Student ID__________________Student Name____________________________________

CS545 Web Applications Architecture and Frameworks

DE Final Exam
July 6, 2013

PRIVATE AND CONFIDENTIAL

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

5 blank pages are provided for writing the solutions and/or scratch paper. All 5 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 [15 points] {15 minutes}

a. **Describe the different facelets tags used in JSF and explain for every tag in one sentence what this tag means**

ui:insert – Used in template file, it defines content that is going to replace by the file that load the template. The content can be replace with “ui:define” tag.

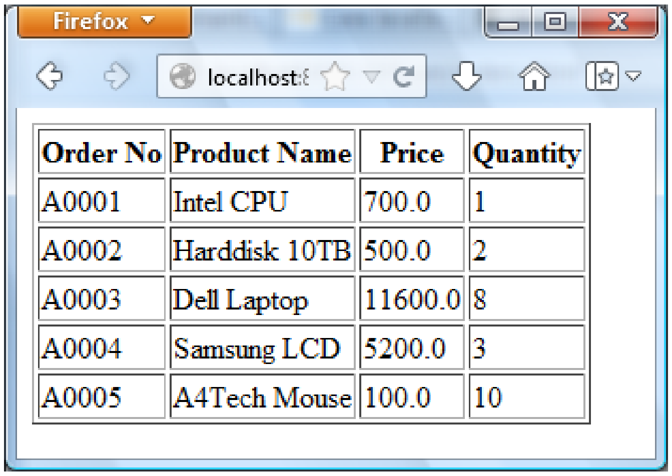
ui:define – Defines content that is inserted into template with a matching “ui:insert” tag.

ui:include – Similar to JSP’s “jsp:include”, includes content from another XHTML page.

ui:composition – If used with “template” attribute, the specified template is loaded, and the children of this tag defines the template layout; Otherwise, it’s a group of elements, that can be inserted somewhere.

Question 2 [20 points] {20 minutes}

We need to write an JSF application that shows a list of orders:



Order No	Product Name	Price	Quantity
A0001	Intel CPU	700.0	1
A0002	Harddisk 10TB	500.0	2
A0003	Dell Laptop	11600.0	8
A0004	Samsung LCD	5200.0	3
A0005	A4Tech Mouse	100.0	10

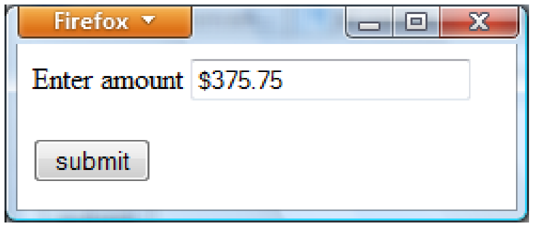
Complete the partial given code. **Do NOT write getter and setter methods!**

```
public class Order {
  private String orderNo;
  private String productName;
  private double price;
  private int qty;

  public Order(String orderNo, String productName,
       double price, int qty) {
    this.orderNo = orderNo;
    this.productName = productName;
    this.price = price;
    this.qty = qty;
  }
  //getter and setter methods are omitted
}

@ManagedBean
@RequestScoped
public class OrderBean {
 private List<Order> orderlist= new ArrayList<Order>();

  public OrderBean() {
    orderlist.add(new Order("A0001", "Intel CPU",700.00, 1));
    orderlist.add(new Order("A0002", "Harddisk 10TB",500.00, 2));
    orderlist.add(new Order("A0003", "Dell Laptop",11600.00, 8));
    orderlist.add(new Order("A0004", "Samsung LCD",5200.00, 3));
    orderlist.add(new Order("A0005", "A4Tech Mouse",100.00, 10));
  }
  //getter and setter methods are omitted
}
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core">
  <h:head>
    <title>Orders</title>
  </h:head>
  <h:body>
    <h:dataTable value="#{orderBean.orderlist}" var="o" border="1">
                        <h:column>
                                <f:facet name="header">Order No</f:facet>
                                <h:outputText value="#{o.orderNo}"/>
                        </h:column>
                        <h:column>
                                <f:facet name="header">Product Name</f:facet>
                                <h:outputText value="#{o.productName}"/>
                        </h:column>
                        <h:column>
                                <f:facet name="header">Price</f:facet>
                                <h:outputText value="#{o.price}"/>
                        </h:column>
                        <h:column>
                                <f:facet name="header">Quantity</f:facet>
                                <h:outputText value="#{o.qty}"/>
                        </h:column>
    </h:dataTable>
  </h:body>
</html>
```

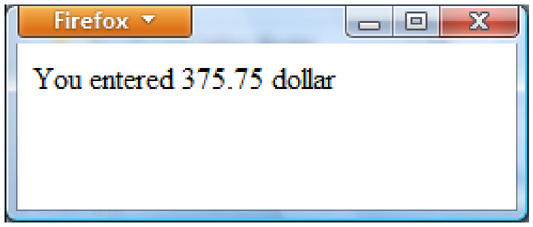
Question 3 [20 points] {25 minutes}

We have to write the following application in JSF:



Enter amount $375.75

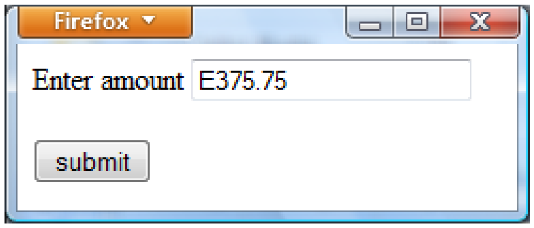
You can enter an amount and the first character should be the currency. If you enter $375.75 and click the submit button, you should see the following page:



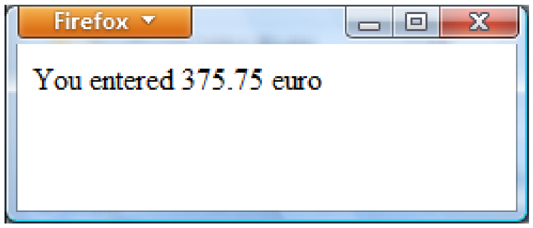
You entered 375.75 dollar

To make this work you need to use a custom converter. This converter should convert the string "$375,75" to the following class:

```
public class Amount {
   private double value;
   private String currency;
   //getter and setter methods are omitted
}
```



Enter amount E375.75



You entered 375.75 euro

If you enter E375,75 then the result page should show 375,75 euro.The application only support dollar amounts (with the character '$') and euro mounts (with the character 'E').

For this exercise you do not need to worry about validation, you can expect that the user always enters a correct string where the first character is either a '$' or a 'E' followed by a certain value.

You can use the following logic to get the particular parts from the entered string:

```
currencysymbol = enteredstring.substring(0, 1);
 stringamountvalue = enteredstring.substring(1, value.length());
 amountvalue = Double.parseDouble(stringamountvalue);
```

Complete the partial given code such that the application works with the given behavior.
IMPORTANT: do not write getter and setter methods!

```
@ManagedBean
@RequestScoped
public class amountbean {

   private Amount amount;

   public String submit(){
      return "showAmount";
   }

   //getter and setter methods are omitted
}
```

```
@FacesConverter(value = "amountConverter")
public class AmountConverter implements Converter {

  public Object getAsObject(FacesContext context, UIComponent component, String value) {
    if ("".equals(value)) {
      return null;
    }

    String currencysymbol = value.substring(0, 1);
    String stringamountvalue = value.substring(1, value.length());
    double amountvalue = Double.parseDouble(stringamountvalue);
    Amount amount = new Amount(currencysymbol, amountvalue);
    return amount;
  }

  public String getAsString(FacesContext context, UIComponent component, Object value) {
    Amount amount = (Amount) value;
    String returnString = amount.getValue() + " ";

    if (amount.getCurrency().equals("$")) {
      returnString += "dollar";
    } else {
      returnString += "euro";
    }
    return returnString;
  }
}
```

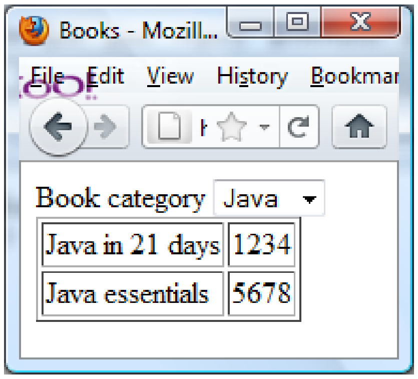
enterAmount.xhtml:

```
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://java.sun.com/jsf/html"
     xmlns:f="http://java.sun.com/jsf/core">
   <h:head>
      <title>Enter amount</title>
   </h:head>
   <h:body>
       <h:form id="form">
           Enter amount  <h:inputText  value="#{amountbean.amount}">
            <f:converter converterId="amountConverter"/>
           </h:inputText>
            <br/>
       <br/>
       <h:commandButton value="submit" action="#{amountbean.submit}" />
       </h:form>
   </h:body>
</html>
```

showAmount.xhtml:

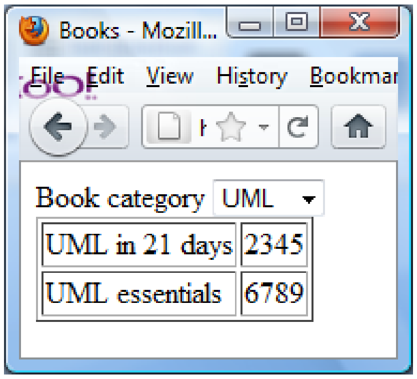
```
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://java.sun.com/jsf/html"
     xmlns:f="http://java.sun.com/jsf/core">
   <h:head>
      <title>Show amount</title>
   </h:head>
   <h:body>
      You entered
      <h:outputText value="#{amountbean.amount}">
         <f:converter converterId="amountConverter"/>
      </h:outputText>
   </h:body>
</html>
```

Question 4 Events [20 points] {25 minutes}



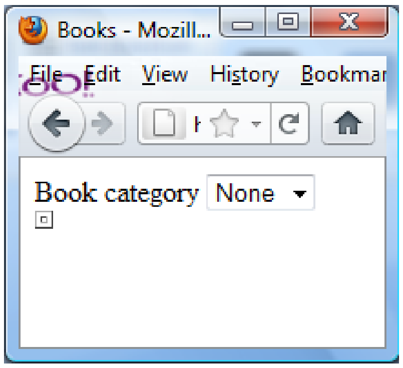
Book category Java

Java in 21 days	1234
Java essentials	5678



Book category UML

UML in 21 days	2345
UML essentials	6789



Book category None

Write an JSF application with the following behavior:
The application shows a **selectOneMenu** control with the values *Java, UML* and *None*.
If you select Java, then you see a table showing 2 Java books.
If you select UML, then you see a table showing 2 UML books.
If you select None, then the table is empty.
The first column in the table is the name of the book, and the second column is the ISBN number (in this example just a string containing 4 characters)

Complete the partial given code such that the application works with the given behavior. You only have to complete the code for the xhtml file and the managed bean.
IMPORTANT: do not write getter and setter methods!

books.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
    <h:head>
        <title>Books</title>
    </h:head>
    <h:body>
        <h:form>
    Book category
    <h:selectOneMenu value="#{bookbean.selectedBook}"
                    valueChangeListener="#{bookbean.changeBook}"
                    onchange="submit()">
        <f:selectItems value="#{bookbean.books}"/>
    </h:selectOneMenu>
    <br />
    <h:dataTable value="#{bookbean.booklist}" var="book" border="1" >
        <h:column>
          <h:outputText value="#{book.name}"/>
        </h:column>
        <h:column>
          <h:outputText value="#{book.isbn}"/>
        </h:column>
        </h:dataTable>
        </h:form>
    </h:body>
```

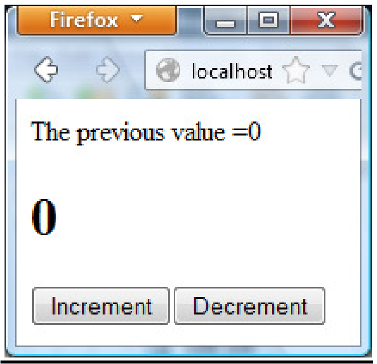
```
@ManagedBean
@RequestScoped
public class Bookbean {


    private String selectedBook;
    private Collection<SelectItem> books = new ArrayList();
    private Collection<Book> booklist= new ArrayList<Book>();

    public Bookbean(){
        books.add(new SelectItem("None","None"));
        books.add(new SelectItem("Java","Java"));
        books.add(new SelectItem("UML","UML"));
    }

    public void changeBook(ValueChangeEvent valueChangeEvent) {
        if (valueChangeEvent.getNewValue().toString().equals("Java")){
            booklist.add(new Book("Java in 21 days", "1234"));
            booklist.add(new Book("Java essentials", "5678"));
        } else if (valueChangeEvent.getNewValue().toString().equals("UML")){
            booklist.add(new Book("UML in 21 days", "2345"));
            booklist.add(new Book("UML essentials", "6789"));
        } else {
            booklist.clear();
        }
    }




}
```

Question 5 [20 points] {25 minutes}

Suppose we have the following simple business logic:

```
public class Counter {

   private int count = 0;

   public void increment() {
      count = count + 1;
   }
   public void decrement() {
      count = count - 1;
   }
   public int getCount() {
      return count;
   }
}
```

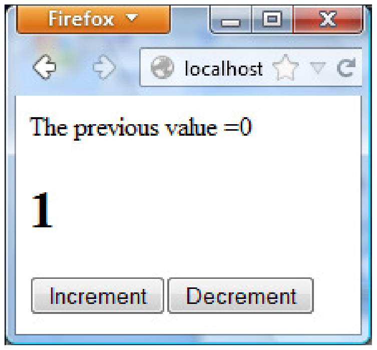
Now we want to write a JSF application using the given Counter class so we can access this Counter through a webpage. The JSF application works as follows



The previous value =0

0

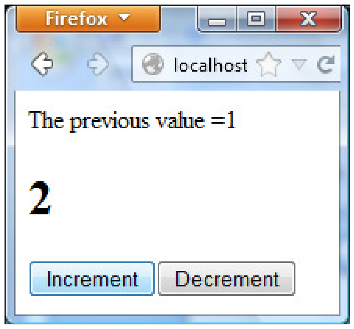
When you call the webpage for the first time, the counter value is 0 shown is a large bold font. The page also shows the previous value of the counter value which is initially 0.



The previous value =0

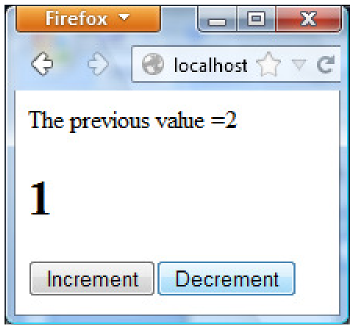
1

When you click the Increment button, the counter value is incremented, and the previous value is 0.


The previous value =1

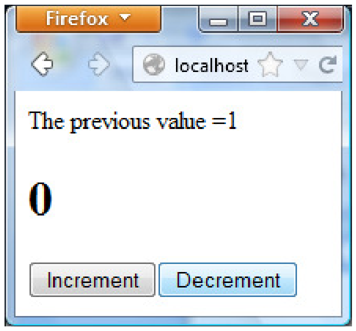
2

When you click the Increment button again, the counter value is incremented to 2, and the previous value is 1.


The previous value =2

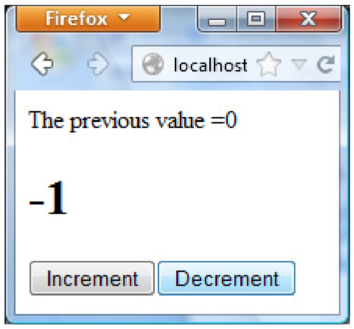
1

When you click the Decrement button, the counter value is decremented to 1, and the previous value is 2.


The previous value =1

0

When you click the Decrement button again, the counter value is decremented to 0, and the previous value is 1.


The previous value =0

-1

When you click the Decrement button again, the counter value is decremented to -1, and the previous value is 0.

Complete the partial given code such that the application works with the given behavior.
IMPORTANT: do not write getter and setter methods!

```
@ManagedBean
@SessionScoped
public class Counter {

   private int count = 0;

   public void increment() {
      count = count + 1;
   }

   public void decrement() {
      count = count - 1;
   }

   public int getCount() {
      return count;
   }
}

@ManagedBean
@RequestScoped
public class CounterBean {
   private String message="The previous value =";
   private int previousvalue=0;

   @ManagedProperty(value="#{counter}")
   private Counter counter;

   public void increment() {
      previousvalue=counter.getCount();
      counter.increment();
   }

   public void decrement() {
      previousvalue=counter.getCount();
      counter.decrement();
   }


}
```

```
<?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>Counter</title>
    </h:head>
    <h:body>
        <h:outputText value="#{counterBean.message}" />
        <h:outputText value="#{counterBean.previousvalue}" />
        <br/>
        <h1><h:outputText value="#{counterBean.counter.count}" /></h1>
        <h:form>
            <h:commandButton value="Increment"
                             action="#{counterBean.increment}"/>
            <h:commandButton value="Decrement"
                             action="#{counterBean.decrement}"/>
        </h:form>
    </h:body>
</html>
```

Question 6 SCI [5 points] {10 minutes}
Describe how we can relate JSF event handling 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 JSF event handling and the principles of SCI.