

Student ID \_\_\_\_\_ Student Name \_\_\_\_\_

**CS545 Web Applications Architecture and Frameworks  
DE Final Exam  
June 3, 2017**

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.

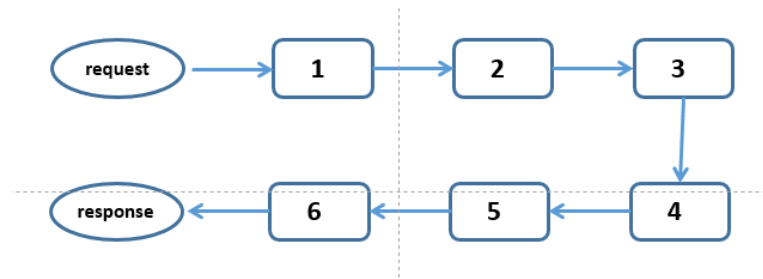
**3 blank** pages are provided for writing the solutions and/or scratch paper. All **3 pages** must be handed in with the exam

**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 [ 15 points ] {15 minutes}**

The JSF lifecycle consists of 6 phases:



- a. Give the name of every phase and in one sentence describe what JSF does in this phase.

Phase name	Short description of what JSF does in this phase
<b>1. Restore view</b>	<b>Create/retrieve component tree</b>
<b>2. Apply request</b>	<b>The values of the components in the component tree are updated with the values from the request</b>
<b>3. Process validations</b>	<b>Validate the values in the component tree</b>
<b>4. Update model values</b>	<b>Find the backingbean and update the values in the backing bean</b>
<b>5. Invoke application</b>	<b>Call the registered action listeners</b>
<b>6. Render response</b>	<b>Create the Http response</b>

- b. JSF splits the JSF lifecycle into 2 parts: execute and render

Give the number(s) of the phases that belong to the execute part

1, 2, 3, 4, 5

Give the number(s) of the phases that belong to the render part

6

## Question 2 [ 40 points ] {45 minutes}

Write a JSF application with the following behavior:



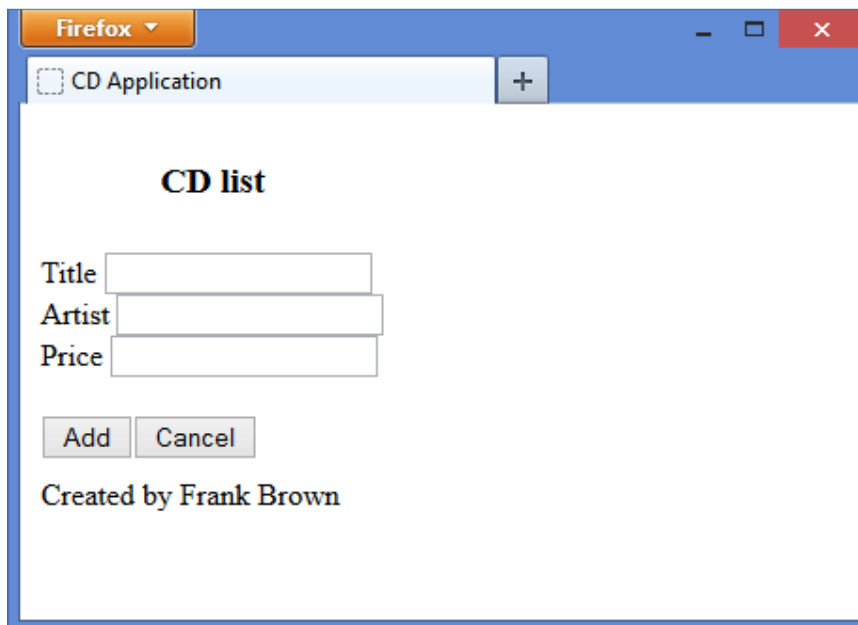
The screenshot shows a web browser window titled "CD Application" with a Firefox icon. The page has a header "CD list" and a table with four rows of CD information. Each row has a "Remove" button. Below the table is an "Add CD" button and a footer "Created by Frank Brown".

Title	Artist	Price	Action
The winner takes it all	ABBA	12.95	Remove
Staying Alive	Bee Gees	11.95	Remove
Blue Hawaii	Elvis Presley	12.05	Remove
Yellow Submarine	The Beatles	9.95	Remove

Created by Frank Brown

The application shows a list of CD's in a table. If you click the Remove button, that particular CD will be removed from the list.

If you click the Add CD button, then this page will be shown:



The screenshot shows a web browser window titled "CD Application" with a Firefox icon. The page has a header "CD list" and a form with three input fields labeled "Title", "Artist", and "Price". Below the form are "Add" and "Cancel" buttons. The footer is "Created by Frank Brown".

**CD list**

Title

Artist

Price

Created by Frank Brown

When you enter the CD information, and you click the Add button, then you navigate to the previous CD table page, and you see that this new CD is added to the list of CD's.

When you click the Cancel button you navigate to the previous CD table page.

The application uses a standard header and footer which is the same for all pages of the application. You have to use facelets to implement a standard header and footer for all pages.

Complete the partial given code such that the application works with the given behavior. You have to implement all java code, required annotation and JSF tags that are missing. This JSF application does not use a faces-config.xml file but uses annotation based configuration. Use the best practices we learned in this course. **IMPORTANT: do not write getter and setter methods!**

#### header.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition>
    <h3>CD list</h3>
  </ui:composition>

</html>
```

#### footer.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition>

    Created by Frank Brown
  </ui:composition>

</html>
```

#### cd.java

```
public class CD {
  private String title;
  private String artist;
  private String price;

  public CD(String title, String artist, String price) {
    this.title = title;
    this.artist = artist;
    this.price = price;
  }
  //getters and setters are not shown
}
```

## cdlistpage.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition template="template.xhtml">
    <ui:define name="header">
      <ui:include src="header.xhtml" />
    </ui:define>

    <ui:define name="content">
      <ui:include src="cdtable.xhtml" />
    </ui:define>

    <ui:define name="footer">
      <ui:include src="footer.xhtml" />
    </ui:define>
  </ui:composition>
</html>
```

## addcdpage.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition template="template.xhtml">
    <ui:define name="header">
      <ui:include src="header.xhtml" />
    </ui:define>

    <ui:define name="content">
      <ui:include src="addcdform.xhtml" />
    </ui:define>

    <ui:define name="footer">
      <ui:include src="footer.xhtml" />
    </ui:define>
  </ui:composition>

</html>
```

## cdtable.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition>

    <h:form>
      <h:dataTable value="#{cdtable.cdlist}" var="cd" border="1" >
        <h:column>
          <h:outputText value="#{cd.title}"/>
        </h:column>
        <h:column>
          <h:outputText value="#{cd.artist}"/>
        </h:column>
        <h:column>
          <h:outputText value="#{cd.price}"/>
        </h:column>
        <h:column>
          <h:commandButton value="Remove"
            action="#{cdtable.removeCD(cd)}/>
        </h:column>
      </h:dataTable>
      <h:commandButton value="Add CD"
        action="addcdpage"/>
    </h:form>
  </ui:composition>

</html>
```

## addcdform.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition>
    <h:form>
      Title
      <h:inputText value="#{cdtable.title}"/>
      <br />
      Artist
      <h:inputText value="#{cdtable.artist}"/>
      <br />
      Price
      <h:inputText value="#{cdtable.price}"/>
      <br />
      <br />
      <h:commandButton value="Add"
        action="#{cdtable.add}"/>
      <h:commandButton value="Cancel" type="submit"
        action="cdlistpage"/>
    </h:form>
  </ui:composition>
</html>
```



## template.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://xmlns.jcp.org/jsf/html"
  xmlns:f="http://xmlns.jcp.org/jsf/core"
  xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <h:head>
    <title>CD Application</title>
  </h:head>
  <h:body>
    <h:panelGrid columns="1" >
      <f:facet name="header">
        <ui:insert name="header" > Default Header </ui:insert>
      </f:facet>
      <ui:insert name="content" > Default Content </ui:insert>
      <f:facet name="footer">
        <ui:insert name="footer" > Default Footer </ui:insert>
      </f:facet>
    </h:panelGrid>
  </h:body>
</html>
```

## cdtable.java

```
@Named
@SessionScoped
public class Cdtable implements Serializable{
    private Collection<CD> cdlist = new ArrayList();
    private String title = "";
    private String artist = "";
    private String price = "";

    public Cdtable() {
        cdlist.add(new CD("The winner takes it all ", "ABBA", "12.95"));
        cdlist.add(new CD("Staying Alive ", "Bee Gees", "11.95"));
        cdlist.add(new CD("Blue Hawaii ", "Elvis Presley", "12.05"));
        cdlist.add(new CD("Yellow Submarine ", "The Beatles", "9.95"));
    }

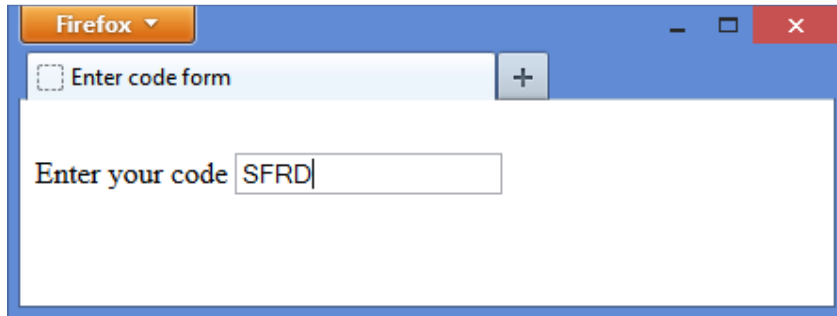
    public void removeCD(CD removeCd) {
        Iterator<CD> iter = cdlist.iterator();
        while (iter.hasNext()) {
            CD cd = iter.next();
            if (cd.getTitle().equals(removeCd.getTitle())) {
                iter.remove();
            }
        }
    }

    public String add() {
        CD newCD = new CD(title, artist, price);
        cdlist.add(newCD);
        return "cdlistpage";
    }

    //getters and setters are not shown
}
```

### Question 3 [ 20 points ] {20 minutes}

We have to write the following application in JSF:



The application shows a form that asks you to enter a code.

When you start typing in the inputtext, automatically all entered characters are shown as uppercase characters. So if you type a lowercase "s", the page will show an uppercase "S".

Use **AJAX** to implement this behavior.

A String in Java has the **toUpperCase()** method to transform all character to uppercase characters.

**Complete the partial given code such that the application works with the given behavior. You have to implement all java code, required annotation and JSF tags that are missing. This JSF application does not use a faces-config.xml file but uses annotation based configuration.**

**IMPORTANT: do not write getter and setter methods!**

entercode.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
  <h:head>
    <title>Enter code form</title>
  </h:head>
  <h:body>
    <h:form >
      <br/>
      Enter your code
      <h:inputText id="in" value="#{codebean.code}" >
        <f:ajax event="keyup" render="in"
              listener="#{codebean.update}" />
      </h:inputText>
      <br/>
    </h:form>
  </h:body>
</html>
```

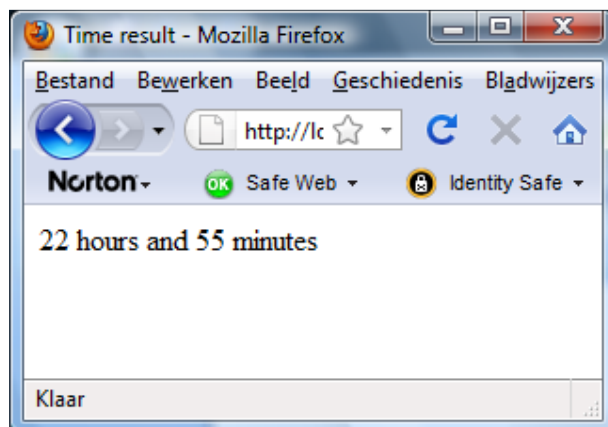
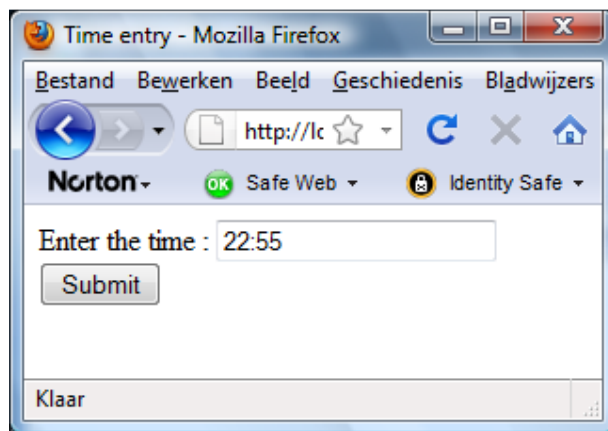
```
@Named
@RequestScoped
public class Codebean {
    private String code;

    public void update(AjaxBehaviorEvent event) {
        code=code.toUpperCase();
    }

    //getter and setters are not shown
}
```

#### Question 4 [ 20 points ] {30 minutes}

Write a JSF application with the following behavior:



The application consists of 2 pages, an entry page and a result page.

In the entry page you type in a string of the format **hh:mm** where hh are the hours and mm are the minutes. For example when you enter 22:55 and click the Submit button, the result page is shown with the text 22 hours and 55 minutes. And when you enter 12:15 and click the Submit button, the result page is shown with the text 12 hours and 15 minutes.

Write your own converter to implement this behavior.

**Complete the partial given code such that the application works with the given behavior. You have to implement ALL java code, required annotations and JSF tags that are missing.**

**IMPORTANT: do not write getter and setter methods!**

### entry.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>Time entry</title>
  </h:head>
  <h:body>
    <h:form>
      Enter the time :

      <h:inputText value="#{timeBean.time}">
        <f:converter converterId="timeConverter"/>
      </h:inputText>
      <br />
      <h:commandButton value="Submit"
        action="result.xhtml" />

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

### result.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>Time result</title>
  </h:head>
  <h:body>

    <h:outputText value="#{timeBean.time}">
      <f:converter converterId="timeConverter"/>
    </h:outputText>

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

#### TimeBean.java

**@ManagedBean**

**@RequestScoped**

**public class TimeBean {**

**MyTime time;**

**//getters and setters are not shown  
}**

#### MyTime.java

**public class MyTime {**

**private int hours;**

**private int minutes;**

**//getters and setters are not shown  
}**

#### TimeConverter.java

**@FacesConverter( value="timeConverter" )**

**public class TimeConverter implements Converter {**

**@Override**

**public Object getAsObject(FacesContext context, UIComponent component, String value) {**

**MyTime time = new MyTime();**

**time.setHours(Integer.parseInt(value.substring(0,value.indexOf(":"))));**

**time.setMinutes(Integer.parseInt(value.substring(value.indexOf(":")+1,value.length())));**

**return time;**

**}**

**@Override**

**public String getAsString(FacesContext context, UIComponent component, Object value) {**

**MyTime time = (MyTime)value;**

**return time.getHours()+" hours and "+time.getMinutes()+" minutes";**

**}**

**}**

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

Describe how we can relate **JSF custom components** 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 custom components** and the principles of SCI.