



HØGSKOLEN I SØR-TRØNDELAG

Avdeling for informatikk og e-læring -

AITeL

Kandidatnr:	
Eksamensdato:	3. desember 2009
Varighet:	0900-1100
Emnekode:	LO348D/LN349D
Emnenavn:	Web-applikasjoner med JSP og JSF
Klasse(r):	2 ING, nettstudenter
Studiepoeng:	
Faglærer(e):	Else Lervik, Grethe Sandstrak
Kontaktperson (adm.):	
Hjelpemidler:	Ingen
Oppgavesettet består av:	4 oppgaver og 6 sider (inkludert forside og vedlegg)
Vedlegg består av:	2 sider

Merknad: Oppgaveteksten kan beholdes av studenter som sitter eksamenstiden ut.

Lykke til!

Oppgave 1 (JSP og JSF) – vekt 25%

Et nettsted kan ha mange besøkende og utføre mange ulike interaksjoner med de ulike brukerne. Når du programmerer en nettside kan det være nødvendig å assosiere data med den enkelte besøkende på nettsiden. Beskriv kort ulike måter vi kan gjøre dette på. Gi også eksempel på typiske bruksområder for de ulike metodene du beskriver. (Se bort fra datafiler og databaser i denne oppgaven.)

Oppgave 2 (JSP) – vekt 20%

Vi har et vareregister og har laget en jsp-side for å registrere nye varer i registeret. Til å hjelpe oss har vi laget en Java-klasse – *Vare*, som ligger i pakken *EksDes2009*. Skjemaet med tilhørende kode ser slik ut:

<p>Registrer varer</p> <p>Varenavn: <input type="text"/></p> <p>Pris: <input type="text"/></p> <p><input type="button" value="Registrer ny vare"/></p>	<pre><html> <body> <h2>Registrer varer</h2> Varenavn: <input type="text" name="navn"/> Pris: <input type="text" name="pris"/> <input type="submit" name="regVare" value="Registrer ny vare"/> <% ArrayList<Vare> varene = new ArrayList<Vare>(); String navn = request.getParameter("navn"); double pris = Double.parseDouble(request.getParameter("pris")); Vare v = new Vare(navn, pris); varene.add(v); %> </body></html></pre>
---	---

Oppgave a)

Slik koden er laget nå, vil det opprettes et tomt vareregister hver gang nettsiden lastes. Hvilke(n) av teknikk(er) vil du bruke for at vareregisteret ikke skal nullstilles for hver gang, og slik at alle kunder som kommer på nettsiden får tilgang til det samme registeret? Se bort fra datafiler og databaser i denne oppgaven.

Oppgave b)

Dessverre så viser det seg at siden ikke virker som den skal – det kastes to unntak når koden kjøres:

“ArrayList cannot be resolved to a type”

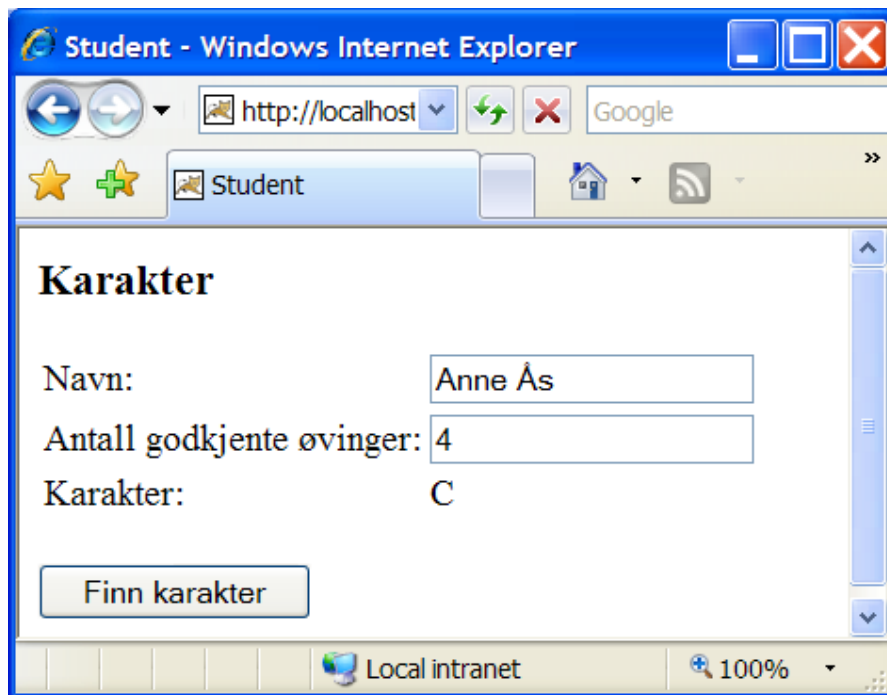
“Vare cannot be resolved to a type”

Hva kan være galt, og hvordan kan du løse dette problemet?

Oppgave c)

Du har løst problemet i oppgave b) og får fram skjemaet. Du fyller inn data om ny vare og trykker på knappen “Registrer ny vare”. Knappen fungerer ikke. Hvorfor?

Oppgave 3 (JSF) – vekt 35%



Dette er en meget primitiv applikasjon for beregning av karakter på det øvingsbaserte prosjektet.

Alle deloppgavene forholder seg til denne applikasjonen. Kildeteksten er gitt i vedlegg 1.

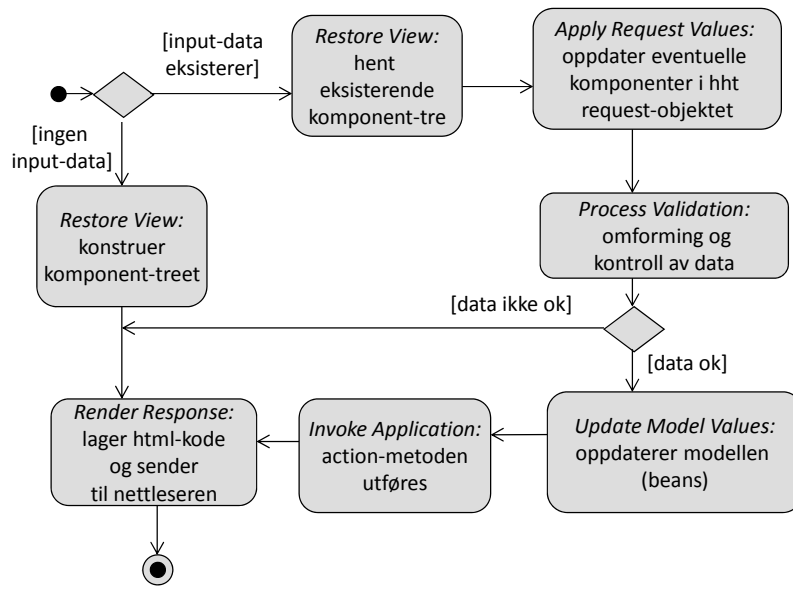
Oppgave a)

Forsøk på å kjøre applikasjonen gir følgende feilmelding:

```
javax.servlet.ServletException: /index.jsp(17,41) '#{student.karakter}'  
Property 'karakter' not found on type student.StudentBean  
javax.faces.webapp.FacesServlet.service(FacesServlet.java:277)
```

Forklar hvorfor denne feilmeldingen kommer. Hva må gjøres for at applikasjonen skal fungere?

I resten av denne oppgaven skal du, i tillegg til applikasjonen nevnt foran, forholde deg til ulike faser i livssyklusen til en JSF-applikasjon:



Oppgave b)

Fasen "Process Validation" består av to delprosesser: Omforming (converting) og kontroll av data (validation). Forklar helt konkret hva som skjer i de to delprosessene når applikasjonen foran kjører. Relater svaret til jsf-komponentene i index-filen i vedlegg 1.

Oppgave c)

Hva skjer i fasen "Update Model Values"? Relater svaret til koden i vedlegg 1.

Oppgave d)

Både på figuren over og i index-filen i vedlegg 1 refereres det til *action*-attributtet. Det er også et attributt som heter *actionListener*. Hva skiller disse attributtene fra hverandre? Når bør man bruke det ene, og når bør man bruke det andre? Hva er sammenhengen mellom dem?

Oppgave 4 (databaser og web-applikasjoner) – vekt 20%

I Java-koden kan vi gi tilgang til en databaseforbindelse enten ved å skrive

```
Connection forbindelse = DriverManager.getConnection(databasenavn);
```

eller

```
Connection forbindelse = ds.getConnection(); // ds er et object av klassen DataSource
```

- Hva skiller disse måtene å hente en databaseforbindelse på?
- Hvorfor bør vi bruke PreparedStatement-objekter i stedet for Statement-objekter når vi jobber med databaser i web-applikasjoner?

Vedlegg 1 – kildekode til applikasjon i oppgave 3

index.jsp

```
<!-- Oppgave, eksamen 2009, filnavn: index.jsp -->
<html>
  <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
  <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
  <f:view>
    <head>
      <title>Student</title>
    </head>
    <body>
      <h3>Karakter</h3>
      <p></p>
      <h:form id="eksamen">
        <h:panelGrid columns="2">
          Navn: <h:inputText value="#{student.navn}"/>
          Antall godkjente øvinger:<h:inputText id="antall" value="#{student.antGodkjent}">
            <f:validateLongRange minimum="0" maximum="5"/>
          </h:inputText>
          Karakter:<h:outputText value="#{student.karakter}"/>
        </h:panelGrid>
      <p></p>
      <h:message for="antall"/>
      <p></p>
      <h:commandButton value="Finn karakter" action="#{student.oppdaterKarakter}"/>
    </h:form>
  </body>
</f:view>
</html>
```

faces-config.xml

```
<?xml version="1.0"?>
<faces-config 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-facesconfig_1_2.xsd"
  version="1.2">
  <managed-bean>
    <managed-bean-name>student</managed-bean-name>
    <managed-bean-class>student.StudentBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
</faces-config>
```

StudentBean.java

```
package student;

public class StudentBean {
    private String navn = "";
    private int antGodkjent = 0;
    private char karakter = 'F';

    public StudentBean() {
    }

    public String getNavn() {
        return navn;
    }

    public void setNavn(String navn) {
        this.navn = navn;
    }

    public int getAntGodkjent() {
        return antGodkjent;
    }

    public void setAntGodkjent(int antGodkjent) {
        this.antGodkjent = antGodkjent;
    }

    public void oppdaterKarakter() {
        System.out.println("oppdaterKarakter");
        if (antGodkjent >= 4) {
            karakter = 'C';
        } else if (antGodkjent == 3) {
            karakter = 'D';
        } else if (antGodkjent == 2) {
            karakter = 'E';
        }
    }
}
```