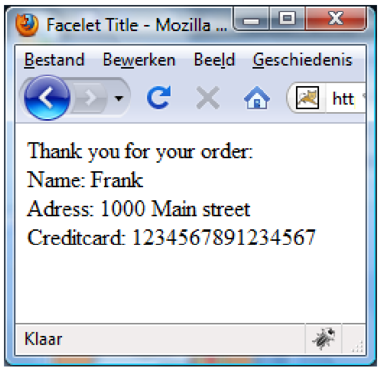
Name: Frank

Address 1000 Main street
Next


Enter creditcard - Moz...
Bestand Bewerken Beeld Geschiedenis
htt
Klaar

Name: Frank
Adress: 1000 Main street

CreditCard number
1234567891234567
Next


Facelet Title - Mozilla ...
Bestand Bewerken Beeld Geschiedenis
htt
Klaar

Thank you for your order:
Name: Frank
Adress: 1000 Main street
Creditcard: 1234567891234567

The first page asks you to enter your name. When you click the Next button, the second page is shown.
The second page shows the entered name and asks you to enter your address. When you click the Next button, the third page is shown.
The third page shows the entered name and address and asks you to enter your credit card number. When you click the Next button, the last page is shown.
The last page shows the entered name, address and credit card number.

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:

```
@ManagedBean
@SessionScoped
public class OrderBean {
   private String name;
   private String address;
   private String creditcard;
   //getter and setter methods are not shown
}

@ManagedBean
@RequestScoped
public class Namebean {
   @ManagedProperty(value="#{orderBean}")
   private OrderBean order;
   //getter and setter methods are not shown
}

@ManagedBean
@RequestScoped
public class Addressbean {
   @ManagedProperty(value="#{orderBean}")
   private OrderBean order;
  //getter and setter methods are not shown
}

@ManagedBean
@RequestScoped
public class Creditcardbean {
   @ManagedProperty(value="#{orderBean}")
   private OrderBean order;
  //getter and setter methods are not shown
}

@ManagedBean
@RequestScoped
public class Confirmationbean {
   @ManagedProperty(value="#{orderBean}")
   private OrderBean order;
  //getter and setter methods are not shown
}
```

name.xhtml (only the body of the xhtml page is shown):

```
      <h:form>
          Name <h:inputText value="#{namebean.order.name}"/>
          <br />
          <h:commandButton value="Next" action="address.xhtml"/>
       </h:form>
```

address.xhtml (only the body of the xhtml page is shown):

```
<h:form>
         Name: <h:outputText value="#{addressbean.order.name}"/>
         <br />
         <br />
          Address <h:inputText value="#{addressbean.order.address}"/>
          <br />
          <h:commandButton value="Next" action="creditcard.xhtml"/>
       </h:form>
```

creditcard.xhtml (only the body of the xhtml page is shown):

```
<h:form>
         Name: <h:outputText value="#{creditcardbean.order.name}"/><br />
         Adress: <h:outputText value="#{creditcardbean.order.address}"/>
         <br />
         <br />
          CreditCard number <h:inputText
                      value="#{creditcardbean.order.creditcard}"/>
          <br />
          <h:commandButton value="Next" action="confirmation.xhtml"/>
       </h:form>
```

confirmation.xhtml (only the body of the xhtml page is shown):

```
 <h:body>
        Thank you for your order:
        <br/>
        Name: <h:outputText value="#{confirmationbean.order.name}"/><br />
        Adress: <h:outputText value="#{confirmationbean.order.address}"/><br />
        Creditcard: <h:outputText value="#{confirmation.order.creditcard}"/><br />
    </h:body>
```

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

Describe how we can relate the concept of **web container** 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 **web container** and the principles of SCI.