

# Projektopgave 3-ugers 2014

## 02324 Videregående programmering

Projektnavn: **53\_final** Gruppe nr: **53** Afleveringsfrist: **fredag den 20/06 2014 Kl. 11:59**

Denne rapport er afleveret skriftligt og via CN (der skrives ikke under pga. elektronisk aflevering)

Denne rapport indeholder **29** sider incl. bilag og denne side

Studie nr, Efternavn, Fornavne

**s110795, Mortensen, Thomas Martin**

Kontakt person (Projektleder)



**s133979, Dickow, Jeppe Mads**



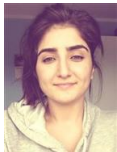
**s030876, Westh, Mikkel**



**s134001, Vorborg, Morten Guldhammer**



**s113786, Yapici, Melike**



**s123744, Wahidi, Lida**



**s130184, Frederiksen, Nikolaj Birck**



# Timeregnskab

Timeregnskab							
Deltagere	analyse	design	implementering	test	rapport/andet	I alt	
Jeppe Dickow	4	14	66	16	14	114	
Lida Wahidi	4	8	60	5	22	99	
Melike Yapici	5	13	56	4	19	97	
Mikkel Westh	6	18	92	16	10	142	
Morten Vorborg	5	12	64	17	14	112	
Nikolaj Frederiksen	6	15	62	16	16	115	
Thomas Mortensen	6	14	70	24	20	134	
I alt	36	94	470	98	115	813	

**Figur 0.1:** *Timeregnskab*

## Installationsvejledning

Her er installationsvejledningen til afvejningssystemet.

### Administrationsdelen

For at køre Web-interfacet til administration, er det nødvendigt at have en **Tomcat**-server på den server, der skal køre administrationsdelen. Samtidig skal der installeres en **MySQL** server til at indeholde data. Der er vedlagt et **MySQL** script til at oprette de tabeller der skal bruges Dette kan gøres på samme maskine, eller på en separat, efter behov. Installation af disse kan findes på <http://tomcat.apache.org> for **Tomcat**-server og <http://www.mysql.com> for en **MySQL** server. Herefter lægger man den medfølgende .war fil ind i undermappen *webapps* i **Tomcat** server mappen *apache-tomcat-x.x.xx*, for nemhedens skyld har vi kaldt vores .war fil for 53\_02324\_F14.war. x'erne definerer versionsnummer på **Tomcat** installationen.

### ASE


Afvejningsenheden (**ASE**), er let at gå til. Her skal man blot ligge jar-filen ASE.jar over på den computer man ønsker at køre den fra og herefter dobbeltklikke for at starte den, evt kan man også køre filen gennem kommando prompt, her er det muligt at give adressen på den vægt man gerne vil forbinde til som argument. Herefter gennemgår man sin afvejningsprocedure på vægten.

### Vægt simulator

Præcis som **ASE**'en kommer vægtsimulatoren som en jar-fil, vi har valgt at kalde den for Simulator.jar. Denne ligger man over på den ønskede computer og kører ved at dobbeltklikke.

Brugervejledningen til daglig brug findes i appendiks C på side 181

## Poster



# Afvejningssystem med Web-interface

- 3 ugers projekt af gruppe 53

$$f(x+\Delta x) = \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!} f^{(n)}(x)$$

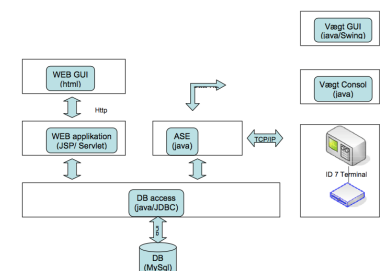
$$\int_a^b \Theta + \Omega \int_a^b \delta e^{\frac{i\pi}{2}} = \frac{2.7182818284}{x^2} \gg \frac{1}{x^2}$$

### Introduktion

Introduktion

Afvejningssystemet er blevet udviklet på baggrund af de udliveredede, kravspecifikationer. De overordnede krav har været, at systemet skal bruges til afvejning, samt dokumentation af råvarerafvejning og lagerstyring.

Det er i systemet muligt at definere råvarer i forskellige råvarerbatches. Derudover kan der oprettes recepter, som anvendes til at skabe produktbatches.



The diagram shows the system architecture. It includes a WEB GUI (HTML) connected to a WEB application (JSP/Servlet) via HTTP. The WEB application connects to an ASE (Java) component. The ASE connects to a DB access (Java/JDBC) layer, which in turn connects to a DB (MySQL). There are also components for Vægt GUI (Java/Swing), Vægt Console (Java), and ID 7 Terminal.

### Arkitektur


Systemet er sat sammen med forskellige delkomponenter:

- **Databasen**  
Databasen har oplysninger om operatører, råvarer, råvarerbatches, recepter, og planlagte producerede produktbatches.
- **Vægt simulator**  
En vægtsimulator til nemmere at kunne demonstrere vores program, uden at skulle have den rigtige vægt med.
- **Webinterface(GUI)**  
Webinterfacet, bliver brugt af farmaceuten, hvor de kan administrere recepter, råvarer og værkførereren kan oprette produktbatches, der er basis for dialog mellem vejeterminalen og databasen.
- **ASE'en**  
ASE (Afvejnings Styrings Enhed). Forbindelsen til vægten, der er bygget som en state machine.

### Systemets aktører

Der er 4 aktører, som er følgende:

- **Administrator**  
Superbruger, som har adgang til alle systemets rettigheder
- **Farmaceut**  
Varetager administrationen af råvarer og recepter
- **Værkfører**  
Varetager administrationen af råvarerbatches og produktbatches
- **Operatør**  
Tager sig af selve afvejningen.



The screenshot shows the 'Hovedmenu' (Main Menu) of the system. It has a dark blue background with a list of menu items in white text: Skift kodeord, Brugeradministration, Råvareadministration, Receptadministration, Råvarebatchadministration, Produktbatchadministration, and Log ud.

### Metoder

Til design af systemet, er der gjort brug af 3-lagsmodellens principper, hvor man arbejder med systemet i 3 mindre dele. Disse udgør: grænseflade, funktionalitet og data. Dette princip gør systemet overskueligt og nemt at vedligeholde.

Grænsefladen er udviklet vha. jsp, html og css, da det var et krav at systemet skulle laves med webinterface.

### Konklusion

- God integration fra tidligere delafleveringer.
- Design har udgangspunkt i trelagsmodellen.
- Operatøren har ansvar for afvejning (Use case 5).
- Alle krav mere eller mindre opfyldt.
- Flere funktioner kunne tilføjes. Mere tid.

.Danmarks Teknisk Universitet- 02324 videregående programmering

Figur 0.2: Poster til afvejningssystem

# Indhold

<b>1</b>	<b>Indledning</b>	<b>1</b>
<b>2</b>	<b>Projektplaner</b>	<b>1</b>
<b>3</b>	<b>Kravspecificering og analyse</b>	<b>1</b>
3.1	Kravspecificering . . . . .	1
3.2	Use Cases . . . . .	2
<b>4</b>	<b>Design</b>	<b>3</b>
4.1	ASE Design . . . . .	3
4.2	Administrationsdelen . . . . .	4
4.3	Sekvens Diagram . . . . .	5
4.4	Konsekvenser af desingvalg . . . . .	6
<b>5</b>	<b>Implementering</b>	<b>6</b>
5.1	ASE . . . . .	6
5.2	Administration . . . . .	7
5.3	Simulator . . . . .	10
5.4	Delkonklusion og oplagte forbedringer . . . . .	10
<b>6</b>	<b>Database</b>	<b>11</b>
<b>7</b>	<b>Test</b>	<b>13</b>
7.1	Generel teststrategi . . . . .	13
7.2	JUnit . . . . .	14
7.3	Indvirkning af teststrategi . . . . .	15
<b>8</b>	<b>Konklusion</b>	<b>16</b>
	Referencer . . . . .	17
<b>9</b>	<b>Anvendte værktøjer</b>	<b>17</b>
<b>A</b>	<b>Kode</b>	<b>18</b>
A.1	ase . . . . .	18
A.1.1	ase.boundary . . . . .	18
A.1.2	ase.controller . . . . .	22
A.2	admin . . . . .	28
A.2.1	admin.controller . . . . .	28
A.2.2	admin.data . . . . .	67
A.2.3	admin.test . . . . .	109
A.3	simulator . . . . .	117
A.3.1	simulator.boundary . . . . .	117
A.3.2	simulator.controller . . . . .	129
A.3.3	simulator.data . . . . .	132
A.3.4	simulator.test . . . . .	139
A.4	WebContent . . . . .	141
A.5	WEBINF . . . . .	174
<b>B</b>	<b>Arbejdsgang</b>	<b>179</b>
<b>C</b>	<b>Brugervejledning</b>	<b>181</b>
<b>D</b>	<b>Simulator</b>	<b>189</b>



## 1 Indledning

Formålet med denne rapport er at give et endeligt overblik over sammensætning og resultat af de tidligere delopgaver, som der er lavet tidligere i kurset 02324 Videregående programmering.

Projektet i 3-ugers perioden, går ud på at udvikle et afvejningssystem, der skal hjælpe en fiktiv medicinalvirksomhed til afvejning af råvarer, som bruges til forskellige recepter. Derudover skal systemet arkivere og lave dokumentation af de afvejninger, der bliver foretaget og systemet skal også kunne anvendes som lagerstyring af produkter og råvarer.

### Problemformulering

Laves ud fra opgaveoplægget

- Hvordan designer vi programmet?
- Hvad gør vi i forhold til implementering?
- Hvordan afgrænser vi eventuelt systemet?
- Hvilken teststrategi skal vi bruge?

## 2 Projektplaner

I det følgende beskrives, hvordan vi generelt har arbejdet med projektet.

### Udviklingsmetoder

Vores arbejdsgang og brug af udviklingsmetoder, har været fuldstændig magen til fremgangsmåden fra det første CDIO-projekt [Gru14a]. Derfor vil vi ikke beskrive det nærmere i denne rapport, da det blot vil blive en gentagelse af denne.

## 3 Kravspecificering og analyse

Her vil vi beskrive vores kravspecificering og hvad vi har lavet i analysefasen

### 3.1 Kravspecificering

I dette afsnit vil vi gennemgå det krav, der var opstillet til opgaven.

#### Funktionelle krav

Følgende punkter er funktionelle krav til opgaven, det er mere eller mindre de samme krav som fra cdio1.

- 4 aktører
  - Administrator
  - Farmaceut
  - Værkfører
  - Operatør
- Brugere skal logge ind med ID og password

- Systemet skal give den enkelte rettigheder afhængig af brugerens rolle
- Administrator skal kunne oprette, rette, slette og vise aktører til systemet
- Farmaceuten skal kunne oprette og vise recepter, råvarer i systemet
- Operatør skal kunne veje råvarer og indtaste produktbatches
- Testprogram for at kunne simulere vægt
- Webapplikationen skal være designet efter 3-lagsmodellen

### sletning af brugere

Det er givet som krav at man ikke skal kunne slette en bruger fra systemet når først man er oprettet, det krav har vi dog set på og vurderet at man kunne gøre det på en smartere måde, fordi hvis, f.eks. en bruger er blevet oprettet men aldrig udfører en afvejning, så fylder han bare plads i databasen, selvom det ikke bliver det største problem er det ligegyldigt at beholde en operatør i systemet, som principielt er ligegyldig, derfor har vi tilføjet en metode i vores kode der gør det muligt at fjerne en operatør, hvis han ikke har udført nogen afvejninger, eller deltaget i noget andet i systemet.

### afvejning af råvare

I vores afvejning af råvarer har vi valgt at betragte proceduren som en state machine, dvs. at vi går kun videre i states når kravene for det nuværende state som vi befinder os i er opfyldt. Derudover har vi valgt at man som operatør af vægten også er tilknyttet det batch id som man indtaster, fordi man bliver tildelt et produktbatch man skal afveje, det skal ikke være muligt for en anden operatør at afveje andres opgaver, det sikrer vi os i mod ved at checke om id'erne passer sammen.

### Ikke funktionelle krav

- Kodes i **JSP** og ved brug af **HttpServlet**
- **UML** artefakter efter eget valg bruges til dokumentation.

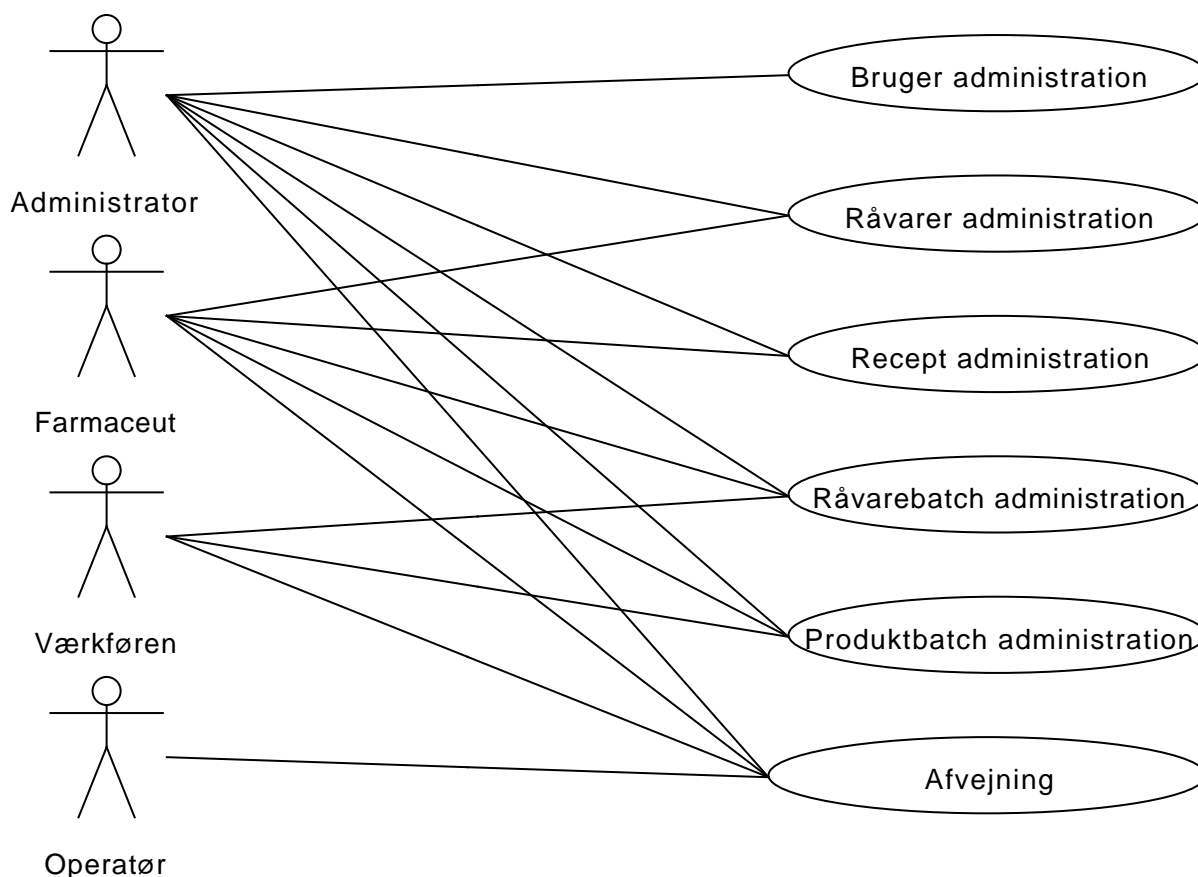
## 3.2 Use Cases

Vi har til opgaven fået udleveret 6 use-cases, som vi har brugt som udgangspunkt for systemet. For overblikkets skyld har vi lavet et use-case diagram, der viser disse i figur 3.1 på næste side. Vi har valgt at bibeholde navnene som de er udleveret. Det havde måske været nemmere for en udvikler, hvis vi havde kaldt de use-case, der administrerer noget for **CRUD**, da det ville være en pænere jargon i forhold til udviklerne. Vi har dog valgt at det skal være let for kunden. Vi ved desuden godt, at administration dækker over elementer i **CRUD**.

### Beslutninger

De valg vi har truffet, har vi primært truffet for at tilfredstille kunden. Det er i kundens interesse at der er sporbarhed i systemet. Det opnår vi i høj grad ved at tilknytte en operatør til en produktbatch, så andre operatører ikke kan afveje til denne. Det gør det sværere at lave fejl i denne del af systemet. At vi kan fjerne brugere fra systemet, var ikke en del af opgaven, men det er i kundens favør at der ikke flyder unyttige operatører rundt i systemet, som alligevel ikke har haft nogen indflydelse på det heller.





Figur 3.1: Use-case diagram, der viser de udleverede use-cases

## 4 Design

I dette kapitel vil vi belyse, hvordan vi har designet vores system på, hvad vi synes er, en hensigtsmæssig måde.

### 4.1 ASE Design

Vores ASE er opbygget som en state machine (se figur 4.1 på den følgende side). Dette giver et nemt og hurtigt overblik over selve Afvejnings systemet og hvordan det kommer til at fungere for operatøren, som skal lave selve afvejningen.

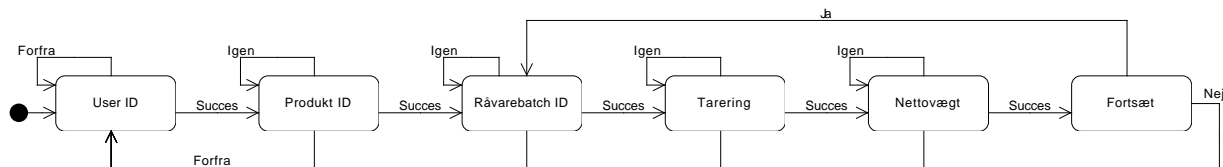
For overskuelighedens skyld har vi delt controlleren op i de trin der sker under en afvejning, altså de situationer som operatøren bliver sat i og skal agere ud fra.

I et perfekt forløb skal operatøren indtaste bruger id, indtaste produktbatch id på det produktbatch han vil afveje, indtaste råvarebatch id på det batch han har hentet den nødvendige råvare fra, kontrollere at vægten er tom og trykke "OK", sætte tarabeholder på vægten og trykke "OK", helde den korrekte mængde af råvaren i tarabeholderen og trykke "OK" og til sidst vælge om han vil veje flere komponenter af.

Ved alle undtagen det sidste trin er der en risiko for at der er et eller andet der ikke går som det skal. I disse tilfælde skal operatøren så vælge om han vil forsøge igen eller om det der er sket er så galt at han enten vil forlade vægten eller starte forfra. I det sidste trin har operatøren mulighed for

enten at forsætte med afvejningen ved at gå tilbage til indtastning af råvarebatch id eller om han vil starte helt forfra.

Hvis man vil forlade vægten på ordentlig vis således at den er klar til en anden operatør, gøres dette ved at hoppe tilbage til indtastning af user id hvorefter vægten er klar til en ny operatør. Dette ses i figur 4.1.

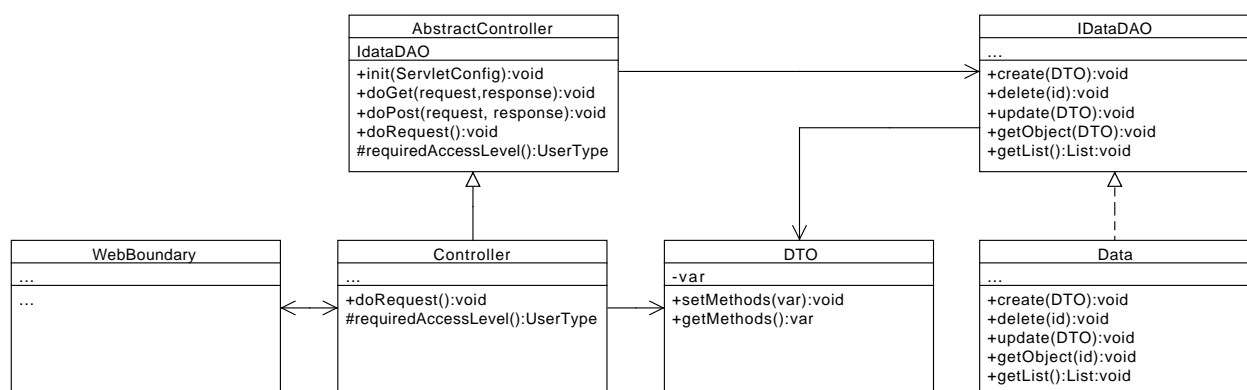


Figur 4.1: State machine over ASE

## 4.2 Administrationsdelen

Til at dokumentere vores administrationsdel, har vi valgt at lave et generelt klassediagram. Det beskriver den generelle opbygning af systemet. Hvis man kigger på figur 4.2, og sammenligner den med vores kode, vil man hurtigt kunne se at det ikke er nødvendigt at have alle klasser med i diagrammet. Princippet for vores opbygning ville bare give en masse klasser, der var stort set ens, men med forskellige navne.

Man vil hurtigt se at vores system inddelt efter 3-lags modellen [A07]. Det vil sige vi har et grænsefladelag, hvor vi har lagt alt hvad brugeren skal kunne se og aktivere. I dette tilfælde er det al vores JSP. Dette er alt sammen samlet i samme pakke. Her efter kan brugeren ikke længere se hvad der sker. Hermed tager vores funktionalitetslag over. Her kontrollerer vi det input vi har fået fra brugeren, og ser om vi eventuelt skal indsætte eller hente noget fra vores datalag, der udelukkende indeholder data. I vores tilfælde henter datalaget dog sine data fra en database vi har givet adgang til. Ved at opbygge administrationsdelen på denne måde, kommer vi også til at understøtte MVC mønstret [DTU14].



Figur 4.2: Det generelle klassediagram over administrationsdelen

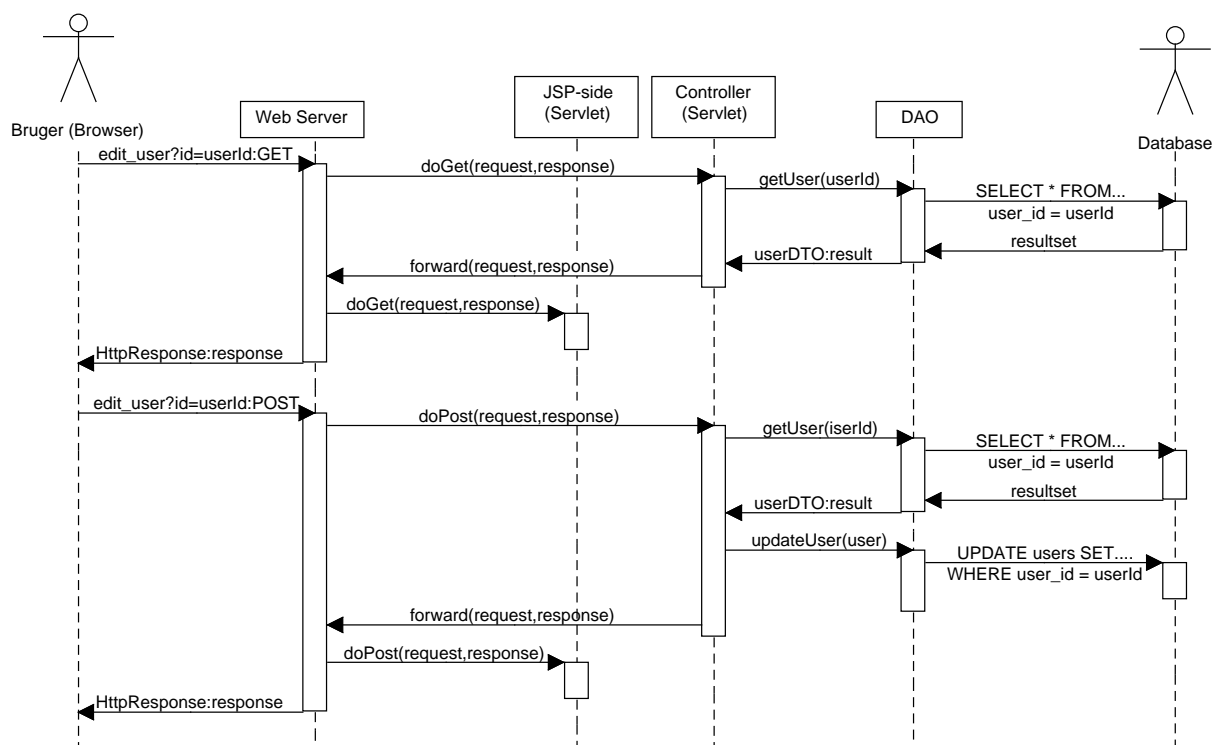
### 4.3 Sekvens Diagram

Vores Sekvens Diagram figur 4.3 illustrere hvordan et forløb, hvor en administrator skal ændre i oplysningerne om en bruger, kan se ud som konsekvens af opbygningen af vores system. Det ses hvordan administratoren til at starte med, beder om at se oplysningerne for den enkelte bruger. herefter bliver disse hentet frem fra databasen via datalaget og gemmes i request-objektet. Herefter forwarder vores controller servlet, via webserveren, til vores jsp-side, der så generer det endelige output, der så sender til administratoren.

Efter at administratoren har fået oplysningerne om brugeren, og har haft mulighed for at rette i dem, sender han så de rettede oplysninger tilbage til vores controller. Dette sker i praksis, ved at han trykker på submit-knappen på den formular, der er på siden. Dette resulterer i at alle oplysningerne i input-felterne bliver sendt med sammen med POST-requestet.

På samme måde som før hentes oplysningerne fra databasen. De nye oplysninger forsøges indsat i det `UserDTO`-objekt, vi har fået fra datalaget, og vi forsøger at gemme de ændringer administratoren har lavet. Dette gøres ved at kalde `updateUser(UserDTO user)` på vores `UserDAO`, der så laver disse oplysninger om til en SQL komando, der sendes til databasen. Oplysninger om hvad der er gemt, og om der var nogen fejl sendes så videre til jsp-siden, der igen generere et output til administratoren. Hvis der ikke var nogen fejl, vil dette output være en bekræftelse af, at ændringerne er blevet udført. Hvis der opstod en eller flere fejl i forbindelse med ændringerne, vil disse blive vist for administratoren.

Hvis der var fejl i oplysningerne vil administratoren få muligheden for at rette i oplysningerne igen således at der ikke er fejl i mere og så foretage et nyt POST-request, på samme måde som tidligere.



Figur 4.3: Sekvensdiagram over et ideelt forløb af redigering af brugeroplysninger

## 4.4 Konsekvenser af desingvalg

I forhold til administrationdelen får vi mange fordele ud af netop dette design. Ved at holde det i de forskellige lag er det let at rette i eksempelvis grænsefladelaget, da en ændring her, sjældent vil effektuere en ændring i andre lag. Sådan kan man gentage historien for funktionalitetslaget. Det vil også sjældent influere længere nede i systemet. Datalaget vil man nok sjældent ændre i. Skulle det ske en ændring er nødvendig her, ved man dog hvor man skal starte. Dog kan man argumentere for at en ændring i datalaget, også vil kræve ændringer i andre lag. Man skal jo kunne indsætte/trække disse nye data til de andre lag.

## 5 Implementering

### 5.1 ASE

Vi har valgt at implementere vores ASE som en state machine der giver os muligheden for at hoppe tilbage i programmet skulle operatøren vælge at trykke cancel, eller hvis vi har nogle værdier der ikke stemmer overens med databasen.

#### ASEBoundary

Vores ASEBoundary fungerer som den direkte forbindelse til vægten/vægtsimulatoren. Klassen har en client socket der forbinder direkte til vægten når den instantieres i ASEControlleren. Derudover har vi en lang række metoder til at kommunikere med vægten f.eks. ved hjælp af kommandoer som "RM20 8...", et eksempel på en RM20 8 besked kan ses i figur 5.1. Alt efter hvilken metode der bliver kaldt i ASEBoundary sender den så en bestemt kommando til vægten og lytter derefter på et svar. Når den modtager et svar, parser vi det enten ved hjælp af regular expressions eller ved blot at tage en sub string baseret på positionen af bestemte tegn, for således kun at få den info vi skal bruge til at sende tilbage til ASEControlleren. F.eks. hvis der bliver sendt "RM20..." kommando vil vægten svare tilbage med "RM20 B" og derefter "RM20 A "userinput"". Her er det vigtigt vi sørger for at fjerne alt på nær userinput.

Hvis forbindelsen til vægten ryger vil man få en Exception så snart man forsøger at sende noget igennem den socket vi bruger, hvilket sker hver gang man kalder en metode på boundaryet, dette har vi dog valgt at tage højde for i controlleren i stedet for at lave mere forkromet løsning i boundaryet.

```
102 public double getTara() throws IOException {
103     String sentence = "RM20 8 \"Placer beholder\" \"OK/Cancel\" \"\"";
104     socketOutput.writeBytes(sentence + "\r\n");
105     String input;
106     do
107         input = socketReader.readLine();
108     while (!input.startsWith("RM20") || input.startsWith("RM20 B") || input.startsWith("RM20 I"));
109     if (!input.startsWith("RM20 A"))
110         return -1.0;
```

Figur 5.1: RM20 8

#### ASEController

ASEController er, som nævnt tidligere, opbygget efter en state machine, det kan f.eks ses på figur 5.2 på næste side netop for at kunne hoppe tilbage i tilfælde af en fejl eller hvis operatøren trykker cancel. Den sørger så for at lave alle sammenligningerne imellem alt information der kommer fra databasen og hvad der kommer ind fra ASEBoundary'et. Efter alle afvejninger er klaret sørger ASEcontroller for at sende alt information til databasen. Da forbindelse ryger hvis man slukker for vægten, har vi valgt at sørge for at ASEControlleren forsøger at instantiere et ASEBoundary, så længe der ikke er forbindelse til vægten. Så lader vi garbage collectoren tage sig af alle de ASEBoundary-objekter vi ikke bruger. Det er ikke den kønneste løsning, men den var hurtig at lave og den virker.

```
52 public static void menu(ASEBoundary bound) throws IOException {  
53     while (!done) {  
54         reset = false; // Bliver sat til true, hvis vi ønsker at  
55                         // genstarte.  
56         step_done = false;  
57         while (!step_done) { // Step 3+4  
58             String userName;  
59  
60             userID = bound.getID();  
61             if (userID < 0)  
62                 continue;  
63             try {  
64                 IUserDAO users = new UserData();  
65                 userName = users.getUser(userID).getUsername();  
66                 if (bound.sendConfirm(userName)) {  
67                     step_done = true;  
68                 } else {  
69                     continue;  
70                 }  
71             }  
72         }  
73     }  
74 }
```

Figur 5.2: ASE Controller

## 5.2 Administration

Webdelen, der skal gøre det muligt for henholdsvis Administratoren, Farmaceuten og Værkføren at administrere deres forskellige funktioner i databasen, er opbygget omkring et menu-system bestående af en hovedmenu. Før man kan komme ind på hovedmenuen, skal der logges ind med id og kodeord. Ud fra id kan systemet se hvilken rolle man har og herefter kommer hovedmenuen frem med de muligheder, der passer til rollen. De enkelte Websider er lavet med http-servlets, HTML+JSP og med styling via CSS. Vi har valgt at holde designet så simpelt som muligt, i farverne lilla baggrund, hvide overskrifter, grå knapper, hvid på sort baggrund til labels og sort til diverse oplysninger omkring elementer og hvid på rød baggrund til fejlbeskedder. Alt design med hensyn til farve, format osv er lavet i .css-filen.

### Boundary

Alle vores boundaries er implementeret som JSP-sider, der henter data fra request context, der skal bruges i den aktuelle visning. Dette gøres ved hjælp af henholdsvis beans med scope="request" og metoden request.getAttribute() til de datatyper der ikke kan fungere som beans. Der er et minimum af logik disse JSP-sider, der er dog nogle forskellige ting vi er nød til at kunne tage højde for:

- Vores fejlbeskedder vises i en <div class="error"> der er beregnet til at have en særskilt ramme og baggrund, disse skal kun vises hvis de rent faktisk indeholder en fejlbesked.
- En del af vores boundaries har en "fuldført" status der er en særlig visning der bekræfter at en handling er udført korrekt, et element er blevet oprettet, redigeret eller slettet korrekt.
- En del visninger skal tage højde for en eller flere lister af varierende længde og tilpasse visningen herefter.

Vores JSP sider indeholder allesammen en formular med indtastningsfelter der submittes med et POST request til den tilhørende controller og/eller forskellige knapper (links der ligner knapper) der henviser til andre controllere eventuelt med et id parameter tilføjet i de tilfælde hvor der skal ændres ved specifikke elementer i databasen.

Vores JSP sider er lavet helt uden formatering men ved flittigt brug af forskellige css-klasser. Således er alt formatering er indeholdt i vores style.css.

## Controller

### AbstractController

Alle vores controllere nedarver fra vores `AbstractController`. Vores `AbstractController` er en abstrakt klasse der nedarver fra `HttpServlet` der sørger for følgende som alle vores controllere har brug for:

- Den har en `Init()`-metode der sørger for at holde referencer til alle vores data objekter og oprette instanser af dem, hvis de ikke er oprettet.
- Ved hjælp af en instans af `UserSession`-klassen, som vi gemmer i vores session, sørger den for at kontrollere om en bruger er logget ind, og hvis brugeren ikke er logget ind, sender den brugeren videre til vores loginside (hvis brugeren ikke allerede er på vej til loginsiden).
- Den har en abstrakt metode `requiredAccessLevel()` der "tvinger" alle controllere der implementere `AbstractController` til at fortælle hvilken `UserType` det kræver for at få lov til at se siden.
- Ved hjælp af `requiredAccessLevel()` kontrollerer den om den bruger, der er logget ind, har rettigheder til at se siden og hvis brugeren ikke har rettigheder til at se siden, sendes der en fejl tilbage der fortæller at brugeren ikke er autoriseret. (Det burde ikke ske at en bruger kommer ind på en side vedkomne ikke har adgang til, ved normal brug af web-interfacet. Eftersom man selv kan indtaste en adresse på en side man ikke har adgang til, har vi lavet denne sikkerhedsforanstaltning).
- Ved kald til enten `doPost` eller `doGet` kontrolleres de ting der kontrolleres hvorefter den abstrakte metode `doRequest` kaldes, således at den enkelte controller kun skal tage hensyn til det der er specifikt for den. `doPost` og `doGet` er begge final, således at man ikke kan komme til at springe noget over ved at implementere disse.

### Controller generelt

Alle vores controllere er implementeret som en servlet, tilknyttet hver deres adresse i `web.xml`, således at vi både nemt kan linke mellem de forskellige opgaver og en bruger f.eks. kan gemme et bogmærke til en bestemt opgave, hvis de er interesserede. Alle controllere nedarver fra `AbstractController` hvilket betyder at alt hvad den enkelte controller fortager sig, ligger i metoden `doRequest()`. Derudover vil metoden `requiredAccessLevel()` fortælle hvilken brugertype der har rettigheder til at se den side der hører til controlleren. f.eks. returnerer `requiredAccessLevel()` i vores brugeradministrationscontroller `UserType.ADMIN` hvorimod den samme metode i vores controller til hovedmenuen returnerer `null`.

Selvom der er meget der er det samme i vores controller uanset om det er et POST eller et GET request der bliver sendt er der også én væsentlig forskel. Hvis vores controller modtager et GET request, betyder det at der ikke er blevet indtastet nogen oplysninger og vi derfor bare kun skal hente nogle data frem og vise dem, men ikke forholde os til bruger input. Hvis der derimod er tale om et POST request, skal vi først kontrollere om de indtastede oplysninger er gyldige og overholder de regler der er sat for dem. Hvis oplysningerne er gyldige osv. skal vi (alt efter controller) fortage nogle ændringer i databasen via vores datalag. Hvis oplysningerne ikke er korrekte eller hvis der skete en fejl da vi forsøgte at ændre i databasen, skal der genereres nogle passende fejlbeskeder der skal vises til brugeren.

Når vi har hentet hvad vi skal bruge fra databasen og behandlet det eventuelle input, sørger controlleren for at gemme alle de nødvendige oplysninger i vores `requestcontext` og så forwarde til den JSP-side der hører til controlleren. På den måde skal JSP-siden kun forholde sig til at vise de oplysninger der kommer ud af et request på en fornuftig måde, mens al logikken ligger i controlleren. Vi har kaldt alle vores JSP-sider "`<adresse på controller>_boundary.jsp`".

## Data

Her gennemgår vi alle interessante elementer i vores datalag. Vores datalag er det der sørger for at vores **Java** program kan snakke sammen med vores **MySQL** database.

## Commodity

I `commodityBatchData` og i `commodityData` gør vi brug af standard **MySQL** kald til at forbinde til vores database med, de fleste metoder tager imod et argument eller flere for at returnere det ønskede. F.eks. er det muligt at få en liste over alle `Commodities` i vores database, men det er også muligt at få en specific `commodity`. For at slette en `commodityBatch` er det dog nødvendigt at den ikke har været brugt i nogen afvejninger, det sikrer vi os ved at lave et **MySQL** kald der spørger efter samtlige `commodityBatches` som indgår i et produktbatch eller flere, hvis det giver et tomt resultat, kan vi efterfølgende slette den, da vi nu er sikre på at den ikke er blevet brugt.

## Connector

Vores `Connector` klasse er lavet for at gøre koden mere overskuelig fordi vi åbner en forbindelse til databasen i alle vores metoder, gav det mere mening at samle denne forbindelse i en separat klasse, den har vi så gjort mere eller mindre statisk så når først forbindelsen er åben er den gemt, og så kan vi lukke den ved et enkelt kald og på den måde spare på ressourcer, i stedet for at holde en forbindelse til vores **MySQL** database hele tiden. Vi har gemt alle informationerne om vores database i en separat `Constant` klasse så det er let at ændre database hvis man får brug for det på et tidspunkt.

## Prescription

I vores `Prescription` klasser er det muligt at oprette `prescriptions` og det er muligt at slette dem igen, en vigtig detalje omkring sletning er dog at vi ikke har mulighed for at slette en `prescription`, hvis den har nogen `prescriptionComponents` tilknyttet. Det sikrer at vi ikke lige pludselig har en `prescriptionComponents` som er ugyldige i vores database. det kan ses i figur 5.3

```
98      try {
99          Connector
100              .doQuery("SELECT * FROM prescription WHERE prescription_id IN "
101                      + "(SELECT prescription_id from prescriptioncomponent);");
102      } catch (DALException e) {
103          Connector
104              .doUpdate("DELETE FROM prescription WHERE prescription_id = "
105                      + id + ";");
106          Connector.closeConnection();
107      }
```

Figur 5.3: *prescriptionCode*.

Ellers er det muligt ligesom i de fleste andre klasser, at få en liste over alle `Prescriptions` hentet ud fra databasen, eller bestemte `Prescriptions`. I `prescriptionComponents` er det muligt at slette samtlige `prescriptionComponents` som er tilknyttet et særligt `PrescriptionId`, men hvis man kun ønsker at slette en enkelt `prescriptionComponent` er det også muligt at gøre dette, det kræver blot at man også oplyser et `commodityId` på den måde er man sikker på det er den rigtige `prescriptionComponent` man får fjernet fra databasen, fordi de to tilsammen udgør nøglen i databasen, og der kan dermed ikke være flere af dem. Man kan få en liste over alle `prescriptionComponents` hvis man oplyser et specifikt `prescriptionId` på den måde får man kun alle dem som tilhører netop det id.

## ProductBatch

I vores `productbatchComponents` er det muligt at hente en liste ud over alle `productbatchcomponents`, det er også muligt at slette samtlige `productbatchcomponents`, fordi det giver ikke mening at kunne fjerne enkelte komponenter, da de tilhører et samlet produkt som er lavet ud fra vores `prescriptions`, derfor hvis man slettet noget skal det hele fjernes, hvis man ønsker at ændre i en enkelt `productbatch-`

component skal man benytte sig af update metoden, på den måde kan man ændre i en eksisterende komponent. Det er også muligt at få fat i productbatchcomponents via et natural join mellem commoditybatch og productbatchcomponents, se figur 5.4.

```

52      ResultSet rs = Connector
53          .doQuery("SELECT * FROM productbatchcomponent NATURAL JOIN commoditybatch WHERE pb_id = "
54                  + pb_id + " AND commodity_id = "
55                  + commodity_id + ";");
56      Connector.closeConnection();
57      try {
58          if (!rs.first()) {
59              throw new DALException(
60                  "Produkt batchen med det givne produkt batch id = "
61                  + pb_id
62                  + " og det tilsvarende råvare batch id = "
63                  + commodity_id
64                  + " eksisterer ikke i databasen");
65          }
66          return new ProductBatchCompDTO(rs.getInt("pb_id"),
67                                          rs.getInt("commoditybatch_id"), rs.getInt("user_id"),
68                                          rs.getDouble("tara"), rs.getDouble("netto"));

```

Figur 5.4: produktbatch join

I ProductBatchData er det kun muligt at slette en productbatch hvis der ikke hører nogen afvejninger til den, dvs. at den ikke er påbegyndt endnu, og så er det muligt at fjerne den igen.

## UserData

Der er ikke ændret noget i de metoder som vores UserData understøtter siden [Gru14a], forskellen er at den nu er implementeret op imod en database, og at alle metode kaldene derfor benytter sig af **MySQL**. Bruger databasen er dog blevet ændret en smule da den nu skulle kunne understøtte 4 typer af brugere, samt om brugeren er inaktiv. Det er vi kommet udenom ved at bruge en INT som går fra 0 til 4, på den måde kan vi let aflæse hvilken rolle en bruger har, når vi henter ham/hende ud af databasen.

## 5.3 Simulator

Vægtsimulatoren er mere eller mindre genbrugt fra [Gru14b]. Vi har dog ændret vores spinner fra den opgave til at være endnu et JTextPane. Dette er sket fordi vi fandt ud af, måden hvorpå vi havde implementeret vores spinner gav en fejl, hvis man ændrede på vægten rigtig mange gange.

Herudover har vi tilføjet funktionalitet, så vi kan modtage \RM30 kommandoer. Dette er sket fordi vi fandt ud af en måde hvorpå vi kunne lave "soft"buttons på den rigtige vægt igennem vores **ASE**. Derfor har vi også implementeret logikken til at kunne modtage disse kommandoer i simulatoren. Forskellen er dog at vi ikke har "soft"buttons, der kan kaldes frem efter behov, da vi i forvejen havde en hardbutton til "OK". Derfor tilføjede vi i stedet en "hard"button til "cancel".

Resten af koden til hele systemet findes i bilag appendiks A på side 18.

## 5.4 Delkonklusion og oplagte forbedringer

Vores endelige system opfylder på nær følgende punkter alle krav i kravsspecifikationen:

- I stedet for at forhindre sletning af enhver bruger af typen "operatør", har vi i stedet valgt at forhindre sletning af enhver bruger der har foretaget en afvejning, altså enhver bruger hvis id optræder i et produktbatchkomponent kan hverken slettes eller få ændret sit bruger id.
- I usecase 5 står der: "Når værkføren har oprettet en ny produktbatch udprintes denne og uddeles til en udvalgt operatør. Operatøren har herefter ansvaret for produktionen disse.(se bilag 5 og 6)". Dette læser vi som at ethvert produktbatch har en operatør der står for at skulle lave afvejningerne. På baggrund af dette har vi valgt at tilføje et id til protuktbatches, således at det kun



er den operatør der har ansvaret for et produktbatch, der kan lave afvejningerne på det pågældende batch. På den måde kan en operatør ikke komme til at lave afvejninger på et produktbatch som en anden har ansvaret for. Værkføren kan altid ændre dette id, så længe produktbatchet ikke er under produktion eller er afluttet.

- På grund af den begrænsede tid har vi primært fokuseret på at implementere et så komplet system som muligt der er funktionelt. Vi har kun i begrænset omfang sat ressourcer af til at teste programmet ud over den funktionstest der er forgået i forbindelse med afprøvning af at de forskellige dele af programmet har fungeret sammen. Hvilket også betyder at vi vores GUI kun tager højde for de mest typiske fejl der kan opstå i forhold til bruger input, men ikke nødvendigvis giver en særlig brugbar fejlbesked i tilfælde hvor nogle ændringer brugeren forsøger at foretage ikke kan lade sig gøre på grund af bestemte forhold i databasen, f.eks. hvis han/hun forsøger at slette en råvare der er blevet brugt i en recept.

Nogle oplagte forbedringer man kunne implementere, men som vi ikke har haft tid til er:

- I stedet for indtastningsfelter de steder hvor der skal angives et id på et element der allerede skal være i databasen, f.eks. når man skal oprette et råvare batch af en råvare, så ville det være smartere at have en "drop-down"-liste man kunne vælge mellem eksisterende elementer i databasen fra i stedet for at skulle huske hvilke id'er der findes og hvilket id en specifik råvare har.
- Det kunne være en go' ide at have nogle ekstra værktøjer til administratoren, således at han/hun kunne vælge hvilken database der skal forbindes til, oprette en ny database og måske kopiere en database fra et sted til et andet.
- Hvis databasen bliver stor nok, ville helt sikkert være rart at kunne sortere de forskellige listevisninger på andet end ID eller endda kunne søge i de forskellige tabeller
- Det kunne også være en ide at give operatøren læserettigheder de produktbatches han har ansvaret for og kunne printe sedler ud selv eller få dem vist i sin browser på en smartphone eller tablet.
- Vores vægtsimulator kan stortset kun lige det vi skal bruge i forhold til vores ASE og vi skyder også nogle genveje i den som vi kun kan tillade os fordi vi kender sekvensen af de beskedder der kommer fra vores ASE, den kunne bestemt godt udviddes til at simulere en vægt bedre.

Alt i alt er vi ret godt tilfredse med hvad vi har nået at implementere og vi føler også det er lavet ud fra et solidt design der overholder 3-lagsmodellen på en go' måde. Vi kunne godt have brugt mere tid på at teste vores program, men vi følte at det var vigtigere at have noget der opfyldte kravsspecifikationen så vidt muligt i stedet for et grundigt testet system der ikke kunne alt det skulle kunne.

## 6 Database

Her vil vi beskrive vores tabeller i databasen hver for sig. Vi vil beskrive dem ud fra det **MySQL** script, som bruges til at oprette databasen.

### **user**

user tabellen indeholder alle de vigtige oplysninger om en bruger i vores system, som er vigtige at holde styr på. vores `user_id` er primær nøgle i user tabellen, den må ikke være NULL. derudover indeholder tabellen `user_name` som er en `varchar(20)` hvilket betyder at det er en String som maks må være 20 karakterer lang. resten af variablerne i vores user tabel, bruger ikke nye værdier blot kortere længder af strenge eller færre cifre i vores integers.

## productbatch

i productbatch tabellen har vi 1 primær nøgle om er pb\_id og som er unikt for hvert produktbatch. derudover indeholder tabellen prescription\_id som er en fremmed nøgle som kommer fra prescription tabellen, for at den kan oprettes skal prescription\_id'et altså findes i en allerede eksisterende tuppel i prescription. Noget som vi ikke tidligere har haft med i vores andre tabeller er er en date, det er en bestemt måde at holde en dato formateret på og i dette tilfælde er formateringen: '2014-18-06', det giver os en ekstra feature i vores system, så vi også kan holde styr på hvornår en prescription er blevet oprettet.

## productbatchcomponent

productbatchcomponent indeholder 3 nøgler og de er alle fremmednøgler fra andre tabeller pb\_id kommer fra productbatch, commoditybatch\_id kommer fra commoditybatch og user\_id kommer fra user, derudover indeholder tabellen to variabler mere, det er tara og netto som begge er decimal værdier med decimal(7,4) som betyder at der er 7 cifre hvor de 4 sidste er decimal tal.

## commodity

i commodity har vi en enkelt primær nøgle som er commodity\_id, derudover indeholder den to varchar variabler som er henholdsvis commodity\_name og supplier. Begge strenge med længden maks 20.

## commoditybatch

commoditybatch har to nøgler hvor af den ene er primær nøgle. det er commodity\_id som er primær nøgle, den er unik og må ikke være NULL det er en INT(11) værdi, derudover indeholder tabellen en anden nøgle som er en fremmednøgle fra commodity og det er commodity\_id som er denne nøgle, den indeholder også en amount, som svarer til den mængde af en commodity man har på lager.

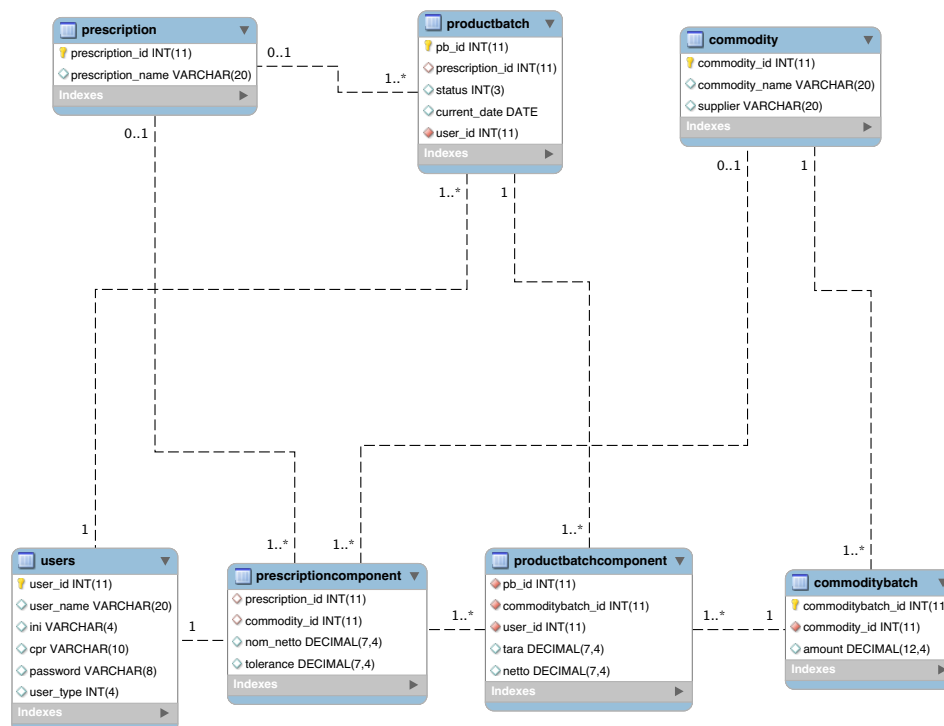
## prescription

prescription har vi brugt nøgler fra i tidligere tabeller, det er primær nøglen *prescription\_id* som er blevet brugt i andre tabeller. derudover har en prescription et prescription\_name.

## prescriptioncomponent

prescriptioncomponent har to nøgler hvor de begge to er fremmednøgler fra andre tabeller. prescription\_id kommer fra prescription og commodity\_id kommer fra commodity. tabellen indeholder også to variabler mere: nom\_netto og tolerance, det er begge decimal(7,4).

Overblikket over hele databasen, kan fåes i figur 6.1 på næste side.



Figur 6.1: EER Diagram over DB

## 7 Test

### 7.1 Generel teststrategi

Vores generelle teststrategi, var at teste hver gang vi var færdig med en del af web-interfacet, at teste den op mod vores database. Den omvendte situation har vi selvfølgelig også gjort brug af.

Vores interface har vi implementeret så det viser evt. intastningsfejl. Dermed kunne vi hurtigt se, hvis vi prøvede at lægge en forkert værdi i vores database, eller vores database var opsat forkert i forhold til værdien. Et eksempel på en fejlmeddelelse ses i figur 7.1.

The screenshot shows a web form with the title "Indtast nye råvareoplysninger". A red error message box at the top states: "Det indtastede råvare id er ikke et tal". Below the error, there are three input fields:

- Råvare ID**: string
- Råvarenavn**: stringid
- Leverandør**: stringtesteren

At the bottom of the form, there are three buttons: "Opret", "Tilbage", and "Log ud".

Figur 7.1: Visning af fejl i web-interface

Et eksempel på et successcenarie ses i figur 7.2 på næste side.



Figur 7.2: Visning af successscenarie i web-interface

Derudover kan det nævnes vi har brugt **Eclipse's** *debug* værktøj til at holde øje med, hvordan vores program opfører sig i bestemte sekvenser. Sekvenserne blev bestemt med passende *breakpoints*. Herfra kunne vi så se hvor vores program sprang hen og holde øje med værdierne på vores variable.

## 7.2 JUnit

For at dokumentere vi rent faktisk har testet vores program bare en smule, har vi valgt at inkludere et par **JUnit** testklasser. Vi valgte at lave en tesklasse til vores brugere og en til vores råvarer. Disse er valgt, da de er helt essentielle for at kunne bruge systemet, senere oprette recepter og i sidste ende produkt-batches.

Vi vil ikke gennemgå begge testklasser, da de fungerer på samme måde, men vi vil løbe klassen, der tester på brugere igennem.

### UserDataTest

Det første vi gør er, at oprette testobjekter til brug i testen. I vores `setUp()` metodede erklærer vi objekterne med værdier, vi skal bruge i testen. Samtidig har vi en `tearDown()` metode, der rydder op, når vores test er færdig. Vi har efterfølgende lige en metode, der tjekker om objekterne overhovedet kommer til at eksistere.

Endelig kommer vi til at teste de metoder, vi også bruger i den oprindelige klasse. I dette tilfælde tester vi en `testCreateUser` metode, der opretter en bruger med et bestemt ID. Når brugeren er oprettet i databasen tjekker vi om hans ID passer med det vi har tildelt ham i vores **Java** kode. Til sidst sletter vi ham igen fra databasen.

I `testGetUser` sker der faktisk det samme, her opretter vi nemlig en bruger i databasen inden tjekket, og henter ham som test. Til sidst tjekkes der om ID passer og ryddes op.

Vi kan selvfølgelig også hente en liste af brugere, og denne er måske lidt mere interessant, da vi jo ikke bare skal tjekke en værdi, men tjekke om alle brugere oprettes som forventet.

Dette gøres med et for-loop figur 7.3. Dette loop henter en liste ud fra databasen og laver en `arrayList`. Den nye liste matches herefter op mod vores oprindelige liste.

```
List<UserDTO> actual = userData.getUserList();

// Run through the list and check if elements match
for (int i = 0; i < expected.size(); i++) {
    expected.equals(actual);
}
```

Figur 7.3: Udsnit af testklassen, der viser loop-testen

Den sidste metode vi tester fungerer lidt på samme måde. Her tjekker vi bare om listen er tom.

I bunden af **Eclipse** kan man se om ens test er gået godt figur 7.4 på næste side. I så fald vil alle metoder være markeret med en grøn farve og en tid, der indikerer at testen er god nok og tog den skrevne tid at færdiggøre.



Figur 7.4: Testens resultat

### 7.3 Indvirkning af teststrategi

Vi fik fanget en del fejl i vores generelle test mønster i web-delen. Dette bevirkede faktisk at vores **JUnit** tests tog forholdsvis lang tid at lave i forhold til, at vi ikke fandt nogen fejl ved hjælp af dem. Man kunne dog udvide vores tests til at indeholde flere grænseværdier, da det generelt er meget få brugere vi tester på. Dette vil helt sikkert være noget man skulle vægte lidt højere, hvis tidsrammen også var lidt større for projektet.

## 8 Konklusion

Resultatet af vores delopgaver i løbet af kurset gjorde at vi havde forholdsvis let ved at stykke disse sammen til opbygningen af det endelige projekt. Vi har fået udviklet et afvejningssystem, der kan hjælpe vores fiktive medicinalvirksomhed med at afveje deres råvarer. Disse råvarer kan indgå i forskellige recepter, og heraf kan der laves produkt-batches. Alle informationer om de forskellige elementer kan arkiveres i en database til dokumentation. Derudover har vi udviklet et web-interface, der kan bruges til lagerstyring. Dette kommunikerer selvfølgelig med selvsamme database.

Vores design af systemet tager udgangspunkt i den 3-lagsmodel, vi har fået gennemgået i løbet af kurset. Samtidig understøtter vores web-arkitektur **MVC** mønstret. Vi har derfor opnået et robust system, der er let at vedligeholde.

Vores implementering er lavet, så vi følger vores designprincip. Det vil sige klasserne er opdelt i lag, der svarer til 3-lagsmodellen. Vi har taget nogle valg i forhold til uoverensstemmelser eller fortolkbare oplysninger i opgavebeskrivelsen. Dette kommer specielt til udtryk i forhold til use-case 5 i opgavebeskrivelsen, hvor værkføreren tildeler en ny produktbatch til en udvalgt operatør. Bilagene siger herefter at Operatøren har ansvaret for afvejning af disse. Dette har vi tolket som om, det kun er den tildelte operatør, der kan stå for afvejningen, da han har fået tildelt ansvaret for den. Vi har dog givet mulighed for, at værkføreren kan ændre operatøren, hvis bestemte kriterier er opfyldt.

Vi har forsøgt at nå at implementere alle use-cases på en tilfredsstillende måde, og mener faktisk vi har opnået en implementering der opfylder minimumskravene. Dog kunne vi nemt forbedre systemet med funktioner, der kunne gøre det endnu mere brugbart. Dette ville dog kræve længere tid, og dermed flere penge for kunden. Indenfor de rammer vi har haft må vi konstatere, at vi godt kan komme op med et brugbart system. Hvis man som kunde vil have endnu flere detaljer i programmet, må kunden også forvente en større udskrivning og en længere ventetid på et bedre system.

At tid er penge gør sig også gældende i forhold til test. Vi har testet systemet løbende og har også løbende fundet fejl. Vi er også helt sikre på vi kunne finde flere hvis vi havde mere tid. Vi synes dog at vi har fået et okay resultat ud af vores løbende test. Samtidig har vi lavet et par **JUnit** teskasser, som gav et synligt bevis på at netop de metoder virkede efter hensigten. Dog kunne vi sagtens have lavet dem med flere grænseværdier og måske endda teste flere klasser. Igen ville dette bare kræve mere tid, og dermed igen gå ud over kunden.

Alt i alt mener vi at vi er kommet ud med en udemærket løsning i forhold til den tid vi har brugt på systemet. Da løsningen er nået inden for tidsrammen vil vi mene at kunden også må være tilfreds. Hvis kunden ønskede en endnu bedre løsning, var han formegentlig også indstillet på at vente lidt længere og betale mere.

## Referencer

- [A07] DTU-Netteknologi A. *3lagsmodellen*. 2007. URL: <http://dtu20.be/3LM.pdf>.
- [DTU14] DTU. *Webarkitektur*. 2014. URL: <https://docs.google.com/presentation/d/1Pk9VznAttLL9fIzf5vLhS/edit#slide=id.p15>.
- [Gru14a] Gruppe53. „CDIO 1 02324“. I: (2014).
- [Gru14b] Gruppe53. „CDIO 2 02324“. I: (2014).

## 9 Anvendte værktøjer

Vi har til fremstillingen af afvejningssystemet benyttet os af følgende værktøjer: **Eclipse** har vi brugt til at skrive **Java**, **JSP**, **HTML**, og **CSS**. Vi bruger **Tomcat** server til afvikle vores servlets og .jsp-filer på. Derudover har vi brugt **MySQL workbench** til at generere vores database, og en **MySQL** server for at kunne tilgå databasen.

Vi har i forbindelse med implementeringen også brugt **GitHub**, for at kunne arbejde samtidig på forskellige dele af koden. Samtidig har vi brugt det til versionsstyring.

Til at udarbejde rapporten har vi brugt  $\text{\LaTeX}$ , med forskellige pakker som vi ikke vil komme nærmere ind på. Dog skal det nævnes at det er en online distribution af  $\text{\LaTeX}$  kaldet **ShareLaTeX**, der gør at vi alle kan sidde og skrive i det samme dokument samtidig.

Til at udarbejde **UML** diagrammer har vi benyttet **UMlet**. Til at dele forskellige filer og kontakt til hindanden har vi benyttet os af en gruppe oprettet til formålet på **Facebook**.

Vi har skrevet et lille afsnit om selve samarbejdet i gruppen i appendiks B på side 179.

## A Kode

Her vil hele vores kode til programmet være repræsenteret som bilag.

### A.1 ase

Al kode til vores ASE

#### A.1.1 ase.boundary

Koden i vores ase boundary pakke

##### ASEBoundary

```
1 package ase.boundary;
2
3 import java.io.BufferedReader;
4 import java.io.DataOutputStream;
5 import java.io.IOException;
6 import java.io.InputStreamReader;
7 import java.net.Socket;
8
9 /**
10  * Class to take input from a client
11  *
12  * @author Gruppe 53
13  *
14  */
15 public class ASEBoundary {
16     BufferedReader socketReader;
17     private DataOutputStream socketOutput;
18
19     /**
20      * Constructor that makes the program ready for user inputs
21      */
22     public ASEBoundary(String adress) {
23         try {
24
25             @SuppressWarnings("resource")
26             Socket sock = new Socket(adress, 8000);
27             socketReader = new BufferedReader(new InputStreamReader(
28                 sock.getInputStream()));
29             socketOutput = new DataOutputStream(sock.getOutputStream());
30
31         } catch (Exception e) {
32             System.out.println("Kunne ikke forbinde til vægten.");
33             System.out.println(e.getMessage());
34         }
35     }
36
37     public int getID() throws IOException {
38
39         String sentence = "RM20 4 \"Indtast operatoer id\" \"\" \"nr\"";
40         socketOutput.writeBytes(sentence + "\r\n");
```



```
41
42     String input;
43     do{
44         input = socketReader.readLine();
45         System.out.println(input);}
46     while (input.equals("") || input.startsWith("RM20 B") || input.
        startsWith("RM20 I"));
47
48     if (input.equals("RM20 C"))
49         return -1;
50
51     input = input.substring(input.indexOf("\") + 1,
52         input.lastIndexOf("\")+1);
53     input = input.replace("\", \"").trim();
54     try {
55         int id = Integer.parseInt(input);
56         return id;
57     } catch (NumberFormatException e) {
58         return -1;
59     } catch (Exception e) {
60         System.err.println(input);
61         return -1;
62     }
63 }
64
65 public int getProductBatchID() throws IOException {
66
67     String sentence = "RM20 4 \"Produktbatchnummer?\" \"\" \"nr\"";
68     socketOutput.writeBytes(sentence + "\r\n");
69     String input;
70     do
71         input = socketReader.readLine();
72     while (input.equals("") || input.startsWith("RM20 B") || input.
        startsWith("RM20 I"));
73
74     input = input.substring(input.indexOf("\") + 1,
75         input.lastIndexOf("\"));
76     try {
77         int id = Integer.parseInt(input);
78         return id;
79     } catch (NumberFormatException e) {
80         return -1;
81     }
82 }
83
84 public boolean clearWeight() throws IOException {
85
86     String sentence = "RM20 8 \"Toem vaegt\" \"Tryk OK\" \"\"";
87     socketOutput.writeBytes(sentence + "\r\n");
88
89     String input;
90     do
91         input = socketReader.readLine();
92     while (input.equals("") || input.startsWith("RM20 B") || input.
        startsWith("RM20 I"));
```

```

93         if (input.startsWith("RM20 A")) {
94             socketOutput.writeBytes("T\r\n");
95             return true;
96         } else
97             return false;
98     }
99 }
100
101
102 public double getTara() throws IOException {
103     String sentence = "RM20 8 \"Placer beholder\" \"OK/Cancel\" \"\"";
104     socketOutput.writeBytes(sentence + "\r\n");
105     String input;
106     do
107         input = socketReader.readLine();
108     while (!input.startsWith("RM20") || input.startsWith("RM20 B") ||
109            input.startsWith("RM20 I"));
110     if (!input.startsWith("RM20 A"))
111         return -1.0;
112
113     socketOutput.writeBytes("T\r\n");
114     do
115         input = socketReader.readLine();
116     while (input.equals(""));
117     if (input.startsWith("T S")) {
118         try {
119             return Double.parseDouble(input.replaceAll("[^\\d\\.]", ""));
120         } catch (NumberFormatException e) {}
121     }
122     return -1.0;
123 }
124
125
126 public double getNettoWeight(double target, double tolerance)
127     throws IOException {
128     String sTol = "" + tolerance;
129     String sTar = "" + target;
130     try {
131         sTol = sTol.substring(0, sTol.indexOf(".") + 2);
132         sTar = sTar.substring(0, sTar.indexOf(".") + 5);
133     } catch (IndexOutOfBoundsException e) {}
134
135     String sentence = "P111 \"\" + target + "kg\t" + sTol + "%\"";
136     socketOutput.writeBytes(sentence + "\r\n");
137     sentence = "RM30 \"\" \"\" \"\" \"\" \"\" \"\" \"OK\" \"Cancel\"";
138     socketOutput.writeBytes(sentence + "\r\n");
139     socketOutput.writeBytes("RM39 1\r\n");
140
141     String input;
142
143     double netto = -1.0;
144     do {
145         input = socketReader.readLine();
146     } while (!input.startsWith("RM30 A"));

```

```
147
148     if (!input.startsWith("RM30 A 5"))
149         return -1.0;
150
151     socketOutput.writeBytes("RM39 2\r\n");
152
153     socketOutput.writeBytes("S\r\n");
154     do {
155         input = socketReader.readLine();
156     } while (!input.startsWith("S S"));
157
158     try {
159         netto = Double.parseDouble(input.replaceAll("[^\\d\\.]", ""));
160         return netto;
161     } catch (NumberFormatException e) {
162         return -1.0;
163     }
164 }
165
166 public int getRaavareBatchID(int commodityID) throws IOException {
167     String sentence = "RM20 4 \"Raavarebatch id\" \"\" \"ID\"";
168     socketOutput.writeBytes(sentence + "\r\n");
169
170     String input;
171     do
172         input = socketReader.readLine();
173     while (input.equals("") || input.startsWith("RM20 B") || input.
        startsWith("RM20 I"));
174     if (!input.startsWith("RM20 A"))
175         return -1;
176
177     input = input.substring(input.indexOf("\"") + 1,
178         input.lastIndexOf("\""));
179     try {
180         int id = Integer.parseInt(input);
181         return id;
182     } catch (NumberFormatException e) {
183         return -1;
184     }
185 }
186
187 public boolean getQuit() throws IOException {
188     String sentence = "RM20 8 \"Fortsat Afvejning?\" \"OK = ja Cancel =
        nej\" \"\"";
189     socketOutput.writeBytes(sentence + "\r\n");
190
191     String input;
192     do
193         input = socketReader.readLine();
194     while (input.equals("") || input.startsWith("RM20 B") || input.
        startsWith("RM20 I"));
195     if (input.startsWith("RM20 A"))
196         return false;
197     return true;
198 }
```

```

199
200 public boolean retry() throws IOException {
201
202     String sentence = "RM20 8 \"Vil du proeve igen?\" \"OK = ja Cancel =
        nej\" \"\"";
203     socketOutput.writeBytes(sentence + "\r\n");
204     String input;
205     do
206         input = socketReader.readLine();
207     while (input.equals("") || input.startsWith("RM20 B") || input.
        startsWith("RM20 I"));
208     if (input.startsWith("RM20 A"))
209         return true;
210     return false;
211 }
212
213
214 public void sendError(String msg) throws IOException {
215     String sentence = "RM20 8 \"" + msg + "\" \"\" \"\"";
216     socketOutput.writeBytes(sentence + "\r\n");
217     String input;
218     do
219         input = socketReader.readLine();
220     while (input.equals("") || input.startsWith("RM20 B") || input.
        startsWith("RM20 I"));
221
222 }
223
224 public boolean sendConfirm(String msg) throws IOException {
225     String sentence = "RM20 8 \"" + msg + "\" \"\" \"\"";
226     socketOutput.writeBytes(sentence + "\r\n");
227     String input;
228     do
229         input = socketReader.readLine();
230     while (input.equals("") || input.startsWith("RM20 B") || input.
        startsWith("RM20 I"));
231     if (input.startsWith("RM20 A"))
232         return true;
233     return false;
234 }
235 }
236
237 }

```

### A.1.2 ase.controller

Koden i vores ase controller pakke

#### ASEController

```

1 package ase.controller;
2
3 import java.util.List;
4 import java.io.IOException;
5
6 import admin.data.CommodityBatchDTO;

```

```
7 import admin.data.CommodityBatchData;
8 import admin.data.DALException;
9 import admin.data.ICommodityBatchDAO;
10 import admin.data.IPrescriptionDAO;
11 import admin.data.IProductBatchCompDAO;
12 import admin.data.IProductBatchDAO;
13 import admin.data.IUserDAO;
14 import admin.data.PrescriptionCompDTO;
15 import admin.data.PrescriptionDTO;
16 import admin.data.PrescriptionData;
17 import admin.data.ProductBatchCompDTO;
18 import admin.data.ProductBatchCompData;
19 import admin.data.ProductBatchDTO;
20 import admin.data.ProductBatchData;
21 import admin.data.StatusType;
22 import admin.data.UserData;
23 import ase.boundary.ASEBoundary;
24
25 public class ASEController {
26
27     private static boolean done, reset = false;
28     private static boolean step_done, subRoutineDone;
29     private static int userID, productBatchID, commodityID,
        commodityBatchID;
30     private static double tare, netto;
31
32     static IProductBatchDAO products = new ProductBatchData();
33     static IProductBatchCompDAO proComps = new ProductBatchCompData();
34
35     public static void main(String args[]) throws IOException {
36         String address = "169.254.2.2";
37         try {
38             address = args[0];
39         } catch (Exception e) {}
40
41         while(true){
42             ASEBoundary bound = new ASEBoundary(address);
43             try {
44                 menu(bound);
45             } catch (Exception e) {}
46             try {
47                 Thread.sleep(1000);
48             } catch (InterruptedException e) {}
49         }
50     }
51
52     public static void menu(ASEBoundary bound) throws IOException {
53         while (!done) {
54             reset = false; // Bliver sat til true, hvis vi ønsker at
55                             // genstarte.
56             step_done = false;
57             while (!step_done) { // Step 3+4
58                 String userName;
59
60                 userID = bound.getID();
```

```
61         if (userID < 0)
62             continue;
63         try {
64             IUserDAO users = new UserData();
65             userName = users.getUser(userID).getUsername();
66             if (bound.sendConfirm(userName)) {
67                 step_done = true;
68             } else {
69                 continue;
70             }
71         }
72
73         catch (DAException e) {
74             bound.sendError("Fejl i bruger id");
75             continue;
76         }
77     }
78
79     step_done = false;
80     while (!reset && !step_done) { // step 5+6 Produktbatch input
81         // state
82         productBatchID = bound.getProductBatchID();
83         try {
84             IPrescriptionDAO prescriptions = new PrescriptionData();
85             ProductBatchDTO item = products
86                 .getProductBatch(productBatchID);
87             PrescriptionDTO prescription = prescriptions
88                 .getPrescription(item.getPrescriptionId());
89
90             if (item.getUserId() != userID) {
91                 bound.sendError("Forkert Operatoer");
92                 if (bound.retry()) {
93                     continue;
94                 } else {
95                     reset = true;
96                 }
97             } else if (item.getStatus() != StatusType.NEW
98                 && item.getStatus() != StatusType.PAUSED) {
99                 bound.sendError("Fejl i status:"
100                     + item.getStatus().ordinal());
101                 if (bound.retry()) {
102                     continue;
103                 } else {
104                     reset = true;
105                 }
106             } else {
107                 if (!bound.sendConfirm(prescription.getName()))
108                     if (bound.retry()) {
109                         continue;
110                     } else {
111                         reset = true;
112                     }
113                 step_done = true;
114             }
115         }
```

```
116         } catch (DAException e) {
117             bound.sendError(e.getMessage());
118             if (bound.retry()) {
119                 continue;
120             } else {
121                 reset = true;
122             }
123         }
124     }
125 }
126
127 while (!subRoutineDone && !reset) {
128     // step 12,5
129     PrescriptionCompDTO component = null;
130
131     try {
132         List<PrescriptionCompDTO> preCompList = proComps
133             .getUnfulfilledComps(productBatchID);
134         if (preCompList.isEmpty()) {
135             bound.sendError("Batch afsluttet");
136             ProductBatchDTO product = products
137                 .getProductBatch(productBatchID);
138             product.setStatus(StatusType.FINISHED);
139             products.updateProductBatch(product);
140             reset = true;
141         } else {
142             component = preCompList.get(0);
143             commodityID = component.getCommodityId();
144         }
145     } catch (DAException e) {
146         bound.sendError("Der skete en fejl");
147         reset = true;
148     }
149
150     // step 7–12. Raavarebatch input
151     // state
152     if (!reset && !bound.clearWeight()) {
153         reset = true;
154     }
155
156     if (!reset)
157         tare = bound.getTara();
158     if (tare < 0) {
159         reset = true;
160     }
161
162     step_done = false;
163     while (!reset && !step_done) { // step 13. Indtast på varebatch
164         // ID
165         // state
166         ICommodityBatchDAO commBatch = new CommodityBatchData();
167         commodityBatchID = bound.getRaavareBatchID(commodityID);
168         if (commodityBatchID == -1) {
169             reset = true;
170         }
171     }
```

```
171     try {
172         CommodityBatchDTO comBatch = commBatch
173             .getCommodityBatch(commodityBatchID);
174         if (!reset && comBatch.getCommodityId() != commodityID) {
175             bound.sendError("BatchID forkert.");
176             if (bound.retry()) {
177                 continue;
178             } else {
179                 reset = true;
180             }
181         }
182         step_done = true;
183     } catch (DAException e) {
184         bound.sendError("Der skete en fejl");
185         reset = true;
186     }
187 }
188
189
190 step_done = false;
191 while (!reset && !step_done) { // step 14
192
193     double dataNetto = component.getNomNetto();
194     double dataTolerance = component.getTolerance();
195     netto = bound.getNettoWeight(dataNetto, dataTolerance);
196     if (netto == -1.0) {
197         reset = true;
198     } else if (netto >= dataNetto * (1 + dataTolerance / 100)) {
199         bound.sendError("Nettovaegt for stor");
200         if (bound.retry())
201             continue;
202         else {
203             reset = true;
204         }
205     } else if (netto <= dataNetto * (1 - dataTolerance / 100)) {
206         bound.sendError("Nettovaegt for lav");
207         if (bound.retry())
208             continue;
209         else {
210             reset = true;
211         }
212     } else {
213         try {
214             ProductBatchCompDTO saveData = new ProductBatchCompDTO(
215                 productBatchID, commodityBatchID, userID,
216                 tare, netto);
217             proComps.createProductBatchComp(saveData);
218             step_done = true;
219         } catch (DAException e) {
220             bound.sendError("Kunne ikke gemme");
221             if (bound.retry())
222                 continue;
223             else {
224                 reset = true;
225             }
226         }
227     }
228 }
```



```
226
227     }
228   }
229 }
230
231 step_done = false;
232 while (!reset && !step_done) { // step 14,5
233   if (bound.getQuit()) {
234     reset = true;
235     try {
236       ProductBatchDTO product = products
237         .getProductBatch(productBatchID);
238       List<PrescriptionCompDTO> preCompList = proComps
239         .getUnfulfilledComps(productBatchID);
240       if (preCompList.isEmpty()) {
241         product.setStatus(StatusType.FINISHED);
242       } else {
243         product.setStatus(StatusType.PAUSED);
244       }
245       products.updateProductBatch(product);
246     } catch (DAException e) {
247       bound.sendError("Uventet fejl!");
248       reset = true;
249     }
250   } else
251     step_done = true;
252 }
253 }
254 }
255 }
256 }
```

## A.2 admin

Al kode til administrering af webinterface

### A.2.1 admin.controller

Koden i vores **admin** controller pakke

#### AbstractController

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Error><Code>NoSuchKey</Code><Message>The specified key does not exist.</
  Message><Key>5339631428048190250001ec/53a3313f50d221e952a561b2</Key><
  RequestId>EA505402538C01C5</RequestId><HostId>o8H76FW/+
  iQLo5kxs0PXys030yswIuvbtRsBGKKmKR31u+uA0Hv+spimSgWT9KAr</HostId></
  Error>

```

#### BatchCommodityAdminController

```

1 package admin.controller;
2
3 import java.io.IOException;
4 import java.util.List;
5
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 import admin.data.CommodityBatchDTO;
12 import admin.data.UserType;
13
14 public class BatchCommodityAdminController extends AbstractController {
15
16     private static final long serialVersionUID = 1L;
17
18     @Override
19     protected UserType requiredAccessLevel() {
20         return UserType.PHARMACIST;
21     }
22
23     @Override
24     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
        throws ServletException, IOException{
25
26         try {
27             List<CommodityBatchDTO> commodityBatchList = comBatches.
                getComBatchList();
28             request.setAttribute("commodityBatchList", commodityBatchList);
29         } catch (Exception e){
30
31         }
32
33         RequestDispatcher dispatcher =
34             getServletContext().getRequestDispatcher("/
                commodityBatch_admin_boundary.jsp");

```

```
35     dispatcher.forward(request, response);
36 }
37 }
38 }

BatchCommodityConfirmDeleteController

1  package admin.controller;
2
3  import java.io.IOException;
4
5  import javax.servlet.RequestDispatcher;
6  import javax.servlet.ServletException;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 import admin.data.CommodityBatchDTO;
11 import admin.data.DALException;
12 import admin.data.UserType;
13
14 public class BatchCommodityConfirmDeleteController extends
    AbstractController {
15
16     private static final long serialVersionUID = 1L;
17
18     @Override
19     protected UserType requiredAccessLevel() {
20         return UserType.PHARMACIST;
21     }
22
23     @Override
24     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
25         throws ServletException, IOException{
26
27         String error = "";
28         CommodityBatchDTO comBatch = null;
29         boolean success = false;
30
31         int id = 0;
32         try {
33             id = Integer.parseInt(request.getParameter("id"));
34         } catch (NumberFormatException e) {
35             error = "Råvare id er ugyldigt";
36         }
37
38         try {
39             comBatch = comBatches.getCommodityBatch(id);
40         } catch (DALException e) {
41             error = "Der findes ingen råvarebatch med det givne id";
42         }
43
44         if (request.getMethod().equals("POST")){
45             try {
46                 comBatches.deleteCommodityBatch(id);
47                 success = true;

```

```
48         } catch (DALException e) {
49             error = "Der skete en fejl under sletning <BR>";
50             error += e.getMessage();
51         }
52     }
53
54     request.setAttribute("comBatch", comBatch);
55
56     request.setAttribute("done", success);
57     request.setAttribute("error", error);
58
59     RequestDispatcher dispatcher = getServletContext()
60         .getRequestDispatcher("/commodityBatch_confirm_delete_boundary.
        jsp");
61     dispatcher.forward(request, response);
62
63 }
64 }
```

### BatchCommodityEditController

```
1  package admin.controller;
2
3  import java.io.IOException;
4
5  import javax.servlet.RequestDispatcher;
6  import javax.servlet.ServletException;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 import admin.data.CommodityBatchDTO;
11 import admin.data.DALException;
12 import admin.data.UserType;
13
14 public class BatchCommodityEditController extends AbstractController {
15
16     private static final long serialVersionUID = 1L;
17
18     @Override
19     protected UserType requiredAccessLevel() {
20         return UserType.PHARMACIST;
21     }
22
23     @Override
24     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
25         throws ServletException, IOException{
26
27         int iNewId = 0;
28         boolean isNew = false;
29         boolean anyError = false;
30         String majorError = "";
31         String idError = "";
32         String comIdError = "";
33         String amountError = "";
34         String sNewId = "";
```

```
35 String sNewComId = "";
36 String sNewAmount = "";
37 CommodityBatchDTO comBatch = null;
38
39 String sComBatchId = request.getParameter("id");
40 int iOldId = 0;
41
42 if (sComBatchId != null && sComBatchId.equals("new")) {
43     isNew = true;
44 } else {
45     try {
46         iOldId = Integer.parseInt(sComBatchId);
47     } catch (NumberFormatException e) {
48         majorError = "RåvarebatchId er ugyldigt";
49         anyError = true;
50     }
51 }
52
53 if (isNew) {
54     comBatch = new CommodityBatchDTO();
55 } else {
56     try {
57         comBatch = comBatches.getCommodityBatch(iOldId);
58         sNewId = sComBatchId;
59         sNewComId = "" + comBatch.getCommodityBatchId();
60         sNewAmount = "" + comBatch.getAmount();
61     } catch (DALError e1) {
62         majorError = "Der findes intet råvarebatch med det angivne id";
63         anyError = true;
64     }
65 }
66
67 if (! anyError && request.getMethod().equals("POST")) {
68     sNewId = request.getParameter("newId");
69     sNewComId = request.getParameter("newComId");
70     sNewAmount = request.getParameter("newAmount");
71
72     try {
73         iNewId = Integer.parseInt(sNewId);
74         comBatch.setCommodityBatchId(iNewId);
75     } catch (NumberFormatException e) {
76         idError = "Det indtastede råvarebatch id er ikke et helt tal";
77         anyError = true;
78     } catch (DALError e) {
79         idError = e.getMessage();
80         anyError = true;
81     }
82
83     try {
84         int iNewComId = Integer.parseInt(sNewComId);
85         comBatch.setCommodityId(iNewComId);
86     } catch (NumberFormatException e) {
87         comIdError = "Det indtastede råvare id er ikke et helt tal";
88         anyError = true;
89     } catch (DALError e) {
```

```
90         comIdError = e.getMessage();
91         anyError = true;
92     }
93
94     try {
95         double dNewAmount = Double.parseDouble(sNewAmount);
96         comBatch.setAmount(dNewAmount);
97     } catch (NumberFormatException e) {
98         amountError = "Den indtastede mængde er ikke et tal";
99         anyError = true;
100     } catch (DALErrorException e) {
101         amountError = e.getMessage();
102         anyError = true;
103     }
104
105     if (!anyError) {
106         if (isNew) {
107             try {
108                 comBatches.createCommodityBatch(comBatch);
109             } catch (DALErrorException e) {
110                 idError = e.getMessage();
111                 anyError = true;
112             }
113         } else if (iOldId == iNewId) {
114             try {
115                 comBatches.updateCommodityBatch(comBatch);
116             } catch (DALErrorException e) {
117                 idError = e.getMessage();
118                 anyError = true;
119             }
120         } else {
121             try {
122                 comBatches.createCommodityBatch(comBatch);
123             } try {
124                 comBatches.deleteCommodityBatch(iOldId);
125             } catch (DALErrorException e) {
126                 idError = "Råvarebatch id kunne ikke ændres <BR>";
127                 idError += e.getMessage();
128                 anyError = true;
129                 comBatches.deleteCommodityBatch(iNewId);
130             }
131         } catch (DALErrorException e) {
132             idError = e.getMessage();
133             anyError = true;
134         }
135     }
136 }
137
138 }
139
140 request.setAttribute("majorError", majorError);
141 request.setAttribute("idError", idError);
142 request.setAttribute("comIdError", comIdError);
143 request.setAttribute("amountError", amountError);
144
```

```

145     request.setAttribute("create", isNew);
146     request.setAttribute("complete", !anyError
147         && request.getMethod().equals("POST"));
148
149     request.setAttribute("newId", sNewId);
150     request.setAttribute("newComId", sNewComId);
151     request.setAttribute("newAmount", sNewAmount);
152     request.setAttribute("comBatch", comBatch);
153
154
155     RequestDispatcher dispatcher = getServletContext()
156         .getRequestDispatcher("/commodityBatch_edit_boundary.jsp");
157     dispatcher.forward(request, response);
158
159 }
160 }

```

### CommodityAdminController

```

1  package admin.controller;
2
3  import java.io.IOException;
4  import java.util.List;
5
6  import javax.servlet.RequestDispatcher;
7  import javax.servlet.ServletException;
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10
11 import admin.data.CommodityDTO;
12 import admin.data.UserType;
13
14 public class CommodityAdminController extends AbstractController {
15
16     private static final long serialVersionUID = 1L;
17
18     @Override
19     protected UserType requiredAccessLevel() {
20         return UserType.PHARMACIST;
21     }
22
23     @Override
24     public void doRequest(HttpServletRequest request, HttpServletResponse
25         response)
26         throws ServletException, IOException{
27         try {
28             List<CommodityDTO> commodityList = commodities.getComList();
29             request.setAttribute("commodityList", commodityList);
30         } catch (Exception e){
31         }
32
33         RequestDispatcher dispatcher =
34             getServletContext().getRequestDispatcher("/
35                 commodity_admin_boundary.jsp");
36         dispatcher.forward(request, response);
37     }
38 }

```

```
36
37     }
38
39 }

CommodityConfirmDeleteController

1  package admin.controller;
2
3  import java.io.IOException;
4
5  import javax.servlet.RequestDispatcher;
6  import javax.servlet.ServletException;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 import admin.data.CommodityDTO;
11 import admin.data.DALException;
12 import admin.data.UserType;
13
14 public class CommodityConfirmDeleteController extends AbstractController
15 {
16     private static final long serialVersionUID = 1L;
17
18     @Override
19     protected UserType requiredAccessLevel() {
20         return UserType.PHARMACIST;
21     }
22
23     @Override
24     public void doRequest(HttpServletRequest request, HttpServletResponse
25         response)
26         throws ServletException, IOException{
27
28         String error = "";
29         CommodityDTO com = null;
30         boolean success = false;
31
32         int id = 0;
33         try {
34             id = Integer.parseInt(request.getParameter("id"));
35         } catch (NumberFormatException e) {
36             error = "Råvare id er ugyldigt";
37         }
38
39         try {
40             com = commodities.getCommodity(id);
41         } catch (DALException e) {
42             error = "Der findes ingen råvare med det givne id";
43         }
44
45         if (request.getMethod().equals("POST")){
46             try {
47                 commodities.deleteCommodity(id);
48                 success = true;
49             }
50         }
51     }
52 }
```



```
48         } catch (DALException e) {
49             error = "Der skete en fejl under sletning <BR>";
50             error += e.getMessage();
51         }
52     }
53
54     request.setAttribute("commodity", com);
55
56     request.setAttribute("done", success);
57     request.setAttribute("error", error);
58
59     RequestDispatcher dispatcher = getServletContext()
60         .getRequestDispatcher("/commodity_confirm_delete_boundary.jsp");
61     dispatcher.forward(request, response);
62
63 }
64 }
```

### CommodityEditController

```
1  package admin.controller;
2
3  import java.io.IOException;
4
5  import javax.servlet.RequestDispatcher;
6  import javax.servlet.ServletException;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10 import admin.data.CommodityDTO;
11 import admin.data.DALException;
12 import admin.data.UserType;
13
14 public class CommodityEditController extends AbstractController {
15
16     private static final long serialVersionUID = 1L;
17
18     @Override
19     protected UserType requiredAccessLevel() {
20         return UserType.PHARMACIST;
21     }
22
23     @Override
24     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
        throws ServletException, IOException{
25
26         int iNewId = 0;
27         boolean isNew = false;
28         boolean anyError = false;
29         String majorError = "";
30         String idError = "";
31         String nameError = "";
32         String supError = "";
33         String sNewId = "";
34         String newName = "";
```

```
36     String newSupplier = "";
37     CommodityDTO commodity = null;
38
39     String sComId = request.getParameter("id");
40     int iOldId = 0;
41
42     if (sComId != null && sComId.equals("new")) {
43         isNew = true;
44     } else {
45         try {
46             iOldId = Integer.parseInt(sComId);
47         } catch (NumberFormatException e) {
48             majorError = "RåvareId er ugyldigt";
49             anyError = true;
50         }
51     }
52
53     if (isNew) {
54         commodity = new CommodityDTO();
55     } else {
56         try {
57             commodity = commodities.getCommodity(iOldId);
58             sNewId = sComId;
59         } catch (DALEException e1) {
60             majorError = "Der findes ingen råvare med det angivne id";
61             anyError = true;
62         }
63     }
64
65     if (! anyError && request.getMethod().equals("POST")) {
66         sNewId = request.getParameter("newId");
67         newName = request.getParameter("newName");
68         newSupplier = request.getParameter("newSupplier");
69
70         try {
71             iNewId = Integer.parseInt(sNewId);
72             commodity.setCommodity_id(iNewId);
73         } catch (NumberFormatException e) {
74             idError = "Det indtastede råvare id er ikke et tal";
75             anyError = true;
76         } catch (DALEException e) {
77             idError = e.getMessage();
78             anyError = true;
79         }
80
81         try {
82             commodity.setCommodity_name(newName);
83         } catch (DALEException e) {
84             nameError = e.getMessage();
85             anyError = true;
86         }
87
88         try {
89             commodity.setSupplier(newSupplier);
90         } catch (DALEException e) {
```

```
91         supError = e.getMessage();
92         anyError = true;
93     }
94
95     if (!anyError) {
96         if (isNew) {
97             try {
98                 commodities.createCommodity(commodity);
99             } catch (DAException e) {
100                 idError = e.getMessage();
101                 anyError = true;
102             }
103         } else if (iOldId == iNewId) {
104             try {
105                 commodities.updateCommodity(commodity);
106             } catch (DAException e) {
107                 idError = e.getMessage();
108                 anyError = true;
109             }
110         } else {
111             try {
112                 commodities.createCommodity(commodity);
113             } try {
114                 commodities.deleteCommodity(iOldId);
115             } catch (DAException e) {
116                 idError = "Råvare id kunne ikke ændres <BR>";
117                 idError += e.getMessage();
118                 anyError = true;
119                 commodities.deleteCommodity(iNewId);
120             }
121         } catch (DAException e) {
122             idError = e.getMessage();
123             anyError = true;
124         }
125     }
126 }
127
128 }
129
130 request.setAttribute("majorError", majorError);
131 request.setAttribute("idError", idError);
132 request.setAttribute("nameError", nameError);
133 request.setAttribute("supError", supError);
134
135 request.setAttribute("create", isNew);
136 request.setAttribute("complete", !anyError
137     && request.getMethod().equals("POST"));
138
139 request.setAttribute("newId", sNewId);
140 request.setAttribute("commodity", commodity);
141
142
143 RequestDispatcher dispatcher = getServletContext()
144     .getRequestDispatcher("/commodity_edit_boundary.jsp");
145 dispatcher.forward(request, response);
```

```
146
147
148     }
149 }

LoginController

1 package admin.controller;
2
3 import java.io.IOException;
4
5 import javax.servlet.*;
6 import javax.servlet.http.*;
7
8 import admin.data.IUsersReadOnly;
9 import admin.data.UserType;
10
11 public class LoginController extends AbstractController{
12
13     private static final long serialVersionUID = 1L;
14
15     @Override
16     protected UserType requiredAccessLevel() {
17         return null;
18     }
19
20     @Override
21     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
        throws ServletException, IOException{
22
23
24         String loginError = "";
25
26         String redirect = request.getParameter("redirect");
27         if (redirect == null)
28             redirect = "/mainmenu";
29
30         String logout = request.getParameter("logout");
31         if (logout != null && logout.toLowerCase().equals("true")){
32             userSession.logout();
33         }
34
35         String context = request.getContextPath();
36         if (userSession.isLoggedIn()){
37             response.sendRedirect(context + redirect);
38             return;
39         }
40
41         String uid = request.getParameter("userId");
42         String pword = request.getParameter("password");
43         if (uid != null && pword != null)
44             try {
45                 int iUid = Integer.parseInt(uid);
46                 if (!userSession.login(iUid, pword)){
47                     loginError = "Det indtastede bruger id og kodeord er ikke
                        korrekt";
```

```

48         } else if (!(userSession.accessLevel() < UserType.INACTIVE.
49             ordinal())){
50             loginError = "Denne bruger er deaktiveret";
51             userSession.logout();
52         } else {
53             response.sendRedirect(context + redirect);
54             return;
55         } catch (Exception e){
56             loginError = "Bruger ID er ikke et tal";
57         }
58     else {
59         uid = "";
60         pword = "";
61     }
62
63     // TODO fjern dette
64     IUsersReadOnly users = super.users;
65     request.setAttribute("users", users);
66     // ^^ er kun til test
67
68     request.setAttribute("error", loginError);
69     request.setAttribute("userId", uid);
70     request.setAttribute("password", pword);
71     request.setAttribute("redirect", redirect);
72
73
74     RequestDispatcher dispatcher =
75         getServletContext().getRequestDispatcher("/login_boundary.jsp");
76     dispatcher.forward(request, response);
77 }
78
79 }

```

### MainMenuController

```

1  package admin.controller;
2
3  import java.io.IOException;
4  import java.util.ArrayList;
5
6  import javax.servlet.*;
7  import javax.servlet.http.*;
8
9  import admin.data.MenuOption;
10 import admin.data.UserType;
11
12 public class MainMenuController extends AbstractController {
13
14     private static final long serialVersionUID = 1L;
15
16     @Override
17     protected UserType requiredAccessLevel() {
18         return null;
19     }
20

```

```

21  @Override
22  public void doRequest(HttpServletRequest request, HttpServletResponse
      response)
23      throws ServletException, IOException{
24
25
26      ArrayList<MenuOption> menuChoices = new ArrayList<MenuOption>();
27      menuChoices.add( new MenuOption("Skift kodeord", "password_change") );
28      ;
29      //menuChoices.add( new MenuOption("Test", "test") );
30      menuChoices.add( new MenuOption("Log ud", "login?logout=true") );
31      if (userSession.isForeman()){
32          menuChoices.add(1, new MenuOption( "Produktbatchadministration", "
              productBatch_admin" ));
33          menuChoices.add(1, new MenuOption( "Råvarebatchadministration", "
              commodityBatch_admin" ));
34      }
35      if (userSession.isPharmacist()){
36          menuChoices.add(1, new MenuOption( "Receptadministration", "
              prescription_admin" ));
37          menuChoices.add(1, new MenuOption( "Råvareadministration", "
              commodity_admin" ));
38      }
39      if (userSession.isAdmin()){
40          menuChoices.add(1, new MenuOption( "Brugeradministration", "
              user_admin" ));
41      }
42      request.setAttribute("Menulist", menuChoices);
43      RequestDispatcher dispatcher =
44          getServletContext().getRequestDispatcher("/mainmenu_boundary.jsp"
              );
45      dispatcher.forward(request, response);
46
47  }
48  }

```

### PasswordChangeController

```

1  package admin.controller;
2
3  import java.io.IOException;
4
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7
8  import admin.data.DALException;
9  import admin.data.UserDTO;
10 import admin.data.UserType;
11
12 /**
13  * Servlet implementation class TestController
14  */
15 public class PasswordChangeController extends AbstractController {
16     private static final long serialVersionUID = 1L;
17

```

```
18  @Override
19  protected UserType requiredAccessLevel() {
20      return null;
21  }
22
23  @Override
24  public void doRequest(HttpServletRequest request, HttpServletResponse
      response)
25      throws ServletException, IOException{
26
27      String pword = request.getParameter("password");
28      String newPword1 = request.getParameter("newPword1");
29      String newPword2 = request.getParameter("newPword2");
30      String pwError = null;
31      String npwError = null;
32      UserDTO opr = null;
33      boolean input = false;
34
35      if (pword != null) {
36          input = true;
37          if (!pword.equals("")) {
38              int id = userSession.getId();
39              try {
40                  opr = users.getUser(id);
41                  if (!opr.getPassword().equals(pword)) {
42                      pwError = "Det indtastede password er ikke korrekt";
43                  }
44              } catch (DAException e) {
45                  response.sendRedirect("login");
46                  return;
47              }
48          } else {
49              pwError = "Du skal indtaste dit nuværende kodeord";
50          }
51      }
52  }
53
54  if (newPword1 != null || newPword2 != null) {
55      input = true;
56      if (newPword1.equals(newPword2)) {
57          npwError = UserDTO.chkPassWithMsg(newPword1);
58      } else {
59          npwError = "De to nye passwords er ikke ens";
60      }
61  }
62  }
63
64  boolean success = false;
65  if (input && pwError == null && npwError == null) {
66      try {
67          opr.setPassword(newPword1);
68          users.updateUser(opr);
69          success = true;
70          userSession.login(userSession.getId(), newPword1);
71      } catch (DAException e) {
```

```
72     }
73 }
74
75 if (npwError != null)
76     npwError = npwError.replace("\n", "<BR>");
77 request.setAttribute("success", success);
78 request.setAttribute("newPword1", newPword1);
79 request.setAttribute("newPword2", newPword2);
80 request.setAttribute("password", pword);
81 request.setAttribute("pwError", pwError);
82 request.setAttribute("npwError", npwError);
83
84 RequestDispatcher dispatcher = getServletContext()
85     .getRequestDispatcher("/password_change_boundary.jsp");
86 dispatcher.forward(request, response);
87 }
88
89 }
```

### PrescriptionAdminController

```
1 package admin.controller;
2
3 import java.io.IOException;
4 import java.util.List;
5
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 import admin.data.PrescriptionDTO;
12 import admin.data.UserType;
13
14 public class PrescriptionAdminController extends AbstractController {
15
16     private static final long serialVersionUID = 1L;
17
18     @Override
19     protected UserType requiredAccessLevel() {
20         return UserType.PHARMACIST;
21     }
22
23     @Override
24     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
        throws ServletException, IOException{
25
26         try {
27             List<PrescriptionDTO> prescriptionList = prescriptions.
                getPrescriptionList();
28             request.setAttribute("prescriptionList", prescriptionList);
29
30         } catch (Exception e){
31
32
33 }
```



```
34
35     RequestDispatcher dispatcher =
36         getServletContext().getRequestDispatcher("/
37             prescription_admin_boundary.jsp");
38     dispatcher.forward(request, response);
39 }
40
41
42 }

PrescriptionConfirmDeleteController

1  package admin.controller;
2
3  import java.io.IOException;
4  import java.util.List;
5
6  import javax.servlet.RequestDispatcher;
7  import javax.servlet.ServletException;
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10
11  import admin.data.DAException;
12  import admin.data.PrescriptionCompDTO;
13  import admin.data.PrescriptionDTO;
14  import admin.data.UserType;
15
16
17  public class PrescriptionConfirmDeleteController extends
18      AbstractController {
19
20      private static final long serialVersionUID = 1L;
21
22      @Override
23      protected UserType requiredAccessLevel() {
24          return UserType.PHARMACIST;
25      }
26
27      @Override
28      public void doRequest(HttpServletRequest request, HttpServletResponse
29          response)
30          throws ServletException, IOException{
31
32          String sPresId = request.getParameter("id");
33          int iPresId = 0;
34          String sError = "";
35          boolean bError = false;
36
37          PrescriptionDTO prescription = null;
38          List<PrescriptionCompDTO> components = null;
39
40          // try and parse the id given in the url.
41          // if the id equals "new" then we're creating a new prescription
42          try {
43              iPresId = Integer.parseInt(sPresId);
```

```

42     prescription = prescriptions.getPrescription(iPresId);
43     components = presComps.getComponentList(iPresId);
44 } catch (NumberFormatException e) {
45     sError = "ReceptID er ugyldigt";
46     bError = true;
47 } catch (DAException e) {
48     sError = e.getMessage();
49     bError = true;
50 }
51
52 if ( !bError && request.getMethod().equals("POST") ) {
53     try {
54         prescriptions.deletePrescription(iPresId);
55         presComps.deletePrescriptionComp(iPresId);
56     } catch (DAException e) {
57         sError = e.getMessage();
58         bError = true;
59     }
60 }
61
62 request.setAttribute("complete", !bError && request.getMethod().
    equals("POST"));
63 request.setAttribute("error", sError);
64
65 request.setAttribute("prescription", prescription);
66 request.setAttribute("components", components);
67
68 RequestDispatcher dispatcher = getServletContext()
69     .getRequestDispatcher("/prescription_confirm_delete_boundary.jsp"
70     );
71 dispatcher.forward(request, response);
72 }
73 }

```

### PrescriptionEditController

```

1  package admin.controller;
2
3  import java.io.IOException;
4  import java.util.ArrayList;
5  import java.util.List;
6
7  import javax.servlet.RequestDispatcher;
8  import javax.servlet.ServletException;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 import admin.data.DAException;
13 import admin.data.PrescriptionCompDTO;
14 import admin.data.PrescriptionDTO;
15 import admin.data.UserType;
16
17 public class PrescriptionEditController extends AbstractController {
18
19     private static final long serialVersionUID = 1L;

```

```
20
21 @Override
22 protected UserType requiredAccessLevel() {
23     return UserType.PHARMACIST;
24 }
25
26 @Override
27 public void doRequest(HttpServletRequest request,
28     HttpServletResponse response) throws ServletException, IOException
29 {
30     boolean isNew = false;
31     boolean anyError = false;
32     boolean submit = true;
33
34     String majorError = "";
35     String idError = "";
36     String nameError = "";
37
38     PrescriptionDTO prescription = null;
39     List<PrescriptionCompDTO> components = null;
40
41     String sNewId = "";
42     String newName = "";
43
44     String sPresId = request.getParameter("id");
45     int iOldId = 0;
46
47     // try and parse the id given in the url.
48     // if the id equals "new" then we're creating a new prescription
49     if (sPresId != null && sPresId.equals("new")) {
50         isNew = true;
51     } else {
52         try {
53             iOldId = Integer.parseInt(sPresId);
54         } catch (NumberFormatException e) {
55             majorError = "ReceptID er ugyldigt";
56             anyError = true;
57         }
58     }
59
60     // if the request is a GET request we know that no data has been
61     // posted.
62     if (request.getMethod().equals("GET")) {
63         submit = false;
64         // if it's a new prescription we just create a new PrescriptionDTO
65         // and be done with it
66         if (isNew) {
67             prescription = new PrescriptionDTO();
68         } else {
69             // if it's not a new prescription we have to get the info on it
70             // from the database
71             try {
72                 prescription = prescriptions.getPrescription(iOldId);
73                 sNewId = sPresId;
```

```
73         } catch (DAException e1) {
74             majorError = "Der findes ingen recept med det angivne id";
75             anyError = true;
76         }
77         // and we have to fetch the list of components associated with
78         // the prescription
79         try {
80             components = presComps.getComponentList(iOldId);
81         } catch (DAException e) {
82             System.err.println(e.getMessage());
83         }
84     }
85 } else {
86     // if the request is a POST request we need to check all the info
87     // submitted and maybe update the database if there are no errors
88     sNewId = request.getParameter("newId");
89     newName = request.getParameter("newName");
90
91     // check which button was pressed
92     // null means it was the regular submit button
93     // any other value means we have to modify the component list
94     String button = request.getParameter("button");
95     if (button != null)
96         submit = false;
97     else
98         button = "";
99
100    // check if the entered id is a number
101    int iNewId = 0;
102    try {
103        iNewId = Integer.parseInt(sNewId);
104    } catch (NumberFormatException e) {
105        iNewId = 0;
106        idError = "Det indtastede id er ikke et tal";
107        anyError = true;
108    }
109
110    try {
111        // Check if we're creating a new recipe or if we're editing an
112        // existing one and if so, get the info on it from the database.
113        if (isNew) {
114            prescription = new PrescriptionDTO();
115            // if we're creating a new prescription there is no old id,
116            // so we use the new one in its place
117            iOldId = iNewId;
118        } else {
119            prescription = prescriptions.getPrescription(iOldId);
120        }
121
122        // Check how long the currently displayed component list is
123        String sCompCount = request.getParameter("compCount");
124        int iCompCount = 0;
125        try {
126            iCompCount = Integer.parseInt(sCompCount);
127        } catch (NumberFormatException e) {
```

```
128     }
129
130     // make the list of components from the data in the form (not
131     // taking the data from the database)
132     int i = 0;
133     components = new ArrayList<PrescriptionCompDTO>();
134     while (iCompCount > components.size()) {
135         // create a new component belonging to the current
136         // prescription
137         PrescriptionCompDTO component = new PrescriptionCompDTO();
138         try {
139             component.setPrescriptionId(iNewId);
140         } catch (Exception e) {
141             anyError = true;
142         }
143
144         // Get the component id from the form and try to insert it
145         // into the DIO.
146         String comIdError = "";
147         String sComId = request.getParameter("comId" + i);
148         try {
149             int iComId = Integer.parseInt(sComId);
150             component.setCommodityId(iComId);
151         } catch (Exception e) {
152             // Post an error message if it fails
153             anyError = true;
154             comIdError = e.getMessage() + "<BR>";
155         }
156
157         // Get the net weight from the form and try to insert it
158         // into the DIO.
159         String sNetto = request.getParameter("netto" + i);
160         try {
161             double dNetto = Double.parseDouble(sNetto);
162             component.setNomNetto(dNetto);
163         } catch (Exception e) {
164             // Post an error message if it fails
165             anyError = true;
166             request.setAttribute("comNetError" + i, e.getMessage());
167         }
168
169         // Get the tolerance from the form and try to insert it into
170         // the DIO.
171         String sTolerance = request.getParameter("tolerance" + i);
172         try {
173             double dTolerance = Double.parseDouble(sTolerance);
174             component.setTolerance(dTolerance);
175         } catch (Exception e) {
176             // Post an error message if it fails
177             anyError = true;
178             request.setAttribute("comTolError" + i, e.getMessage());
179         }
180
181         // before we add the component to the list we check if it
182         // has the same commodity id as any of the others
```

```
183     int j = 1;
184     for (PrescriptionCompDTO oldComp : components) {
185         if (oldComp.getCommodityId() == component
186             .getCommodityId()) {
187             comIdError += "Samme vareID som #" + j + "<BR>";
188         }
189         j++;
190     }
191     request.setAttribute("comIdError" + i, comIdError);
192
193     // finally add the component to the list and increment our
194     // counter
195     components.add(component);
196     i++;
197 }
198
199 // if the new button was pressed we have to add a component to
200 // our list
201 if (button.equals("new")) {
202     components.add(new PrescriptionCompDTO(iOldId, 0, 0, 0));
203 }
204
205 // if a delete button was pressed we have to remove an entry
206 // from the list corresponding to the number of the button
207 if (button.matches("delete[0-9]+")) {
208     String sDelete = button.substring(6);
209     try {
210         int iDelete = Integer.parseInt(sDelete);
211         components.remove(iDelete);
212     } catch (Exception e) {
213     }
214 }
215
216 // Try to insert the new id into our DTO
217 try {
218     prescription.setId(iNewId);
219 } catch (DALEException e) {
220     // Post an error message if it fails
221     idError = e.getMessage();
222     anyError = true;
223 }
224
225 // Try to insert the new name into our DTO
226 try {
227     prescription.setName(newName);
228 } catch (DALEException e) {
229     // Post an error message if it fails
230     nameError = e.getMessage();
231     anyError = true;
232 }
233
234 // if there were no errors so far and the submit button was
235 // pressed we should try to update the database with the new
236 // info
237 if (!anyError && submit) {
```

```
238 // if the new and old IDs are different we have to create a
239 // new entry and remove the old one. The IDs will be the
240 // same if we're creating a new prescription
241 if (iNewId != iOldId) {
242     prescriptions.createPrescription(prescription);
243     prescriptions.deletePrescription(iOldId);
244
245     // we now need to update all the components. We have to
246     // remove the old ones, since they all have the wrong
247     // prescription id and then insert the new ones
248     presComps.deletePrescriptionComp(iOldId);
249     for (PrescriptionCompDTO comp : components) {
250         presComps.createPrescriptionComp(comp);
251     }
252 } else if (isNew) {
253     // This is where we create a new prescription
254     prescriptions.createPrescription(prescription);
255
256     // now we need to add the components to the database
257     for (PrescriptionCompDTO comp : components) {
258         presComps.createPrescriptionComp(comp);
259     }
260 } else {
261     // This is where we update an existing one with no ID
262     // change
263     prescriptions.updatePrescription(prescription);
264
265     // we now need to update all the components. the easiest
266     // way to do that is to remove the old ones and insert
267     // the new ones
268     presComps.deletePrescriptionComp(iOldId);
269     for (PrescriptionCompDTO comp : components) {
270         presComps.createPrescriptionComp(comp);
271     }
272 }
273
274 }
275 } catch (DALEException e) {
276     idError = e.getMessage();
277     anyError = true;
278 }
279
280 }
281
282 request.setAttribute("complete", !anyError && submit);
283 request.setAttribute("create", isNew);
284
285 request.setAttribute("majorError", majorError);
286 request.setAttribute("idError", idError);
287 request.setAttribute("nameError", nameError);
288 request.setAttribute("newId", sNewId);
289 request.setAttribute("prescription", prescription);
290 request.setAttribute("components", components);
291
292 RequestDispatcher dispatcher = getServletContext()
```

```
293         .getRequestDispatcher("/prescription_edit_boundary.jsp");
294     dispatcher.forward(request, response);
295
296 }
297 }
```

### ProductBatchAdminController

```
1  package admin.controller;
2
3  import java.io.IOException;
4  import java.util.List;
5
6  import javax.servlet.RequestDispatcher;
7  import javax.servlet.ServletException;
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10
11 import admin.data.ProductBatchDTO;
12 import admin.data.UserType;
13
14 public class ProductBatchAdminController extends AbstractController {
15
16     private static final long serialVersionUID = 1L;
17
18     @Override
19     protected UserType requiredAccessLevel() {
20         return UserType.FOREMAN;
21     }
22
23     @Override
24     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
25         throws ServletException, IOException{
26
27         try {
28             List<ProductBatchDTO> productList = products.getAllProductBatches()
29                 ;
30             request.setAttribute("productList", productList);
31         } catch (Exception e){
32
33         }
34
35         RequestDispatcher dispatcher =
36             getServletContext().getRequestDispatcher("/
                productBatch_admin_boundary.jsp");
37         dispatcher.forward(request, response);
38     }
39 }
40 }
```

### ProductBatchConfirmDeleteController

```
1  package admin.controller;
2
```



```
3 import java.io.IOException;
4 import java.util.List;
5
6 import javax.servlet.RequestDispatcher;
7 import javax.servlet.ServletException;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 import admin.data.DALException;
12 import admin.data.ProductBatchCompDTO;
13 import admin.data.ProductBatchDTO;
14 import admin.data.UserType;
15
16 public class ProductBatchConfirmDeleteController extends
    AbstractController {
17
18     private static final long serialVersionUID = 1L;
19
20     @Override
21     protected UserType requiredAccessLevel() {
22         return UserType.FOREMAN;
23     }
24
25     @Override
26     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
        throws ServletException, IOException{
27
28         String sProdId = request.getParameter("id");
29         int iProdId = 0;
30         String sError = "";
31         boolean bError = false;
32
33         ProductBatchDTO product = null;
34         List<ProductBatchCompDTO> components = null;
35
36         // try and parse the id given in the url.
37         // if the id equals "new" then we're creating a new prescription
38         try {
39             iProdId = Integer.parseInt(sProdId);
40             product = products.getProductBatch(iProdId);
41             components = prodComps.getCertainProductBatchComps(iProdId);
42         } catch (NumberFormatException e) {
43             sError = "Produktbatch ID er ugyldigt";
44             bError = true;
45         } catch (DALException e) {
46             sError = e.getMessage();
47             bError = true;
48         }
49
50         if ( !bError && request.getMethod().equals("POST")) {
51             try {
52                 products.deleteBatch(iProdId);
53                 prodComps.deleteByBatchID(iProdId);
54             } catch (DALException e) {
```

```
56         sError = e.getMessage();
57         bError = true;
58     }
59 }
60
61 request.setAttribute("complete", !bError && request.getMethod().
    equals("POST"));
62 request.setAttribute("error", sError);
63
64 request.setAttribute("product", product);
65 request.setAttribute("components", components);
66
67 RequestDispatcher dispatcher = getServletContext().
68     .getRequestDispatcher("/prescription_confirm_delete_boundary.jsp")
69     );
69 dispatcher.forward(request, response);
70
71 }
72 }
```

### ProductBatchEditController

```
1  package admin.controller;
2
3  import java.io.IOException;
4  import java.util.List;
5
6  import javax.servlet.RequestDispatcher;
7  import javax.servlet.ServletException;
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10
11 import admin.data.DAException;
12 import admin.data.ProductBatchCompDTO;
13 import admin.data.ProductBatchDTO;
14 import admin.data.StatusType;
15 import admin.data.UserType;
16
17 public class ProductBatchEditController extends AbstractController {
18
19     private static final long serialVersionUID = 1L;
20
21     @Override
22     protected UserType requiredAccessLevel() {
23         return UserType.FOREMAN;
24     }
25
26     @Override
27     public void doRequest(HttpServletRequest request,
28         HttpServletResponse response) throws ServletException, IOException
29     {
30         boolean isNew = false;
31         boolean anyError = false;
32
33         String majorError = "";
```

```
34 String idError = "";
35 String presIdError = "";
36 String statusError = "";
37 String compError = "";
38 String userError = "";
39
40 ProductBatchDTO product = null;
41 List<ProductBatchCompDTO> components = null;
42
43 String sNewId = "";
44 String sNewPresId = "";
45 String sNewStatus = "";
46 String sNewUserId = "";
47
48 String sOldId = request.getParameter("id");
49 int iOldId = 0;
50
51 // try and parse the id given in the url.
52 // if the id equals "new" then we're creating a new prescription
53 if (sOldId != null && sOldId.equals("new")) {
54     isNew = true;
55 } else {
56     try {
57         iOldId = Integer.parseInt(sOldId);
58     } catch (NumberFormatException e) {
59         majorError = "ProduktbatchID er ugyldigt";
60         anyError = true;
61     }
62 }
63
64 // if the request is a GET request we know that no data has been
65 // posted.
66 if (request.getMethod().equals("GET")) {
67     // if it's a new prescription we just create a new PrescriptionDTO
68     // and be done with it
69     if (isNew) {
70         product = new ProductBatchDTO();
71         sNewPresId = "";
72         sNewStatus = "";
73     } else {
74         // if it's not a new prescription we have to get the info on it
75         // from the database
76         try {
77             product = products.getProductBatch(iOldId);
78             sNewId = sOldId;
79             sNewPresId = "" + product.getPrescriptionId();
80             sNewStatus = product.getStatus().name();
81             sNewUserId = "" + product.getUserId();
82         } catch (DALException e) {
83             majorError = "Der findes intet produktbatch med det angivne id";
84             ;
85             anyError = true;
86         }
87     }
88     // and we have to fetch the list of components associated with
89     // the prescription
```

```
87         try {
88             components = prodComps.getCertainProductBatchComps(iOldId);
89         } catch (DALException e) {
90             compError = e.getMessage();
91         }
92     }
93 } else {
94     // if the request is a POST request we need to check all the info
95     // submitted and maybe update the database if there are no errors
96     sNewId = request.getParameter("newId");
97     sNewPresId = request.getParameter("newPresId");
98     sNewStatus = request.getParameter("newStatus");
99     sNewUserId = request.getParameter("newUserId");
100
101     // check if the entered id is a number
102     int iNewId = 0;
103     try {
104         iNewId = Integer.parseInt(sNewId);
105     } catch (NumberFormatException e) {
106         iNewId = 0;
107         idError = "Det indtastede id er ikke et tal";
108         anyError = true;
109     }
110
111     try {
112         // Check if we're creating a new recipe or if we're editing an
113         // existing one and if so, get the info on it from the database.
114         if (isNew) {
115             product = new ProductBatchDTO();
116             // if we're creating a new prescription there is no old id,
117             // so we use the new one in its place
118             iOldId = iNewId;
119         } else {
120             product = products.getProductBatch(iOldId);
121
122             // Get the list of components from the database
123             components = prodComps.getCertainProductBatchComps(iOldId);
124         }
125
126         // Try to insert the new id into our DTO
127         try {
128             product.setPb_id(iNewId);
129         } catch (DALException e) {
130             // Post an error message if it fails
131             idError = e.getMessage();
132             anyError = true;
133         }
134
135         // Try to insert the new prescription id into our DTO
136         try {
137             int iNewPresId = Integer.parseInt(sNewPresId);
138             product.setPrescription_id(iNewPresId);
139         } catch (NumberFormatException e) {
140             // Post an error message if we can't convert the input to an
141             // int
```

```
142     presIdError = "det indtastede recpetId er ikke et tal";
143     anyError = true;
144 } catch (DALErrorException e) {
145     // Post an error message if we can't set the id
146     presIdError = e.getMessage();
147     anyError = true;
148 }
149
150 // Try to insert the new user id into our DIO
151 try {
152     int iNewUserId = Integer.parseInt(sNewUserId);
153     product.setUserId(iNewUserId);
154 } catch (NumberFormatException e) {
155     // Post an error message if we can't convert the input to an
156     // int
157     presIdError = "det indtastede bruger id er ikke et tal";
158     anyError = true;
159 } catch (DALErrorException e) {
160     // Post an error message if we can't set the id
161     userError = e.getMessage();
162     anyError = true;
163 }
164
165 // Try to insert the new status into our DIO
166 try {
167     StatusType status = StatusType.valueOf(sNewStatus);
168     product.setStatus(status);
169 } catch (DALErrorException e) {
170     // Post an error message if we can't set the status
171     statusError = e.getMessage();
172     anyError = true;
173 } catch (Exception e) {
174     // Post an error message if something else went wrong
175     statusError = "der skete en fejl med den nye status";
176     statusError += e.getMessage();
177     anyError = true;
178 }
179
180 // if there were no errors so far and the submit button was
181 // pressed we should try to update the database with the new
182 // info
183 if (!anyError) {
184     // if the new and old IDs are different we have to create a
185     // new entry and remove the old one. The IDs will be the
186     // same if we're creating a new prescription
187     if (iNewId != iOldId) {
188         products.createProductBatch(product);
189         try {
190             products.deleteBatch(iOldId);
191         } catch (DALErrorException e) {
192             products.deleteBatch(product.getPbId());
193         }
194         // we now need to update all the components. We have to
195         // remove the old ones, since they all have the wrong
196         // prescription id and then insert the new ones
```

```
197         prodComps.deleteByBatchID(iOldId);
198         for (ProductBatchCompDTO comp : components) {
199             prodComps.createProductBatchComp(comp);
200         }
201     } else if (isNew) {
202         // This is where we create a new prescription
203         products.createProductBatch(product);
204
205         // we don't add any components to the database since
206         // they only get added by the ASE
207     } else {
208         // This is where we update an existing one with no ID
209         // change
210         products.updateProductBatch(product);
211
212         // we don't change any of the components since they are
213         // still connected to the same batch ID
214     }
215     //product = products.getProductBatch(product.getPbId());
216 }
217 } catch (DALException e) {
218     idError = e.getMessage();
219     anyError = true;
220 }
221
222 }
223
224 request.setAttribute("complete", !anyError && request.getMethod().
    equals("POST"));
225 request.setAttribute("create", isNew);
226
227 request.setAttribute("majorError", majorError);
228 request.setAttribute("idError", idError);
229 request.setAttribute("presIdError", presIdError);
230 request.setAttribute("statusError", statusError);
231 request.setAttribute("compError", compError);
232 request.setAttribute("userError", userError);
233
234 request.setAttribute("newId", sNewId);
235 request.setAttribute("newPresId", sNewPresId);
236 request.setAttribute("newStatus", sNewStatus);
237 request.setAttribute("newUserId", sNewUserId);
238
239 request.setAttribute("product", product);
240 request.setAttribute("components", components);
241
242 RequestDispatcher dispatcher = getServletContext().
243     .getRequestDispatcher("/productBatch_edit_boundary.jsp");
244 dispatcher.forward(request, response);
245
246 }
247
248 }
```

**ProductBatchPrintController**

```
1 package admin.controller;
2
3 import java.io.IOException;
4 import java.sql.Date;
5 import java.util.ArrayList;
6 import java.util.Calendar;
7 import java.util.List;
8
9 import javax.servlet.RequestDispatcher;
10 import javax.servlet.ServletException;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import admin.data.DALException;
15 import admin.data.PrescriptionCompDTO;
16 import admin.data.ProductBatchCompDTO;
17 import admin.data.ProductBatchDTO;
18 import admin.data.ProductListItem;
19 import admin.data.UserType;
20
21 public class ProductBatchPrintController extends AbstractController {
22
23     private static final long serialVersionUID = 1L;
24
25     @Override
26     protected UserType requiredAccessLevel() {
27         return null;
28     }
29
30     @Override
31     public void doRequest(HttpServletRequest request,
32         HttpServletResponse response) throws ServletException, IOException
33     {
34         int iId = 0;
35         double iSumTara = 0;
36         double iSumNetto = 0;
37         String sId = request.getParameter("id");
38
39         String sError = "";
40
41         ProductBatchDTO product = null;
42         List<PrescriptionCompDTO> preCompList = null;
43         List<ProductListItem> itemList = new ArrayList<ProductListItem>();
44         String printDate = new Date(Calendar.getInstance().getTime().getTime
45             ())
46             .toString();
47
48         try {
49             iId = Integer.parseInt(sId);
50         } catch (NumberFormatException e) {
51             sError = "ProduktbatchID er ugyldigt";
52         }
53
54         try {
```

```
54     product = products.getProductBatch(iId);
55     preCompList = presComps.getComponentList(product
56         .getPrescriptionId());
57 } catch (DAException e) {
58     sError = "ProduktbatchID findes ikke i databasen";
59 }
60
61 if (preCompList != null) {
62     for (PrescriptionCompDTO preComp : preCompList) {
63         ProductListItem item = new ProductListItem();
64         item.commodityId = preComp.getCommodityId();
65         try {
66             item.commodityName = commodities.getCommodity(
67                 preComp.getCommodityId()).getComName();
68         } catch (DAException e) {
69         }
70
71         item.ammount = preComp.getNomNetto();
72         item.tolerance = preComp.getTolerance();
73         try {
74             ProductBatchCompDTO proComp = prodComps.getCompByComId(iId,
75                 item.commodityId);
76             item.tara = proComp.getTara();
77             iSumTara += item.tara;
78             item.netto = proComp.getNetto();
79             iSumNetto += item.netto;
80             item.commodityBatch = proComp.getCommoditybatch_id();
81             item.operator = users.getUser(proComp.getUser_id())
82                 .getIni();
83         } catch (DAException e) {
84         }
85         itemList.add(item);
86     }
87 }
88
89 String sSumTara = "" + iSumTara;
90 String sSumNetto = "" + iSumNetto;
91 try {
92     sSumTara = sSumTara.substring(0, sSumTara.indexOf(".") + 4);
93 } catch (Exception e) {}
94 try {
95     sSumNetto = sSumNetto.substring(0, sSumNetto.indexOf(".") + 4);
96 } catch (Exception e) {}
97
98 request.setAttribute("error", sError);
99
100 request.setAttribute("date", printDate);
101 request.setAttribute("sumTara", sSumTara);
102 request.setAttribute("sumNetto", sSumNetto);
103
104 request.setAttribute("product", product);
105 request.setAttribute("components", itemList);
106
107 RequestDispatcher dispatcher = getServletContext()
108     .getRequestDispatcher("/productBatch_print_boundary.jsp");
```



```
109     dispatcher.forward(request, response);
110
111 }
112
113 }

TestController

1 package admin.controller;
2
3 import admin.data.UserType;
4
5 import java.io.IOException;
6 import javax.servlet.*;
7 import javax.servlet.http.*;
8
9
10 /**
11  * Servlet implementation class TestController
12  */
13 public class TestController extends AbstractController {
14     private static final long serialVersionUID = 1L;
15
16     @Override
17     protected UserType requiredAccessLevel() {
18         return null;
19     }
20
21     @Override
22     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
23         throws ServletException, IOException{
24
25         String error = "";
26
27         String sBruto = request.getParameter("bruto");
28         String sTara = request.getParameter("tara");
29
30         double dBruto = 0;
31         double dTara = 0;
32         double dNetto = 0;
33
34         if (sBruto != null || sTara != null){
35             if (sBruto != null && !sBruto.equals("")){
36                 sBruto = sBruto.replace(',', '.', '.');
37                 try{
38                     dBruto = Double.parseDouble(sBruto);
39                 } catch (NumberFormatException e){
40                     error += "Den intastede brutoværdi kan ikke læses som et tal<br
                        >";
41                 }
42             } else
43                 sBruto = "";
44
45             if (sTara != null && !sTara.equals("")){
46                 sTara = sTara.replace(',', '.', '.');
```

```
47         try {
48             dTara = Double.parseDouble(sTara);
49         } catch (NumberFormatException e) {
50             error += "Den intastede taraværdi kan ikke læses som et tal";
51         }
52     } else
53         sTara = "";
54 }
55
56 dNetto = dBruto - dTara;
57
58 request.setAttribute("error", error);
59 request.setAttribute("bruto", sBruto);
60 request.setAttribute("tara", sTara);
61 request.setAttribute("netto", dNetto);
62
63 RequestDispatcher dispatcher =
64     getServletContext().getRequestDispatcher("/test_boundary.jsp");
65 dispatcher.forward(request, response);
66
67 }
68 }
69
70 }
```

### UserAdminController

```
1  package admin.controller;
2
3  import java.io.IOException;
4  import java.util.ArrayList;
5  import java.util.List;
6
7  import javax.servlet.*;
8  import javax.servlet.http.*;
9
10 import admin.data.IUsersReadOnly;
11 import admin.data.UserDTO;
12 import admin.data.UserInfo;
13 import admin.data.UserType;
14
15 /**
16  * Servlet implementation class UserAdminController
17  */
18 public class UserAdminController extends AbstractController {
19     private static final long serialVersionUID = 1L;
20     IUsersReadOnly userData;
21
22     public void init(ServletConfig config) throws ServletException {
23         super.init(config);
24         this.userData = super.users;
25     }
26
27     @Override
28     protected UserType requiredAccessLevel() {
29         return UserType.ADMIN;
```

```

30     }
31
32     @Override
33     public void doRequest(HttpServletRequest request, HttpServletResponse
        response)
34         throws ServletException, IOException{
35
36
37         try {
38             List<UserDTO> users = userData.getUserList();
39             List<UserInfo> userInfos = new ArrayList<UserInfo>();
40             for (UserDTO user : users){
41                 UserInfo userInfo = new UserInfo(user);
42                 userInfos.add(userInfo);
43             }
44
45             request.setAttribute("userlist", userInfos);
46
47         } catch (Exception e){
48
49         }
50
51         RequestDispatcher dispatcher =
52             getServletContext().getRequestDispatcher("/user_admin_boundary.
                jsp");
53         dispatcher.forward(request, response);
54
55     }
56
57 }

```

#### UserConfirmDeleteController

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Error><Code>NoSuchKey</Code><Message>The specified key does not exist.</
    Message><Key>5339631428048190250001ec/53a331b750d221e952a561d0</Key><
    RequestId>E673F5F813C57C37</RequestId><HostId>
    l6mWEoiv35iJWp3oUOfotvOuJ0pUWuR14eX0Di0aQkS52VL9a5bMItdAgG/sRX</
    HostId></Error>

```

#### UserEditController

```

1 package admin.controller;
2
3 import java.io.*;
4
5 import javax.servlet.*;
6 import javax.servlet.http.*;
7
8 import admin.data.DALException;
9 import admin.data.UserDTO;
10 import admin.data.UserInfo;
11 import admin.data.UserType;
12
13 public class UserEditController extends AbstractController {
14     private static final long serialVersionUID = 1L;

```

```
15
16 @Override
17 protected UserType requiredAccessLevel() {
18     return UserType.ADMIN;
19 }
20
21 @Override
22 public void doRequest(HttpServletRequest request, HttpServletResponse
    response)
23     throws ServletException, IOException{
24
25     String sNewId = "";
26     String sNewAccess = "";
27     UserInfo info = new UserInfo();
28     UserInfo old = new UserInfo();
29     UserInfo error = new UserInfo();
30     String idError = "";
31     String accessError = "";
32     String majorError = "";
33     boolean anyError = false;
34     boolean isNewUser = false;
35     String password = "";
36
37     String sUserId = request.getParameter("id");
38     int iUserId = 0;
39
40     if (sUserId != null && sUserId.equals("new")) {
41         isNewUser = true;
42     } else {
43         try {
44             iUserId = Integer.parseInt(sUserId);
45         } catch (NumberFormatException e) {
46             majorError = "Bruger id er ugyldigt";
47             anyError = true;
48         }
49     }
50
51     if (request.getMethod().equals("GET")) {
52         if (isNewUser) {
53             info = new UserInfo();
54             info.ini = "";
55             info.name = "";
56             info.cpr = "";
57         } else {
58             try {
59                 UserDTO user = users.getUser(iUserId);
60                 sNewId = sUserId;
61                 info = new UserInfo(user);
62                 request.setAttribute("info", info);
63             } catch (DAException e1) {
64                 majorError = "Der findes ingen bruger med det angivne id";
65                 anyError = true;
66             }
67         }
68     } else {
```

```
69     info = new UserInfo();
70
71     sNewId = request.getParameter("newId");
72     info.ini = request.getParameter("newIni");
73     info.name = request.getParameter("newName");
74     info.cpr = request.getParameter("newCPR");
75     sNewAccess = request.getParameter("newAccess");
76
77     try {
78         info.id = Integer.parseInt(sNewId);
79     } catch (NumberFormatException e) {
80         info.id = 0;
81         idError = "Det indtastede bruger id er ikke et tal";
82         anyError = true;
83     }
84
85     UserDTO user;
86     try {
87         if (isNewUser) {
88             password = UserDTO.generatePassword();
89             user = new UserDTO(1, "AA", "AA", "0000000000", password);
90         } else {
91             user = users.getUser(iUserId);
92         }
93
94         try {
95             user.setUserId(info.id);
96         } catch (DALErrorException e) {
97             idError = e.getMessage();
98             anyError = true;
99         }
100
101         try {
102             user.setIni(info.ini);
103         } catch (DALErrorException e) {
104             error.ini = e.getMessage();
105             anyError = true;
106         }
107
108         try {
109             user.setUsername(info.name);
110         } catch (DALErrorException e) {
111             error.name = e.getMessage();
112             anyError = true;
113         }
114
115         try {
116             user.setCpr(info.cpr);
117         } catch (DALErrorException e) {
118             error.cpr = e.getMessage();
119             anyError = true;
120         }
121
122         try {
123             info.access = UserType.valueOf(sNewAccess);
```

```
124     user.setAccessLevel(info.access);
125 } catch (IllegalArgumentException e) {
126     info.access = UserType.OPERATOR;
127     accessError = "Der skete en fejl med brugertypen";
128     anyError = true;
129 } catch (NullPointerException e) {
130     info.access = UserType.OPERATOR;
131     accessError = "Der blev ikke valgt nogen brugertype";
132     anyError = true;
133 }
134
135 if (!anyError) {
136     if (isNewUser) {
137         try {
138             users.createUser(user);
139         } catch (DAException e) {
140             idError = e.getMessage();
141             anyError = true;
142         }
143     } else {
144         old = new UserInfo(users.getUser(iUserId));
145         if (iUserId == info.id) {
146             try {
147                 users.updateUser(user);
148             } catch (DAException e) {
149                 idError = e.getMessage();
150                 anyError = true;
151             }
152         } else {
153             try {
154                 users.createUser(user);
155                 users.deleteUser(iUserId);
156             } catch (DAException e) {
157                 idError = e.getMessage();
158                 anyError = true;
159             }
160         }
161     }
162 } else if (iUserId != info.id) {
163     try {
164         users.getUser(info.id);
165         idError = "Dette id er optaget";
166     } catch (DAException e) {
167     }
168 }
169 } catch (Exception e) {
170     //majorError = "Brugeren med det givne id findes ikke længere";
171     majorError = e.getMessage();
172     e.printStackTrace();
173 }
174 }
175
176 request.setAttribute("majorError", majorError);
177 request.setAttribute("error", error);
178 request.setAttribute("idError", idError);
```

```
179     request.setAttribute("accessError", accessError);
180     request.setAttribute("complete", !anyError
181         && request.getMethod().equals("POST"));
182     request.setAttribute("create", isNewUser);
183
184     request.setAttribute("newId", sNewId);
185     request.setAttribute("newPw", password);
186     request.setAttribute("oldId", sUserId);
187     request.setAttribute("info", info);
188     request.setAttribute("old", old);
189
190     RequestDispatcher dispatcher = getServletContext()
191         .getRequestDispatcher("/user_edit_boundary.jsp");
192     dispatcher.forward(request, response);
193
194 }
195 }
```

### UserSession

```
1  package admin.controller;
2
3  import admin.data.IUsersReadOnly;
4  import admin.data.UserDTO;
5  import admin.data.UserType;
6
7  public class UserSession {
8      private int userId;
9      private String password;
10     private IUsersReadOnly data;
11
12     public void init(IUsersReadOnly data){
13         this.data = data;
14     }
15
16     public boolean isinitialized(){
17         return data != null;
18     }
19
20     public boolean login(int id, String password){
21         this.userId = id;
22         this.password = password;
23         return isLoggedIn();
24     }
25
26     public void logout(){
27         this.userId = 0;
28         this.password = "";
29     }
30
31     public boolean isLoggedIn(){
32         try{
33             UserDTO user = data.getUser(userId);
34             if (user.getPassword().equals(password) && user.getUserType() !=
35                 UserType.INACTIVE)
36                 return true;
```

```
36     } catch (Exception e) {}
37     return false;
38 }
39
40 public boolean isAdmin() {
41     return accessLevel() <= UserType.ADMIN.ordinal();
42 }
43
44 public boolean isPharmacist() {
45     return accessLevel() <= UserType.PHARMACIST.ordinal();
46 }
47
48 public boolean isForeman() {
49     return accessLevel() <= UserType.FOREMAN.ordinal();
50 }
51
52
53 public int accessLevel() {
54     try {
55         UserDTO user = data.getUser(userId);
56         return user.getAccessLevel();
57     } catch (Exception e) {}
58     return Integer.MAX_VALUE;
59 }
60
61 public int getId() {
62     return userId;
63 }
64 }
```



### A.2.2 admin.data

Koden i vores **admin** data pakke

#### CommodityBatchData

```
1 package admin.data;
2
3 import java.sql.ResultSet;
4 import java.sql.SQLException;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class CommodityBatchData implements ICommodityBatchDAO {
9
10     @Override
11     public synchronized CommodityBatchDTO getCommodityBatch(
12         int commoditybatch_id) throws DALEException {
13         try {
14             Connector.connect();
15         } catch (Exception e1) {
16             throw new DALEException(
17                 "Der kunne ikke oprettes forbindelse til databasen");
18         }
19         ResultSet rs = Connector.doQuery("SELECT * FROM commoditybatch WHERE
20             "
21             + "commoditybatch_id = " + commoditybatch_id + ";");
22         Connector.closeConnection();
23         try {
24             if (!rs.first()) {
25                 throw new DALEException("the commoditybatch with the id = "
26                     + commoditybatch_id + " does not exist");
27             }
28             return new CommodityBatchDTO(rs.getInt("commoditybatch_id"),
29                 rs.getInt("commodity_id"), rs.getInt("amount"));
30         } catch (SQLException e) {
31             throw new DALEException(
32                 "Der skete en fejl i CommodityBatchData i getCommodityBatch"
33                 + e.getMessage());
34         }
35
36     @Override
37     public synchronized List<CommodityBatchDTO> getComBatchList()
38         throws DALEException {
39         try {
40             Connector.connect();
41         } catch (Exception e1) {
42             throw new DALEException(
43                 "Der kunne ikke oprettes forbindelse til databasen");
44         }
45         List<CommodityBatchDTO> list = new ArrayList<CommodityBatchDTO>();
46         ResultSet rs = Connector.doQuery("SELECT * FROM commoditybatch;");
47         try {
48             while (rs.next()) {
```

```

49         list.add(new CommodityBatchDTO(rs.getInt("commoditybatch_id"),
50             rs.getInt("commodity_id"), rs.getDouble("amount")));
51     }
52 } catch (SQLException e) {
53     throw new DALException(
54         "Der skete en fejl i CommodityBatchData i getComBatchList()"
55         + e.getMessage());
56 }
57 Connector.closeConnection();
58 return list;
59 }
60
61 @Override
62 public synchronized void createCommodityBatch(CommodityBatchDTO
63     commodity)
64     throws DALException {
65     try {
66         Connector.connect();
67     } catch (Exception e1) {
68         throw new DALException(
69             "Der kunne ikke oprettes forbindelse til databasen");
70     }
71     Connector.doUpdate("INSERT INTO commoditybatch VALUES ( "
72         + commodity.getCommodityBatchId() + ", "
73         + commodity.getCommodityId() + ", " + commodity.getAmount()
74         + ");");
75     Connector.closeConnection();
76 }
77
78 @Override
79 public void updateCommodityBatch(CommodityBatchDTO commoditybatch)
80     throws DALException {
81     try {
82         Connector.connect();
83     } catch (Exception e1) {
84         throw new DALException(
85             "Der kunne ikke oprettes forbindelse til databasen");
86     }
87     Connector.doUpdate("UPDATE commoditybatch " + "SET " + " amount = "
88         + commoditybatch.getAmount() + ";");
89     Connector.closeConnection();
90 }
91
92 @Override
93 public void deleteCommodityBatch(int commoditybatch_id) throws
94     DALException {
95     try {
96         Connector.connect();
97     } catch (Exception e1) {
98         e1.printStackTrace();
99     }
100     ResultSet rs = Connector
        .doQuery("SELECT * FROM commoditybatch where commoditybatch_id in
            ( SELECT commoditybatch_id from productbatchcomponent );");
    try {

```

```
101         if (!rs.first()) {
102             Connector
103                 .doUpdate("DELETE FROM commoditybatch WHERE commoditybatch_id
104                     = "
105                     + commoditybatch_id + ";");
106             Connector.closeConnection();
107         } else {
108             throw new DAException(
109                 "You cannot delete the commodity batch, because it has a
110                 productbatchcomponent attached to it's id");
111         }
112     } catch (SQLException e) {
113         throw new DAException(e);
114     }
115 }
116
117 }
```

### CommodityBatchDTO

```
1 package admin.data;
2
3 public class CommodityBatchDTO {
4
5     private int commodityBatchId, commodityId;
6     private double amount;
7
8     /**
9      * Constructor with no parameters and no error checks
10     */
11     public CommodityBatchDTO() {
12
13     }
14
15     /**
16      * Constructor with parameters from set methods and thus error checks
17      *
18      * @param commodityBatchId
19      * @param commodityId
20      * @param amount
21      * @throws DAException
22      */
23     public CommodityBatchDTO(int commodityBatchId, int commodityId,
24         double amount) throws DAException {
25         setAmount(amount);
26         setCommodityBatchId(commodityBatchId);
27         setCommodityId(commodityId);
28     }
29
30     public int getCommodityBatchId() {
31         return commodityBatchId;
32     }
33 }
```

```

34  public void setCommodityBatchId(int commodityBatchId) throws
      DALException {
35      if (commodityBatchId <= 0) {
36          throw new DALException("Råvare batch id'et skal være større end 0")
              ;
37      }
38      this.commodityBatchId = commodityBatchId;
39  }
40
41  public int getCommodityId() {
42      return commodityId;
43  }
44
45  public void setCommodityId(int commodityId) throws DALException {
46      if (commodityId <= 0) {
47          throw new DALException("Råvare id'et skal være større end 0");
48      }
49      this.commodityId = commodityId;
50  }
51
52  public double getAmount() {
53      return amount;
54  }
55
56  public void setAmount(double amount) throws DALException {
57      if (amount < 0) {
58          throw new DALException("Man kan ikke trække en negativ mængde fra i
              lageret");
59      }
60      this.amount = amount;
61  }
62  }

```

### CommodityData

```

1  package admin.data;
2
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.util.ArrayList;
6  import java.util.List;
7
8  public class CommodityData implements ICommodityDAO {
9
10     @Override
11     public synchronized CommodityDTO getCommodity(int commodity_id)
12         throws DALException {
13         try {
14             Connector.connect();
15         } catch (Exception e1) {
16             throw new DALException(
17                 "Der kunne ikke oprettes forbindelse til databasen");
18         }
19         ResultSet rs = Connector
20             .doQuery("SELECT * FROM commodity WHERE commodity_id = "
21                 + commodity_id + ";");

```

```
22     Connector.closeConnection();
23     try {
24         if (!rs.first()) {
25             throw new DALEException("the commodity with the id = "
26                 + commodity_id + " does not exist");
27         }
28         return new CommodityDTO(rs.getInt("commodity_id"),
29             rs.getString("commodity_name"), rs.getString("supplier"));
30     } catch (SQLException e) {
31         throw new DALEException(
32             "Der skete en fejl i Commodity i metoden getCommodity()"
33             + e.getMessage());
34     }
35 }
36
37 @Override
38 public synchronized List<CommodityDTO> getComList() throws DALEException
39 {
40     try {
41         Connector.connect();
42     } catch (Exception e1) {
43         throw new DALEException(
44             "Der kunne ikke oprettes forbindelse til databasen");
45     }
46     List<CommodityDTO> list = new ArrayList<CommodityDTO>();
47     ResultSet rs = Connector.doQuery("SELECT * FROM commodity");
48     Connector.closeConnection();
49     try {
50         while (rs.next()) {
51             list.add(new CommodityDTO(rs.getInt("commodity_id"), rs
52                 .getString("commodity_name"), rs.getString("supplier")));
53         }
54     } catch (SQLException e) {
55         throw new DALEException(
56             "Der skete en fejl i Commodity i metoden getComList()"
57             + e.getMessage());
58     }
59     return list;
60 }
61
62 @Override
63 public synchronized void createCommodity(CommodityDTO commodity)
64     throws DALEException {
65     try {
66         Connector.connect();
67     } catch (Exception e1) {
68         throw new DALEException(
69             "Der kunne ikke oprettes forbindelse til databasen");
70     }
71     Connector.doUpdate("INSERT INTO commodity VALUES ( "
72         + commodity.getComId() + ", ' " + commodity.getComName() + " ', ' "
73         + commodity.getSupplier() + " ');");
74     Connector.closeConnection();
75 }
```

```

76  @Override
77  public void updateCommodity(CommodityDTO commodity) throws DALException
    {
78      try {
79          Connector.connect();
80      } catch (Exception e1) {
81          throw new DALException(
82              "Der kunne ikke oprettes forbindelse til databasen");
83      }
84      Connector.doUpdate("UPDATE commodity " + "SET"
85          + " commodity_name = '" + commodity.getComName()
86          + "', supplier = '" + commodity.getSupplier()
87          + "' WHERE commodity_id = " + commodity.getComId()
88          + ";");
89      Connector.closeConnection();
90  }
91
92  @Override
93  public void deleteCommodity(int commodity_id) throws DALException {
94      try {
95          Connector.connect();
96      } catch (Exception e1) {
97          throw new DALException(
98              "Der kunne ikke oprettes forbindelse til databasen");
99      }
100     ResultSet rs = Connector
101         .doQuery("SELECT * FROM commodity WHERE commodity_id IN "
102             + "(SELECT commodity_id from prescriptioncomponent);");
103     try {
104         if (!rs.next()) {
105             Connector
106                 .doUpdate("DELETE FROM commodity WHERE commodity_id = "
107                     + commodity_id + ";");
108             Connector.closeConnection();
109         }
110         else {
111             throw new DALException("id'et er allerede blevet brugt i nogle
112                 afvejninger");
113         }
114     } catch (SQLException e) {
115         throw new DALException(
116             "Noget gik galt i forbindelse med databasen.");
117     }
118 }
119 }

```

### CommodityDTO

```

1  package admin.data;
2
3  public class CommodityDTO {
4      private int commodity_id;
5      private String commodity_name, supplier;
6
7      /**

```

```
8      * Constructor with no parameters and no error checks
9      */
10     public CommodityDTO() {
11         commodity_id = 0;
12         commodity_name = "";
13         supplier = "";
14     }
15
16     /**
17      * Constructor with parameters from set methods and thus error checks
18      *
19      * @param commodity_id
20      * @param commodity_name
21      * @param supplier
22      * @throws DALEException
23      */
24     public CommodityDTO(int commodity_id, String commodity_name, String
        supplier)
        throws DALEException {
25         setCommodity_id(commodity_id);
26         setCommodity_name(commodity_name);
27         setSupplier(supplier);
28     }
29
30
31     public void setCommodity_id(int commodity_id) throws DALEException {
32         if (commodity_id <= 0) {
33             throw new DALEException("Råvare id'et skal være større end 0");
34         }
35         this.commodity_id = commodity_id;
36     }
37
38     public void setCommodity_name(String commodity_name) throws
        DALEException {
39         this.commodity_name = commodity_name;
40     }
41
42     public void setSupplier(String supplier) throws DALEException {
43         this.supplier = supplier;
44     }
45
46     public int getComId() {
47         return commodity_id;
48     }
49
50     public String getComName() {
51         return commodity_name;
52     }
53
54     public String getSupplier() {
55         return supplier;
56     }
57 }
```

### Connector

```
1 package admin.data;
```

```
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.sql.Statement;
8
9 public class Connector {
10
11     /**
12      * To connect to a MySQL-server
13      *
14      * @param url
15      *      must have the form "jdbc:mysql://<server>/<database>" for
16      *      default port (3306) OR
17      *      "jdbc:mysql://<server>:<port>/<database>" for specific
18      *      port
19      *      more formally "jdbc:subprotocol:subname"
20      * @throws ClassNotFoundException
21      * @throws IllegalAccessException
22      * @throws InstantiationException
23      * @throws SQLException
24      */
25     public static Connection connectToDatabase(String url, String username,
26         String password) throws InstantiationException,
27         IllegalAccessException, ClassNotFoundException, SQLException {
28         // call the driver class' no argument constructor
29         Class.forName("com.mysql.jdbc.Driver").newInstance();
30         // get Connection-object via DriverManager
31         return (Connection) DriverManager
32             .getConnection(url, username, password);
33     }
34
35     private static Connection conn;
36     private static Statement stm;
37
38     /**
39      * Connects with info on server, database, username and password
40      *
41      * @throws InstantiationException
42      * @throws IllegalAccessException
43      * @throws ClassNotFoundException
44      * @throws SQLException
45      */
46     public static void connect() throws InstantiationException,
47         IllegalAccessException, ClassNotFoundException, SQLException {
48         conn = connectToDatabase("jdbc:mysql://" + Constant.server + ":"
49             + Constant.port + "/" + Constant.database, Constant.username,
50             Constant.password);
51         stm = conn.createStatement();
52     }
53
54     /**
55      * Gets a resultset from when you do a query
```



```

56  *
57  * @param cmd
58  *         string with the query
59  * @return
60  * @throws SQLException
61  */
62  public static ResultSet doQuery(String cmd) throws SQLException {
63      try {
64          return stm.executeQuery(cmd);
65      } catch (SQLException e) {
66          throw new SQLException(e);
67      }
68  }
69
70  /**
71   * Used for updating tuples in data base
72   *
73   * @param cmd
74   *         string with the query
75   * @return
76   * @throws SQLException
77   */
78  public static int doUpdate(String cmd) throws SQLException {
79      try {
80          return stm.executeUpdate(cmd);
81      } catch (SQLException e) {
82          throw new SQLException(e);
83      }
84  }
85
86  /**
87   * Close connection to data base
88   */
89  public static void closeConnection() {
90      try {
91          conn.close();
92      } catch (SQLException e) {
93      }
94  }
95  }
96  }

```

#### Constant

```

1  package admin.data;
2
3  /**
4   * Class that stores connection values to the data base
5   *
6   * @author Group 53
7   *
8   */
9  public abstract class Constant {
10     public static final
11     String server = "localhost", database = "weight", username = "root",
        password = "1234";

```

```
12 // public static final String server = "localhost", database = "
    testweight",
13 // username = "root", password = "1234";
14
15 // port number to establish connection
16 public static final int port = 3306;
17 }
```

### DALEXception

```
1 package admin.data;
2
3 import java.sql.SQLException;
4
5 /**
6  * Data Access Layer Exception is thrown when errors occur in the data
7  * layer
8  */
9 public class DALEXception extends Exception {
10     private static final long serialVersionUID = 1L;
11
12     public DALEXception(String s) {
13         super(s);
14     }
15
16     public DALEXception(SQLException s) {
17         super(s);
18     }
19 }
```

### ICommodityBatchDAO

```
1 package admin.data;
2
3 import java.util.List;
4
5 /**
6  * Interface containing the methods used for a commodity batch
7  *
8  * @author Group 53
9  */
10
11 public interface ICommodityBatchDAO {
12
13     /**
14      * Method to get the commodity batch with a specific ID
15      *
16      * @param commoditybatch_id
17      *      ID for the wanted commodity batch
18      * @return commodity batches with the wanted ID
19      * @throws DALEXception
20      */
21     public CommodityBatchDTO getCommodityBatch(int commoditybatch_id)
22         throws DALEXception;
23
24     /**
```

```
25     * Method to get a list of all commodity batches
26     *
27     * @return a list of all commodity batches present in the database
28     * @throws DALException
29     */
30     public List<CommodityBatchDTO> getComBatchList() throws DALException;
31
32     public void createCommodityBatch(CommodityBatchDTO commoditybatch)
33         throws DALException;
34
35     public void updateCommodityBatch(CommodityBatchDTO commoditybatch)
36         throws DALException;
37
38     public void deleteCommodityBatch(int commoditybatch_id) throws
39         DALException;
40 }
41
42 ICommodityDAO
43
44 1 package admin.data;
45 2
46 3 import java.util.List;
47 4
48 5 public interface ICommodityDAO {
49 6
50 7     public CommodityDTO getCommodity(int commodity_id) throws DALException;
51 8
52 9     public List<CommodityDTO> getComList() throws DALException;
53 10
54 11     public void createCommodity(CommodityDTO commodity) throws DALException
55 12         ;
56 13     public void updateCommodity(CommodityDTO commodity) throws DALException
57 14         ;
58 15     public void deleteCommodity(int commodity_id) throws DALException;
59 16
60 17 }
61
62 IPrescriptionCompDAO
63
64 1 package admin.data;
65 2
66 3 import java.util.List;
67 4
68 5 public interface IPrescriptionCompDAO {
69 6
70 7     /**
71 8     * Get the prescription component containing the specified commodity
72 9     * assigned to the specified prescription
73 10     *
74 11     * @param prescriptionId
75 12     *         the id of the prescription containing the component
76 13     * @param commodityId
77 14     *         the id of the commodity in the component
78 15     * @return the one component with the given attributes
```

```
16     * @throws DALException
17     *         if no component with the given specifications exist
18     */
19     public PrescriptionCompDTO getPrescriptionComp(int prescriptionId ,
20         int commodityId) throws DALException;
21
22     /**
23     * Get the list of components assigned to a specific prescription
24     *
25     * @param prescriptionId
26     *         the id of the prescription having the components
27     * @return a list of components assigned to the specified prescription
28     * @throws DALException
29     *         if no prescription with the given id exist.
30     */
31     public List<PrescriptionCompDTO> getComponentList(int prescriptionId)
32         throws DALException;
33
34     /**
35     * Get the list of all prescription components in the database
36     *
37     * @return the complete list of prescription components
38     * @throws DALException
39     *         maybe ?
40     */
41     public List<PrescriptionCompDTO> getComponentList() throws DALException
42         ;
43
44     /**
45     * Creates a new entry in the database with the information given in
46     * the
47     * provided <code>PrescriptionCompDTO</code>
48     *
49     * @param prescription
50     *         the <code>PrescriptionCompDTO</code> containing the
51     *         information to be added to the database
52     * @throws DALException
53     *         if a component with the same prescription id and
54     *         commodity id
55     *         already exists
56     */
57     public void createPrescriptionComp(PrescriptionCompDTO component)
58         throws DALException;
59
60     /**
61     * Removes the prescription component with the given prescription id
62     * and
63     * commodity id from the database
64     *
65     * @param prescriptionId
66     *         the id of the prescription having the component
67     * @param commodityId
68     *         the id of the commodity contained in the component
69     * @throws DALException
```

```
67 public void deletePrescriptionComp(int prescriptionId, int commodityId)
68     throws DALException;
69
70 /**
71  * Removes all prescription components tied to a certain prescription
72  *
73  * @param prescriptionId
74  *     the id of the prescription that is to have all its
75  *     components
76  *     removed
77  * @throws DALException
78  */
79 public void deletePrescriptionComp(int prescriptionId) throws
    DALException;
```

#### IPrescritpionDAO

```
1 package admin.data;
2
3 import java.util.List;
4
5 public interface IPrescritpionDAO {
6
7     public PrescriptionDTO getPrescription(int prescriptionId)
8         throws DALException;
9
10    public List<PrescriptionDTO> getPrescriptionList()
11        throws DALException;
12
13    public void createPrescription(PrescriptionDTO prescription)
14        throws DALException;
15
16    public void updatePrescription(PrescriptionDTO prescription)
17        throws DALException;
18
19    public void deletePrescription(int id)
20        throws DALException;
21
22
23 }
```

#### IProductBatchCompDAO

```
1 package admin.data;
2
3 import java.util.List;
4
5 public interface IProductBatchCompDAO {
6
7     public ProductBatchCompDTO getProductBatchComp(int pb_id, int
8         commoditybatch_id) throws DALException;
9
10    public ProductBatchCompDTO getCompByComId(int pb_id, int commodity_id)
11        throws DALException;
```

```
11  public List<ProductBatchCompDTO> getCertainProductBatchComps(int pb_id)
    throws DALException;
12
13  public List<ProductBatchCompDTO> getAllProductBatchComps() throws
    DALException;
14
15  public void createProductBatchComp(ProductBatchCompDTO productBatchComp
    ) throws DALException;
16
17  public void updateProductBatchComp(ProductBatchCompDTO productBatchComp
    ) throws DALException;
18
19  public void deleteByBatchID(int productBatchId) throws DALException;
20
21  public void deleteProductBatchComp(int pb_id, int commoditybatch_id)
    throws DALException;
22
23  public String getName(int pb_id) throws DALException;
24
25  public List<PrescriptionCompDTO> getUnfulfilledComps(int pb_id) throws
    DALException;
26
27 }
```

#### IUserDAO

```
1  package admin.data;
2
3  /**
4   *
5   * The read and write access interface to the database.
6   * <p>
7   * The interface extends <code>IDataReadOnly</code> with methods needed
    to edit
8   * users in the database
9   *
10  */
11  public interface IUserDAO extends IUsersReadOnly {
12      /**
13       * Inserts a new user entry into the database
14       *
15       * @param user
16       *         the entry to be added to the database
17       * @throws DALException
18       *         if an user with the same id already exist
19       */
20      void createUser(UserDTO user) throws DALException;
21
22      /**
23       * Updates an existing user already in the database
24       *
25       * @param user
26       *         the new info on the user
27       * @throws DALException
28       *         if no user with the same id exist
29       */

```

```

30     void updateUser(UserDTO user) throws DALException;
31
32     /**
33      * Deletes the user with the given ID.
34      * @param id the ID of the user to be deleted
35      * @throws DALException if no user with the given ID exist.
36      */
37     void deleteUser(int id) throws DALException;
38 }

```

**IUsersReadOnly**

```

1  package admin.data;
2
3  import java.util.List;
4
5  /**
6   * The read-only interface to the data layer
7   */
8  public interface IUsersReadOnly {
9      /**
10       *
11       * @param oprId
12       *         the id of the user to get info on
13       * @return the in info on the user with the given id
14       * @throws DALException
15       *         if the id is out of bounds or user id doesn't exist in
16       *         the database
17       */
18     UserDTO getUser(int userId) throws DALException;
19
20     /**
21      *
22      * @return a list of all users present in the database
23      * @throws DALException
24      */
25     List<UserDTO> getUserList() throws DALException;
26 }

```

**MenuOption**

```

1  package admin.data;
2
3  public class MenuOption {
4      public String name;
5      public String url;
6
7      public MenuOption(String name, String url){
8          this.name = name;
9          this.url = url;
10     }
11 }

```

**PrescriptionCompData**

```

1  package admin.data;
2

```

```
3 import java.sql.ResultSet;
4 import java.sql.SQLException;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class PrescriptionCompData implements IPrescriptionCompDAO {
9
10     @Override
11     public synchronized void createPrescriptionComp(PrescriptionCompDTO
12         prescription)
13         throws DALException {
14         try {
15             Connector.connect();
16         } catch (Exception e1) {
17             throw new DALException("Der kunne ikke oprettes forbindelse til
18                 databasen");
19         }
20         Connector.doUpdate("INSERT INTO prescriptioncomponent VALUES ( "
21             + prescription.getPrescriptionId()
22             + ", " + prescription.getCommodityId()
23             + ", " + prescription.getNomNetto()
24             + ", " + prescription.getTolerance() + ");");
25         Connector.closeConnection();
26     }
27
28     @Override
29     public synchronized PrescriptionCompDTO getPrescriptionComp(int
30         prescription_id, int commodity_id)
31         throws DALException {
32         try {
33             Connector.connect();
34         } catch (Exception e1) {
35             throw new DALException("Der kunne ikke oprettes forbindelse til
36                 databasen");
37         }
38         ResultSet rs = Connector
39             .doQuery("SELECT * FROM prescriptioncomponent WHERE
40                 prescription_id = "
41                 + prescription_id + " AND commodity_id = " + commodity_id + ";
42                 ");
43         Connector.closeConnection();
44         try {
45             if (!rs.first()) {
46                 throw new DALException("the commodity with the id = "
47                     + prescription_id + " does not exist");
48             }
49             return new PrescriptionCompDTO(rs.getInt("prescription_id"),
50                 rs.getInt("commodity_id"), rs.getDouble("nom_netto"),
51                 rs.getDouble("tolerance"));
52         } catch (SQLException e) {
53             throw new DALException("Der skete en forbindelse med databasen");
54         }
55     }
56
57     @Override
```



```
52 public List<PrescriptionCompDTO> getComponentList(int prescriptionId)
53     throws DALException {
54     try {
55         Connector.connect();
56     } catch (Exception e1) {
57         throw new DALException("Der kunne ikke oprettes forbindelse til
58             databasen");
59     }
60     List<PrescriptionCompDTO> list = new ArrayList<PrescriptionCompDTO>()
61     ;
62     ResultSet rs = Connector
63         .doQuery("SELECT * FROM weight.prescriptioncomponent "
64             + "WHERE prescription_id=" + prescriptionId + " ");
65     Connector.closeConnection();
66     try {
67         while (rs.next()) {
68             list.add(new PrescriptionCompDTO(rs.getInt("prescription_id"),
69                 rs.getInt("commodity_id"), rs.getDouble("nom_netto"),
70                 rs.getDouble("tolerance")));
71         }
72     } catch (SQLException e) {
73         throw new DALException("Der skete en fejl i forbindelse med
74             databasen");
75     }
76     return list;
77 }
78 @Override
79 public List<PrescriptionCompDTO> getComponentList() throws DALException
80 {
81     try {
82         Connector.connect();
83     } catch (Exception e1) {
84         throw new DALException("Der kunne ikke oprettes forbindelse til
85             databasen");
86     }
87     List<PrescriptionCompDTO> list = new ArrayList<PrescriptionCompDTO>()
88     ;
89     ResultSet rs = Connector
90         .doQuery("SELECT * FROM prescriptioncomponent;");
91     Connector.closeConnection();
92     try {
93         while (rs.next()) {
94             list.add(new PrescriptionCompDTO(rs.getInt("prescription_id"),
95                 rs.getInt("commodity_id"), rs.getDouble("nom_netto"),
96                 rs.getDouble("tolerance")));
97         }
98     } catch (SQLException e) {
99         throw new DALException("Der skete en fejl i forbindelse med
100             databasen");
101     }
102     return list;
103 }
```

```

100  @Override
101  public void deletePrescriptionComp(int prescriptionId , int commodityId)
102      throws DALException {
103      try {
104          Connector.connect();
105      } catch (Exception e1) {
106          throw new DALException("Der kunne ikke oprettes forbindelse til
107              databasen");
108      }
109      try {
110          Connector.doUpdate("DELETE FROM prescriptioncomponent WHERE
111              prescription_id = " + prescriptionId
112              + " AND commodity_id = " + commodityId + ";" );
113          Connector.closeConnection();
114      } catch (DALException e) {
115          throw new DALException("recept komponenten kunne ikke slettes ");
116      }
117  }
118  @Override
119  public void deletePrescriptionComp(int prescriptionId) throws
120      DALException {
121      try {
122          Connector.connect();
123      } catch (Exception e1) {
124          throw new DALException("Der kunne ikke oprettes forbindelse til
125              databasen");
126      }
127      Connector.doUpdate("DELETE FROM prescriptioncomponent WHERE
128          prescription_id = " + prescriptionId + ";");
129  }

```

### PrescriptionCompDTO

```

1  package admin.data;
2
3  public class PrescriptionCompDTO {
4
5      private int prescriptionId , commodityId;
6      private double nomNetto, tolerance;
7
8      /**
9       * Constructor with no parameters and no error checks
10      */
11      public PrescriptionCompDTO() {
12          prescriptionId = 0;
13          commodityId = 0;
14          nomNetto = 0;
15          tolerance = 0;
16      }
17
18      /**
19       * Constructor with parameters from set methods and thus error checks

```

```
20  *
21  * @param prescriptionId
22  * @param commodityId
23  * @param nomNetto
24  * @param tolerance
25  * @throws DALException
26  */
27  public PrescriptionCompDTO(int prescriptionId, int commodityId,
28      double nomNetto, double tolerance) throws DALException {
29      setPrescriptionId(prescriptionId);
30      setCommodityId(commodityId);
31      setNomNetto(nomNetto);
32      setTolerance(tolerance);
33  }
34
35  public void setPrescriptionId(int prescriptionId) throws DALException {
36      if(prescriptionId <=0){
37          throw new DALException("Recept id'et skal være større end 0");
38      }
39      this.prescriptionId = prescriptionId;
40  }
41
42  public void setCommodityId(int commodityId) throws DALException {
43      if(commodityId <=0){
44          throw new DALException("Råvare id'et skal være større end 0");
45      }
46      this.commodityId = commodityId;
47  }
48
49  public void setNomNetto(double nomNetto) throws DALException {
50      if(nomNetto < 0.0){
51          throw new DALException("nomNetto skal være en positiv værdi");
52      }
53      this.nomNetto = nomNetto;
54  }
55
56  public void setTolerance(double tolerance) throws DALException {
57      if(tolerance < 0.0){
58          throw new DALException("Man kan ikke have en negativ tolerance");
59      }
60      this.tolerance = tolerance;
61  }
62
63  public int getPrescriptionId() {
64      return prescriptionId;
65  }
66
67  public int getCommodityId() {
68      return commodityId;
69  }
70
71  public double getNomNetto() {
72      return nomNetto;
73  }
74
```

```
75     public double getTolerance() {
76         return tolerance;
77     }
78 }
```

### PrescriptionData

```
1  package admin.data;
2
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.util.ArrayList;
6  import java.util.List;
7
8  public class PrescriptionData implements IPrescriptionDAO {
9
10     @Override
11     public synchronized void createPrescription(PrescriptionDTO
        prescription)
        throws DALException {
12         try {
13             Connector.connect();
14         } catch (Exception e1) {
15             throw new DALException(
16                 "Der kunne ikke oprettes forbindelse til databasen");
17         }
18         Connector.executeUpdate("INSERT INTO prescription VALUES ( "
19             + prescription.getId() + ",'" + prescription.getName() + " ');");
20         Connector.closeConnection();
21     }
22
23
24     @Override
25     public synchronized PrescriptionDTO getPrescription(int prescriptionId)
        throws DALException {
26         try {
27             Connector.connect();
28         } catch (Exception e1) {
29             throw new DALException(
30                 "Der kunne ikke oprettes forbindelse til databasen");
31         }
32         ResultSet rs = Connector
33             .doQuery("SELECT * FROM prescription WHERE prescription_id = "
34                 + prescriptionId + ";");
35         Connector.closeConnection();
36         try {
37             if (!rs.first()) {
38                 throw new DALException("råvaren med råvare id = "
39                     + prescriptionId + " kunne ikke findes i databasen");
40             }
41             return new PrescriptionDTO(rs.getInt("prescription_id"),
42                 rs.getString("prescription_name"));
43         } catch (SQLException e) {
44             throw new DALException(
45                 "Der skete en fejl i getPrescription(int prescriptionId)"
46                 + e.getMessage());
47         }
48     }
49 }
```

```
49     }
50
51     @Override
52     public synchronized List<PrescriptionDTO> getPrescriptionList()
53         throws DALEException {
54         try {
55             Connector.connect();
56         } catch (Exception e1) {
57             throw new DALEException(
58                 "Der kunne ikke oprettes forbindelse til databasen");
59         }
60         List<PrescriptionDTO> list = new ArrayList<PrescriptionDTO>();
61         ResultSet rs = Connector.doQuery("SELECT * FROM prescription;");
62         Connector.closeConnection();
63         try {
64             while (rs.next()) {
65                 list.add(new PrescriptionDTO(rs.getInt("prescription_id"), rs
66                     .getString("prescription_name")));
67             }
68         } catch (SQLException e) {
69             throw new DALEException("Der skete en fejl i getPrescriptionList()"
70                 + e.getMessage());
71         }
72         return list;
73     }
74
75     @Override
76     public synchronized void updatePrescription(PrescriptionDTO
77         prescription)
78         throws DALEException {
79         try {
80             Connector.connect();
81         } catch (Exception e1) {
82             throw new DALEException(
83                 "Der kunne ikke oprettes forbindelse til databasen");
84         }
85         Connector.doUpdate("UPDATE prescription SET "
86             + " prescription_name = '" + prescription.getName() + "'"
87             + " WHERE prescription_id = " + prescription.getId() + ";");
88         Connector.closeConnection();
89     }
90
91     @Override
92     public synchronized void deletePrescription(int id) throws DALEException
93     {
94         try {
95             Connector.connect();
96         } catch (Exception e1) {
97             throw new DALEException(
98                 "Der kunne ikke oprettes forbindelse til databasen");
99         }
100         try {
101             Connector
102                 .doQuery("SELECT * FROM prescription WHERE prescription_id IN "
103                     + "(SELECT prescription_id from prescriptioncomponent);");
```

```
102     } catch (DALException e) {
103         Connector
104             .doUpdate("DELETE FROM prescription WHERE prescription_id = "
105                 + id + ";");
106         Connector.closeConnection();
107     }
108     throw new DALException("du kan ikke slette den ønskede recept");
109 }
110 }
111 }
```

### PrescriptionDTO

```
1 package admin.data;
2
3 public class PrescriptionDTO {
4
5     private int id;
6     private String name;
7
8     /**
9      * Constructor with no parameters and no error checks
10     */
11     public PrescriptionDTO() {
12         id = 0;
13         name = "";
14     }
15
16     /**
17      * Constructor with parameters from set methods and thus error checks
18      *
19      * @param id
20      * @param name
21      * @throws DALException
22      */
23     public PrescriptionDTO(int id, String name) throws DALException {
24         setId(id);
25         setName(name);
26     }
27
28     public void setId(int prescriptionId) throws DALException {
29         if(prescriptionId <= 0){
30             throw new DALException("Recept id'et skal være større end 0");
31         }
32         this.id = prescriptionId;
33     }
34
35     public void setName(String prescriptionName) throws DALException {
36         this.name = prescriptionName;
37     }
38
39     public int getId() {
40         return id;
41     }
42
43     public String getName() {
```

```
44     return name;
45 }
46
47 }

ProductBatchCompData

1 package admin.data;
2
3 import java.sql.ResultSet;
4 import java.sql.SQLException;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class ProductBatchCompData implements IProductBatchCompDAO {
9
10     @Override
11     public synchronized ProductBatchCompDTO getProductBatchComp(int pb_id,
12         int commoditybatch_id) throws DALException {
13         try {
14             Connector.connect();
15         } catch (Exception e1) {
16             throw new DALException(
17                 "Der kunne ikke oprettes forbindelse til databasen");
18         }
19         ResultSet rs = Connector
20             .doQuery("SELECT * FROM productbatchcomponent WHERE pb_id = "
21                 + pb_id + " AND commoditybatch_id = "
22                 + commoditybatch_id + ";");
23         Connector.closeConnection();
24         try {
25             if (!rs.first()) {
26                 throw new DALException(
27                     "Produkt batchen med det givne produkt batch id = "
28                     + pb_id
29                     + " og det tilsvarende r vare batch id = "
30                     + commoditybatch_id
31                     + " eksisterer ikke i databasen");
32             }
33             return new ProductBatchCompDTO(rs.getInt("pb_id"),
34                 rs.getInt("commoditybatch_id"), rs.getInt("user_id"),
35                 rs.getDouble("tara"), rs.getDouble("netto"));
36         } catch (SQLException e) {
37             throw new DALException(
38                 "Der skete en fejl i ProductBatchCompData i metoden
39                 getProductBatchComp(int pb_id, int commoditybatch_id)"
40                 + e.getMessage());
41         }
42
43     @Override
44     public synchronized ProductBatchCompDTO getCompByComId(int pb_id,
45         int commodity_id) throws DALException {
46         try {
47             Connector.connect();
48         } catch (Exception e1) {
```

```
49     throw new DALException(  
50         "Der kunne ikke oprettes forbindelse til databasen");  
51     }  
52     ResultSet rs = Connector  
53         .doQuery("SELECT * FROM productbatchcomponent NATURAL JOIN  
54             commoditybatch WHERE pb_id = "  
55             + pb_id + " AND commodity_id = "  
56             + commodity_id + ";");  
57     Connector.closeConnection();  
58     try {  
59         if (!rs.first()) {  
60             throw new DALException(  
61                 "Produkt batchen med det givne produkt batch id = "  
62                 + pb_id  
63                 + " og det tilsvarende r vare batch id = "  
64                 + commodity_id  
65                 + " eksisterer ikke i databasen");  
66         }  
67         return new ProductBatchCompDTO(rs.getInt("pb_id"),  
68             rs.getInt("commoditybatch_id"), rs.getInt("user_id"),  
69             rs.getDouble("tara"), rs.getDouble("netto"));  
70     } catch (SQLException e) {  
71         throw new DALException(  
72             "Der skete en fejl i ProductBatchCompData i metoden  
73             getCompByComId()" + e.getMessage());  
74     }  
75  
76     @Override  
77     public synchronized List<ProductBatchCompDTO>  
78         getCertainProductBatchComps(  
79             int pb_id) throws DALException {  
80         List<ProductBatchCompDTO> list = new ArrayList<ProductBatchCompDTO>();  
81         try {  
82             Connector.connect();  
83         } catch (Exception e1) {  
84             throw new DALException(  
85                 "Der kunne ikke oprettes forbindelse til databasen");  
86         }  
87         ResultSet rs = Connector  
88             .doQuery("SELECT * FROM productbatchcomponent WHERE pb_id = "  
89             + pb_id + ";");  
90         Connector.closeConnection();  
91         try {  
92             while (rs.next()) {  
93                 list.add(new ProductBatchCompDTO(rs.getInt("pb_id"), rs  
94                     .getInt("commoditybatch_id"), rs.getInt("user_id"), rs  
95                     .getDouble("tara"), rs.getDouble("netto")));  
96             }  
97         } catch (SQLException e) {  
98             throw new DALException(  
99                 "Der skete en fejl i ProductBatchCompData i metoden  
100                 getCertainProductBatchComps(int pb_id)"
```



```
99         + e.getMessage());
100     }
101     return list;
102 }
103
104 @Override
105 public synchronized List<ProductBatchCompDTO> getAllProductBatchComps()
106     throws DALEException {
107     List<ProductBatchCompDTO> list = new ArrayList<ProductBatchCompDTO>();
108     try {
109         Connector.connect();
110     } catch (Exception e1) {
111         throw new DALEException(
112             "Der kunne ikke oprettes forbindelse til databasen");
113     }
114     ResultSet rs = Connector
115         .doQuery("SELECT * FROM productbatchcomponent;");
116     Connector.closeConnection();
117     try {
118         while (rs.next()) {
119             list.add(new ProductBatchCompDTO(rs.getInt("pb_id"), rs
120                 .getInt("commoditybatch_id"), rs.getInt("user_id"), rs
121                 .getDouble("tara"), rs.getDouble("netto")));
122         }
123     } catch (SQLException e) {
124         throw new DALEException(
125             "Der skete en fejl i ProductBatchCompData i metoden
126             getAllProductBatchComps()
127             + e.getMessage());
128     }
129     return list;
130 }
131
132 @Override
133 public synchronized void createProductBatchComp(
134     ProductBatchCompDTO productBatchComp) throws DALEException {
135     try {
136         Connector.connect();
137     } catch (Exception e1) {
138         throw new DALEException(
139             "Der kunne ikke oprettes forbindelse til databasen");
140     }
141     Connector.doUpdate("INSERT INTO productbatchcomponent VALUES ( "
142         + productBatchComp.getPb_id() + ", "
143         + productBatchComp.getCommoditybatch_id() + ", "
144         + productBatchComp.getUser_id() + ", "
145         + productBatchComp.getTara() + ", "
146         + productBatchComp.getNetto() + ");");
147     Connector.closeConnection();
148 }
149
150 @Override
151 public synchronized void updateProductBatchComp(
```

```
152     ProductBatchCompDTO productBatchComp) throws DALException {
153     try {
154         Connector.connect();
155     } catch (Exception e1) {
156         throw new DALException(
157             "Der kunne ikke oprettes forbindelse til databasen");
158     }
159     Connector.doUpdate("UPDATE productbatchcomponent " + "set user_id = "
160         + productBatchComp.getUser_id() + ", tara = "
161         + productBatchComp.getTara() + ", netto = "
162         + productBatchComp.getNetto() + ";");
163     Connector.closeConnection();
164
165 }
166
167 @Override
168 public synchronized void deleteProductBatchComp(int pb_id,
169     int commoditybatch_id) throws DALException {
170     try {
171         Connector.connect();
172     } catch (Exception e1) {
173         throw new DALException(
174             "Der kunne ikke oprettes forbindelse til databasen");
175     }
176     Connector
177         .doUpdate("DELETE FROM productbatchcomponent WHERE pb_id = "
178             + pb_id + " AND commoditybatch_id = "
179             + commoditybatch_id + ";");
180     Connector.closeConnection();
181 }
182
183 @Override
184 public synchronized String getName(int pb_id) throws DALException {
185     try {
186         Connector.connect();
187     } catch (Exception e1) {
188         throw new DALException(
189             "Der kunne ikke oprettes forbindelse til databasen");
190     }
191     try {
192         ResultSet rs = Connector
193             .doQuery("SELECT prescription_name FROM prescription WHERE
194                 prescription_id IN "
195                 + "(SELECT prescription_id FROM productbatch WHERE pb_id ="
196                 + pb_id + ");");
197         return rs.getString("prescription_name");
198     } catch (SQLException e) {
199         throw new DALException(
200             "Der skete en fejl i forbindele med databasen"
201             + e.getMessage());
202     }
203 }
204
205 @Override
```

```

206 public void deleteByBatchID(int productBatchId) throws DAException {
207     try {
208         Connector.connect();
209     } catch (Exception e1) {
210         throw new DAException(
211             "Der kunne ikke oprettes forbindelse til databasen");
212     }
213     Connector.doUpdate("DELETE FROM productbatchcomponent WHERE pb_id = "
214         + productBatchId + ";");
215     Connector.closeConnection();
216 }
217
218 @Override
219 public synchronized List<PrescriptionCompDTO> getUnfulfilledComps(int
    pb_id)
220     throws DAException {
221     List<PrescriptionCompDTO> list = new ArrayList<PrescriptionCompDTO>();
222     try {
223         Connector.connect();
224     } catch (Exception e1) {
225         throw new DAException(
226             "Der kunne ikke oprettes forbindelse til databasen");
227     }
228     ResultSet rs = Connector
229         .doQuery("SELECT * FROM prescriptioncomponent WHERE
    prescription_id IN "
230             + "(SELECT prescription_id from productbatch WHERE pb_id = "
231                 + pb_id + ") "
232             + " AND commodity_id NOT IN (SELECT commodity_id FROM "
233                 + "productbatchcomponent NATURAL JOIN commoditybatch WHERE
    pb_id = "
234                 + pb_id + ");");
234     Connector.closeConnection();
235     try {
236         while (rs.next()) {
237             list.add(new PrescriptionCompDTO(rs.getInt("prescription_id"),
238                 rs.getInt("commodity_id"), rs.getDouble("nom_netto"),
239                 rs.getDouble("tolerance")));
240         }
241     } catch (SQLException e) {
242         throw new DAException("Der skete en fejl i unFulfilledComps "
243             + e.getMessage());
244     }
245     return list;
246 }
247
248 }

```

### ProductBatchCompDTO

```

1 package admin.data;
2
3 public class ProductBatchCompDTO {
4     private int pb_id, commoditybatch_id, user_id;
5     private double tara, netto;

```

```
6
7  /**
8   * Constructor with no parameters and no error checks
9   */
10 public ProductBatchCompDTO() {
11 }
12
13
14 /**
15 * Constructor with parameters from set methods and thus error checks
16 *
17 * @param pb_id
18 * @param commoditybatch_id
19 * @param user_id
20 * @param tara
21 * @param netto
22 * @throws DALException
23 */
24 public ProductBatchCompDTO(int pb_id, int commoditybatch_id, int
    user_id,
25     double tara, double netto) throws DALException {
26     setPb_id(pb_id);
27     setCommoditybatch_id(commoditybatch_id);
28     setUser_id(user_id);
29     setTara(tara);
30     setNetto(netto);
31 }
32
33 public void setPb_id(int pb_id) throws DALException {
34     if(pb_id<=0){
35         throw new DALException("Produkt batch id'et skal være større end 0"
36             );
37     }
38     this.pb_id = pb_id;
39 }
40 public void setCommoditybatch_id(int commoditybatch_id) throws
    DALException {
41     if(commoditybatch_id<=0){
42         throw new DALException("Råvare batch id'et skal være større end 0")
43             ;
44     }
45     this.commoditybatch_id = commoditybatch_id;
46 }
47 public void setUser_id(int user_id) throws DALException {
48     if(user_id<=0){
49         throw new DALException("Bruger id'et skal være større end 0");
50     }
51     this.user_id = user_id;
52 }
53
54 public void setTara(double tara) throws DALException {
55     this.tara = tara;
56 }
```

```

57
58 public void setNetto(double netto) throws DALException {
59     this.netto = netto;
60 }
61
62 public int getPb_id() {
63     return pb_id;
64 }
65
66 public int getCommoditybatch_id() {
67     return commoditybatch_id;
68 }
69
70 public int getUser_id() {
71     return user_id;
72 }
73
74 public double getTara() {
75     return tara;
76 }
77
78 public double getNetto() {
79     return netto;
80 }
81
82 }

```

### ProductBatchData

```

1 package admin.data;
2
3 import java.sql.ResultSet;
4 import java.sql.SQLException;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class ProductBatchData implements IProductBatchDAO {
9
10     public synchronized void createProductBatch(ProductBatchDTO
        productBatch)
        throws DALException {
11         try {
12             Connector.connect();
13         } catch (Exception e1) {
14             throw new DALException(
15                 "Der kunne ikke oprettes forbindelse til databasen");
16         }
17         try {
18             Connector.doUpdate("INSERT INTO productbatch VALUES ( "
19                 + productBatch.getPbId() + ", "
20                 + productBatch.getPrescriptionId() + ", "
21                 + StatusType.getValue(productBatch.getStatus()) + ", '"
22                 + productBatch.getCreationDate() + "', "
23                 + productBatch.getUserId() + " );");
24         } catch (DALException e) {
25             throw new DALException(

```

```
27         "recept id'et findes ikke så det givne product batch kan ikke
           oprettes");
28     }
29     Connector.closeConnection();
30
31 }
32
33 @Override
34 public synchronized void deleteBatch(int pb_id) throws DALException {
35     try {
36         Connector.connect();
37     } catch (Exception e1) {
38         throw new DALException(
39             "Der kunne ikke oprettes forbindelse til databasen");
40     }
41     ResultSet rs = Connector.doQuery("SELECT * FROM productbatch "
42         + "WHERE pb_id in(SELECT pb_id from productbatchcomponent);");
43     try {
44         if (!rs.first()) {
45             Connector.doUpdate("DELETE FROM productbatch WHERE pb_id = "
46                 + pb_id + ";");
47             Connector.closeConnection();
48         } else {
49             throw new DALException(
50                 "Du kan ikke slette denne produkt batch da den allerede er på
                    begyndt");
51         }
52     } catch (SQLException e) {
53         throw new DALException(
54             "Der skete en fejl i ProductBatch i metoden deleteBatch(int
                    pb_id)"
55             + e.getMessage());
56     }
57 }
58
59 @Override
60 public synchronized List<ProductBatchDTO> getAllProductBatches()
61     throws DALException {
62     List<ProductBatchDTO> list = new ArrayList<ProductBatchDTO>();
63     try {
64         Connector.connect();
65     } catch (Exception e1) {
66         throw new DALException(
67             "Der kunne ikke oprettes forbindelse til databasen");
68     }
69     ResultSet rs = Connector.doQuery("SELECT * FROM productbatch;");
70     Connector.closeConnection();
71     try {
72         while (rs.next()) {
73             list.add(new ProductBatchDTO(rs.getInt("pb_id"), rs
74                 .getInt("prescription_id"), rs.getInt("status"), rs
75                 .getDate("current_date"), rs.getInt("user_id")));
76         }
77     } catch (SQLException e) {
78         throw new DALException(
```

```
79         "Der skete en fejl i ProductBatch i metoden
           getAllProductBatches() "
80         + e.getMessage());
81     }
82     return list;
83 }
84
85 @Override
86 public synchronized List<ProductBatchDTO> getListByOperator(int
    operatorId)
87     throws DALException {
88     List<ProductBatchDTO> list = new ArrayList<ProductBatchDTO>();
89     try {
90         Connector.connect();
91     } catch (Exception e1) {
92         throw new DALException(
93             "Der kunne ikke oprettes forbindelse til databasen");
94     }
95     ResultSet rs = Connector
96         .doQuery("SELECT * FROM productbatch WHERE pb_id IN "
97             + "(SELECT pb_id FROM productbatchcomponent WHERE user_id = "
98             + operatorId + " );");
99     Connector.closeConnection();
100    try {
101        while (rs.next()) {
102            list.add(new ProductBatchDTO(rs.getInt("pb_id"), rs
103                .getInt("prescription_id"), rs.getInt("status"), rs
104                .getDate("current_date"), rs.getInt("user_id")));
105        }
106    } catch (SQLException e) {
107        throw new DALException(
108            "Der skete en fejl i ProductBatch i metoden
              getCompletedProductBatch() "
109            + e.getMessage());
110    }
111    return list;
112 }
113
114 @Override
115 public synchronized List<ProductBatchDTO> getProductBatchByStatus(
    StatusType status) throws DALException {
116     List<ProductBatchDTO> list = new ArrayList<ProductBatchDTO>();
117     try {
118         Connector.connect();
119     } catch (Exception e1) {
120         throw new DALException(
121             "Der kunne ikke oprettes forbindelse til databasen");
122     }
123     ResultSet rs = Connector
124         .doQuery("SELECT * FROM productbatch WHERE status = "
125             + StatusType.getValue(status) + " ");
126     Connector.closeConnection();
127     try {
128         while (rs.next()) {
```

```
130         list.add(new ProductBatchDTO(rs.getInt("pb_id"), rs
131             .getInt("prescription_id"), rs.getInt("status"), rs
132             .getDate("current_date"), rs.getInt("user_id")));
133     }
134 } catch (SQLException e) {
135     throw new DALException(
136         "Der skete en fejl i ProductBatch i metoden
            getCompletedProductBatch() "
137         + e.getMessage());
138 }
139 return list;
140 }
141
142 @Override
143 public synchronized void updateProductBatch(ProductBatchDTO product)
144     throws DALException {
145     try {
146         Connector.connect();
147     } catch (Exception e1) {
148         throw new DALException(
149             "Der kunne ikke oprettes forbindelse til databasen");
150     }
151     Connector.doUpdate("UPDATE productbatch SET " +
152         "prescription_id = " + product.getPrescriptionId() +
153         ", status = " + product.getStatus().ordinal() +
154         ", user_id = " + product.getUserId() +
155         " WHERE pb_id = " + product.getPbId() + ";");
156     Connector.closeConnection();
157 }
158
159
160 @Override
161 public synchronized ProductBatchDTO getProductBatch(int id)
162     throws DALException {
163     try {
164         Connector.connect();
165     } catch (Exception e1) {
166         throw new DALException(
167             "Der kunne ikke oprettes forbindelse til databasen");
168     }
169     ResultSet rs = Connector
170         .doQuery("SELECT * FROM productbatch WHERE pb_id = " + id + ";");
171     try {
172         if (!rs.first()) {
173             throw new DALException(
174                 "Der er ikke noget productbatch med det id");
175         }
176         return new ProductBatchDTO(rs.getInt("pb_id"),
177             rs.getInt("prescription_id"), rs.getInt("status"),
178             rs.getDate("current_date"), rs.getInt("user_id"));
179     } catch (SQLException e) {
180         throw new DALException(
181             "Der skete en fejl i forbindelse med databasen "
182             + e.getMessage());
183     }
```



```
184     }
185
186 }

ProductBatchDTO

1  package admin.data;
2
3  import java.sql.Date;
4  import java.util.Calendar;
5
6  public class ProductBatchDTO {
7
8      private int pb_id, prescription_id, user_id;
9      private StatusType status;
10     // create a sql date object so we can use it in our INSERT statement
11     private Date creationDate = new Date(Calendar.getInstance().getTime()
12         .getTime());
13
14     /**
15      * Constructor with no parameters and no error checks
16      */
17     public ProductBatchDTO() {
18
19     }
20
21     /**
22      * Constructor with parameters from set methods and thus error checks
23      *
24      * @param pb_id
25      * @param prescription_id
26      * @param status
27      * @throws DALEException
28      */
29     public ProductBatchDTO(int pb_id, int prescription_id, int status,
30         Date date, int user_id) throws DALEException {
31         setPb_id(pb_id);
32         setPrescription_id(prescription_id);
33         setStatus(status);
34         setCreationDate(date);
35         setUserId(user_id);
36     }
37
38     public Date getCreationDate() {
39         return creationDate;
40     }
41
42     public void setCreationDate(Date currentDate) {
43         this.creationDate = currentDate;
44     }
45
46     public void setPb_id(int pb_id) throws DALEException {
47         if(pb_id <= 0){
48             throw new DALEException("produkt batch id'et skal være positivt");
49         }
50         this.pb_id = pb_id;
```

```
51     }
52
53     public void setPrescription_id(int prescription_id) throws DALErrorException
54     {
55         if(prescription_id <= 0){
56             throw new DALErrorException("Recept id'et skal være større end 0");
57         }
58         this.prescription_id = prescription_id;
59     }
60
61     public void setStatus(StatusType status) throws DALErrorException {
62         this.status = status;
63     }
64
65     public void setStatus(int status) throws DALErrorException {
66         this.status = StatusType.fromInt(status);
67     }
68
69     public int getPbId() {
70         return pb_id;
71     }
72
73     public int getPrescriptionId() {
74         return prescription_id;
75     }
76
77     public StatusType getStatus() {
78         return status;
79     }
80
81     public int getUserId() {
82         return user_id;
83     }
84
85     public void setUserId(int user_id) throws DALErrorException {
86         if(user_id <= 0){
87             throw new DALErrorException("bruger id'et skal være større end 0");
88         }
89         this.user_id = user_id;
90     }
```

#### ProductListItem

```
1 package admin.data;
2
3 public class ProductListItem {
4
5     public int commodityId = 0;
6     public String commodityName = "";
7
8     public double ammount = 0.0;
9     public double tolerance = 0.0;
10    public double tara = 0.0;
11    public double netto = 0.0;
12    public int commodityBatch = 0;
```

```
13
14  public String operator = "";
15 }
```

### StatusType

```
1  package admin.data;
2
3  public enum StatusType {
4      NEW, IN_PRODUCTION, PAUSED, FINISHED;
5
6      String[] uiNames = { "Oprettet",
7                          "Under Produktion",
8                          "På Pause",
9                          "Afluttet",
10     };
11
12     public static int getValue(StatusType type){
13         return type.ordinal();
14     }
15
16     public static StatusType fromInt(int i){
17         return StatusType.values()[i];
18     }
19
20     public String uiName(){
21         return uiNames[this.ordinal()];
22     }
23
24
25 }
```

### UserData

```
1  package admin.data;
2
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.util.ArrayList;
6  import java.util.List;
7
8  public class UserData implements IUserDAO {
9
10     public UserData() {
11         // if there are no users in the database, create a sysadmin
12         try {
13             Connector.connect();
14         } catch (Exception e1) {
15             e1.printStackTrace();
16         }
17         try {
18             ResultSet rs = Connector.doQuery("SELECT COUNT(*) FROM users");
19             Connector.closeConnection();
20             if (rs.next()) {
21                 if (!(rs.getInt(1) > 0))
22                     // creating users for testing

```

```
23         createDefaultUsers();
24     }
25     } catch (Exception e) {
26         e.getMessage();
27     }
28 }
29
30 @Override
31 public synchronized UserDTO getUser(int user_id) throws DALException {
32     try {
33         Connector.connect();
34     } catch (Exception e1) {
35         e1.printStackTrace();
36     }
37     try {
38
39         ResultSet rs = Connector
40             .doQuery("SELECT * FROM users WHERE user_id = " + user_id
41                 + ";");
42         Connector.closeConnection();
43         if (!rs.first()) {
44             throw new DALException("the user with user id: " + user_id
45                 + "does not exist");
46         }
47         return new UserDTO(rs.getInt("user_id"), rs.getString("user_name"),
48             rs.getString("ini"), rs.getString("cpr"),
49             rs.getString("password"), rs.getInt("user_type"));
50     } catch (SQLException e) {
51         throw new DALException(e);
52     }
53 }
54 }
55
56 @Override
57 public List<UserDTO> getUserList() throws DALException {
58     try {
59         Connector.connect();
60     } catch (Exception e1) {
61         e1.printStackTrace();
62     }
63     List<UserDTO> list = new ArrayList<UserDTO>();
64
65     ResultSet rs = Connector.doQuery("SELECT * FROM users;");
66     Connector.closeConnection();
67     try {
68         while (rs.next()) {
69             list.add(new UserDTO(rs.getInt("user_id"), rs
70                 .getString("user_name"), rs.getString("ini"), rs
71                 .getString("cpr"), rs.getString("password"), rs
72                 .getInt("user_type")));
73         }
74     } catch (SQLException e) {
75         throw new DALException(e);
76     }
77 }
```

```
78
79     return list;
80
81 }
82
83 @Override
84 public synchronized void createUser(UserDTO opr) throws DAException {
85     try {
86         Connector.connect();
87     } catch (Exception e1) {
88         e1.printStackTrace();
89     }
90     Connector.doUpdate("INSERT INTO users VALUES ('" + opr.getUserId()
91         + "','" + opr.getUsername() + "','" + opr.getIni()
92         + "','" + opr.getCpr() + "','" + opr.getPassword()
93         + "','" + opr.getAccesLevel() + "');");
94     Connector.closeConnection();
95
96 }
97
98 @Override
99 public synchronized void updateUser(UserDTO opr) throws DAException {
100     try {
101         Connector.connect();
102     } catch (Exception e1) {
103         e1.printStackTrace();
104     }
105     Connector.doUpdate("UPDATE users " + "SET user_name = " + "'"
106         + opr.getUsername() + "'" + ", ini = " + "'" + opr.getIni()
107         + "'" + ", cpr = " + "'" + opr.getCpr() + "'" + ", password = "
108         + "'" + opr.getPassword() + "'" + ", user_type = "
109         + opr.getAccesLevel() + " WHERE user_id = " + opr.getUserId()
110         + ";");
111     Connector.closeConnection();
112 }
113
114 @Override
115 public synchronized void deleteUser(int id) throws DAException {
116     try {
117         Connector.connect();
118     } catch (Exception e1) {
119         e1.printStackTrace();
120     }
121     ResultSet rs = Connector
122         .doQuery("SELECT * FROM users where user_id in ( SELECT user_id
123             from productbatchcomponent );");
124     try {
125         if (!rs.first()) {
126             Connector.doUpdate("DELETE FROM users WHERE user_id = " + id
127                 + ";");
128             Connector.closeConnection();
129         } else {
130             throw new DAException(
131                 "You cannot delete the user, because it has a
132                 productbatchcomponent attached to it's id");
133         }
134     }
135 }
```

```
131     }
132
133     } catch (SQLException e) {
134         throw new DALEXception(e);
135     }
136
137     // try {
138     //     Connector.connect();
139     // } catch (Exception e1) {
140     //     e1.printStackTrace();
141     // }
142     // Connector.doUpdate("UPDATE users SET user_type = 4 WHERE user_id
143     // = "
144     // + id + ";");
145     // Connector.closeConnection();
146 }
147
148 public synchronized void createDefaultUsers() {
149     try {
150         Connector.connect();
151     } catch (Exception e1) {
152         e1.printStackTrace();
153     }
154
155     try {
156         Connector
157             .doUpdate("INSERT INTO users VALUES(1,'sysAdmin', 'SM',
158                 '1234567890', 'Adminpw1', 0); ");
159         Connector
160             .doUpdate("INSERT INTO users VALUES(11,'Test Guy', 'TG',
161                 '1234567890', ''
162                 + UserDTO.generatePassword() + " ', 2);");
163         Connector
164             .doUpdate("INSERT INTO users VALUES(12,'Test Guy 2', 'TG2',
165                 '1234567890', ''
166                 + UserDTO.generatePassword() + " ', 3);");
167         Connector
168             .doUpdate("INSERT INTO users VALUES(13,'Test Guy 3', 'TG3',
169                 '1234567890', ''
170                 + UserDTO.generatePassword() + " ', 2);");
171         Connector
172             .doUpdate("INSERT INTO users VALUES(14,'Test Guy 4', 'TG4',
173                 '1234567890', ''
174                 + UserDTO.generatePassword() + " ', 3);");
175         Connector
176             .doUpdate("INSERT INTO users VALUES(15,'Test Guy 5', 'TG5',
177                 '1234567890', ''
178                 + UserDTO.generatePassword() + " ', 3);");
179         Connector
180             .doUpdate("INSERT INTO users VALUES(16,'Test Guy 6', 'TG6',
181                 '1234567890', ''
182                 + UserDTO.generatePassword() + " ', 1);");
183         Connector
184             .doUpdate("INSERT INTO users VALUES(10, 'Admin', 'AM',
185                 '1234567890', 'Adminpw1', 0); ");
186     }
187 }
```

```
177     Connector.closeConnection();
178 } catch (Exception e) {
179     System.out.println(e.getMessage());
180 }
181 }
182 }
```

### UserDTO

```
1 package admin.data;
2
3 import java.util.Random;
4
5 /**
6  * A data package containing info on an operator
7  */
8 public class UserDTO {
9     private int userId;
10    private String username;
11    private String ini;
12    private String cpr;
13    private String password;
14    private UserType type;
15
16    /**
17     * @param id
18     *         The id of the operator
19     * @param name
20     *         The name of the operator
21     * @param ini
22     *         The operators initials
23     * @param cpr
24     *         The social security number of the operator
25     * @param pw
26     *         The operator's password
27     */
28    public UserDTO(int id, String name, String ini, String cpr, String pw)
29        throws DALException {
30        this(id, name, ini, cpr, pw, 3);
31    }
32
33    public UserDTO(int id, String name, String ini, String cpr, String pw,
34        int access) throws DALException {
35        setUsername(name);
36        setIni(ini);
37        setCpr(cpr);
38        // setPassword(pw);
39        // på denne måde kan man oprette instanser af UserDTO med passwords
40        // der ikke overholder regler.
41        password = pw;
42        setAccessLevel(access);
43        setUserId(id);
44    }
45
46    public int getUserId() {
```

```
47     return userId;
48 }
49
50 public static boolean checkPassword(String password) {
51     return chkPassWithMsg(password) == null;
52 }
53
54 public static String chkPassWithMsg(String password) {
55     int i = 4;
56     String msg = "";
57     if (!password.matches(".*[0-9]+.*")) {
58         i--;
59         msg += "Password har ingen tal\n";
60     }
61     if (!password.matches(".*[a-z]+.*")) {
62         i--;
63         msg += "Password har ingen små bogstaver\n";
64     }
65     if (!password.matches(".*[A-Z]+.*")) {
66         i--;
67         msg += "Password har ingen store bogstaver\n";
68     }
69     if (!password.matches(".*[\\W_]+.*")) {
70         i--;
71         msg += "Password har ingen specialtegn\n";
72     }
73     if (password.length() < 5) {
74         i = 0;
75         msg += "Password er for kort (midre end 5 karakterer)";
76     }
77     if (password.length() > 8) {
78         i = 0;
79         msg += "Password er for langt (mrer end 8 karakterer)";
80     }
81     if (i >= 3)
82         return null;
83     else
84         return msg;
85 }
86
87 public static String generatePassword() {
88     String numbers = "0123456789";
89     String lowerCase = "abcdefghijklmnopqrstuvwxyz";
90     String upperCase = lowerCase.toUpperCase();
91     String specialChar = "!\"#$ %&/'()=_" ;
92     String legalchars = numbers + lowerCase + upperCase + specialChar;
93     String returnString = "";
94     returnString += lowerCase.charAt(new Random().nextInt(lowerCase
95         .length()));
96     returnString += numbers.charAt(new Random().nextInt(numbers.length())
97         );
98     returnString += upperCase.charAt(new Random().nextInt(upperCase
99         .length()));
100    returnString += specialChar.charAt(new Random().nextInt(specialChar
101        .length()));
```



```
101     while (returnString.length() < 7) {
102         returnString += legalchars.charAt(new Random().nextInt(legalchars
103             .length()));
104     }
105     return returnString;
106 }
107
108 public void setUserId(int id) throws DALException {
109     if (id < 1)
110         throw new DALException("ID skal være større end 0");
111     if (id > 99999999)
112         throw new DALException("ID skal være mindre end 99999999");
113     this.userId = id;
114 }
115
116 public String getUsername() {
117     return username;
118 }
119
120 public String getIni() {
121     return ini;
122 }
123
124 public String getCpr() {
125     return cpr;
126 }
127
128 public String getPassword() {
129     return password;
130 }
131
132 public boolean isAdmin() {
133     return type == UserType.ADMIN;
134 }
135
136 public int getAccesLevel() {
137     return type.ordinal();
138 }
139
140 public void setUsername(String newname) throws DALException {
141     if (newname.length() < 2)
142         throw new DALException("Operatør navn er for kort");
143     if (newname.length() > 20)
144         throw new DALException("Operatør navn er for langt");
145     this.username = newname;
146 }
147
148 public void setIni(String newini) throws DALException {
149     if (newini.length() < 2)
150         throw new DALException("Operatør initialer er for få");
151     if (newini.length() > 3)
152         throw new DALException("Operatør initialer er for mange");
153     this.ini = newini;
154 }
155
```

```
156 public void setCpr(String newcpr) throws DALException {
157     if (!newcpr.matches("\\d{10}"))
158         throw new DALException("CPR-nummer er ikke 10 cifre");
159     this.cpr = newcpr;
160 }
161
162 public void setPassword(String newpassword) throws DALException {
163     String msg = chkPassWithMsg(newpassword);
164     if (msg != null)
165         throw new DALException(msg);
166     this.password = newpassword;
167 }
168
169 public void setAccessLevel(int access) {
170     this.type = UserType.values()[access];
171 }
172
173 public void setAccessLevel(UserType type) {
174     this.type = type;
175 }
176
177 public UserType getUserType() {
178     return type;
179 }
180
181 }
```

### UserInfo

```
1 package admin.data;
2
3 public class UserInfo {
4     public int id;
5     public String ini;
6     public String name;
7     public String cpr;
8     public UserType access;
9     public String deleURL;
10    public String editURL;
11
12    public UserInfo() {
13
14    }
15
16    public UserInfo(UserDTO operator) {
17        id = operator.getId();
18        ini = operator.getIni();
19        name = operator.getUsername();
20        cpr = operator.getCpr();
21        access = operator.getUserType();
22        deleURL = "user_confirm_delete?id=" + id;
23        editURL = "user_edit?id=" + id;
24    }
25
26 }
```

**UserType**

```
1 package admin.data;
2
3 public enum UserType {
4     ADMIN, PHARMACIST, FOREMAN, OPERATOR, INACTIVE;
5
6     String[] uiNames = { "Administrator",
7         "Farmaceut",
8         "Værkfører",
9         "Operator",
10        "–deaktiveret–"
11    };
12
13     public static int getLevel(UserType type){
14         return type.ordinal();
15     }
16
17     public static UserType fromInt(int i){
18         return UserType.values()[i];
19     }
20
21     public String uiName(){
22         return uiNames[this.ordinal()];
23     }
24
25
26 }
```

**A.2.3 admin.test**

Koden i vores **admin** data pakke

**CommodityDataTest**

```
1 package admin.test;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.junit.After;
7 import org.junit.Assert;
8 import org.junit.Before;
9 import org.junit.Test;
10
11 import admin.data.CommodityDTO;
12 import admin.data.CommodityData;
13 import admin.data.DALException;
14
15 public class CommodityDataTest {
16
17     // Objects needed for testing
18     private CommodityDTO commodity100, commodity200, commodity300,
19         commodity400, commodity500, commodity600;
20     private CommodityData commodityData;
21 }
```

```
22  /**
23   * Sets up the entities needed for the test
24   *
25   * @throws Exception
26   */
27  @Before
28  public void setUp() throws Exception {
29      this.commodity100 = new CommodityDTO(100, "ethundrede", "lev100");
30      this.commodity200 = new CommodityDTO(200, "tohundrede", "lev200");
31      this.commodity300 = new CommodityDTO(300, "trehundrede", "lev300");
32      this.commodity400 = new CommodityDTO(400, "firehundrede", "lev400");
33      this.commodity500 = new CommodityDTO(500, "femhundrede", "lev500");
34      this.commodity600 = new CommodityDTO(600, "sekshundrede", "lev600");
35      this.commodityData = new CommodityData();
36  }
37
38  /**
39   * Cleans up what is used for testing
40   *
41   * @throws Exception
42   */
43  @After
44  public void tearDown() throws Exception {
45      this.commodity100 = new CommodityDTO(100, "ethundrede", "lev100");
46      this.commodity200 = new CommodityDTO(200, "tohundrede", "lev200");
47      this.commodity300 = new CommodityDTO(300, "trehundrede", "lev300");
48      this.commodity400 = new CommodityDTO(400, "firehundrede", "lev400");
49      this.commodity500 = new CommodityDTO(500, "femhundrede", "lev500");
50      this.commodity600 = new CommodityDTO(600, "sekshundrede", "lev600");
51      this.commodityData = new CommodityData();
52  }
53
54  /**
55   * Test of entities
56   */
57  @Test
58  public void testEntities() {
59      Assert.assertNotNull(this.commodity100);
60      Assert.assertNotNull(this.commodity200);
61      Assert.assertNotNull(this.commodity300);
62      Assert.assertNotNull(this.commodity400);
63      Assert.assertNotNull(this.commodity500);
64      Assert.assertNotNull(this.commodity600);
65  }
66
67  @Test
68  public void testGetCommodity() throws DAException {
69      commodityData.createCommodity(commodity200);
70      int expected = 200;
71      int actual = this.commodity200.getComId();
72      Assert.assertEquals(expected, actual);
73
74      // Perform the action to be tested
75      commodityData.getCommodity(actual);
76  }
```

```
77     expected = 200;
78     actual = commodityData.getCommodity(commodity200.getComId()).getComId
        ();
79     Assert.assertEquals(expected, actual);
80
81     // Delete from DB
82     commodityData.deleteCommodity(commodity200.getComId());
83 }
84
85 @Test
86 public void testGetComList() throws DAException {
87     // create list of users for testing
88     List<CommodityDTO> expected = new ArrayList<CommodityDTO>();
89
90     // Put users in DB
91     commodityData.createCommodity(commodity300);
92     commodityData.createCommodity(commodity400);
93
94     // Put users in List
95     expected.add(commodity300);
96     expected.add(commodity400);
97
98     List<CommodityDTO> actual = commodityData.getComList();
99
100    // Run through the list and check if elements match
101    for (int i = 0; i < expected.size(); i++) {
102        expected.equals(actual);
103    }
104    // Delete after use
105    commodityData.deleteCommodity(commodity300.getComId());
106    commodityData.deleteCommodity(commodity400.getComId());
107    expected.clear();
108    actual.clear();
109 }
110
111 @Test
112 public void testCreateCommodity() throws DAException {
113     int expected = 100;
114     int actual = this.commodity100.getComId();
115     Assert.assertEquals(expected, actual);
116
117     // Perform the action to be tested
118     commodityData.createCommodity(commodity100);
119
120     expected = 100;
121     actual = commodityData.getCommodity(commodity100.getComId()).getComId
        ();
122     Assert.assertEquals(expected, actual);
123
124     // Delete from DB
125     commodityData.deleteCommodity(commodity100.getComId());
126 }
127
128 @Test
129 public void testUpdateCommodity() throws DAException {
```

```

130     // Put user in the DB
131     commodityData.createCommodity(commodity500);
132
133     // expected and actual values before the test
134     String expected = "femhundrede";
135     String actual = this.commodity500.getComName();
136     Assert.assertEquals(expected, actual);
137
138     // Perform the action to be tested
139     commodity500.setCommodity_name("femfemhundrede");
140     commodityData.updateCommodity(commodity500);
141
142     // expected and actual values after the test
143     expected = "femfemhundrede";
144     actual = commodityData.getCommodity(commodity500.getComId())
145         .getComName();
146
147     Assert.assertEquals(expected, actual);
148     commodityData.deleteCommodity(commodity500.getComId());
149 }
150
151 @Test
152 public void testDeleteCommodity() throws DAException {
153     // create list of users for testing
154     List<CommodityDTO> expected = new ArrayList<CommodityDTO>();
155
156     // Put user in the DB
157     commodityData.createCommodity(commodity600);
158
159     // Put users in List
160     expected.add(commodity600);
161
162     // Perform the action to be tested
163     commodityData.deleteCommodity(commodity600.getComId());
164
165     // List after deletion
166     List<CommodityDTO> actual = commodityData.getComList();
167
168     // Run through the list and check if elements match
169     if (expected.isEmpty() && actual.isEmpty()) {
170         expected.equals(actual);
171     }
172
173     // clear lists after use
174     expected.clear();
175     actual.clear();
176 }
177
178 }

```

#### UserDataTest

```

1 package admin.test;
2
3 import java.util.ArrayList;
4 import java.util.List;

```

```
5
6 import org.junit.After;
7 import org.junit.Assert;
8 import org.junit.Before;
9 import org.junit.Test;
10
11 import admin.data.DALException;
12 import admin.data.UserDTO;
13 import admin.data.UserData;
14
15 public class UserDataTest {
16
17     // Objects needed for testing
18     private UserDTO user100, user200, user300, user400, user500, user600;
19     private UserData userData;
20
21     /**
22      * Sets up the entities needed for the test
23      *
24      * @throws Exception
25      */
26     @Before
27     public void setUp() throws Exception {
28         this.userData = new UserData();
29         this.user100 = new UserDTO(100, "UserTestGuy100", "U1", "1111111111",
30             "12345678", 1);
31         this.user200 = new UserDTO(200, "UserTestGuy200", "U2", "2222222222",
32             "Adminpw1", 0);
33         this.user300 = new UserDTO(300, "UserTestGuy300", "U3", "3333333333",
34             "12345678", 1);
35         this.user400 = new UserDTO(400, "UserTestGuy400", "U4", "4444444444",
36             "12345678", 1);
37         this.user500 = new UserDTO(500, "UserTestGuy500", "U5", "5555555555",
38             "12345678", 1);
39         this.user600 = new UserDTO(600, "UserTestGuy600", "U6", "6666666666",
40             "12345678", 1);
41     }
42
43     /**
44      * Cleans up what is used for testing
45      *
46      * @throws Exception
47      */
48     @After
49     public void tearDown() throws Exception {
50         this.userData = new UserData();
51         this.user100 = new UserDTO(100, "UserTestGuy100", "U1", "1111111111",
52             "12345678", 1);
53         this.user200 = new UserDTO(200, "UserTestGuy200", "U2", "2222222222",
54             "Adminpw1", 0);
55         this.user300 = new UserDTO(300, "UserTestGuy300", "U3", "3333333333",
56             "12345678", 1);
57         this.user400 = new UserDTO(400, "UserTestGuy400", "U4", "4444444444",
58             "12345678", 1);
59         this.user500 = new UserDTO(500, "UserTestGuy500", "U5", "5555555555",
```

```
60         "12345678", 1);
61     this.user600 = new UserDTO(600, "UserTestGuy600", "U6", "6666666666",
62         "12345678", 1);
63 }
64
65 /**
66  * Test of entities
67  */
68 @Test
69 public void testEntities() {
70     Assert.assertNotNull(this.user100);
71     Assert.assertNotNull(this.user200);
72     Assert.assertNotNull(this.user300);
73     Assert.assertNotNull(this.user400);
74     Assert.assertNotNull(this.user500);
75     Assert.assertNotNull(this.user600);
76 }
77
78 /**
79  * Method to test the creation of users in DB
80  *
81  * @throws DALException
82  */
83 @Test
84 public void testCreateUser() throws DALException {
85     int expected = 100;
86     int actual = this.user100.getUserId();
87     Assert.assertEquals(expected, actual);
88
89     // Perform the action to be tested
90     userData.createUser(user100);
91
92     expected = 100;
93     actual = userData.getUser(user100.getUserId()).getUserId();
94     Assert.assertEquals(expected, actual);
95
96     // Delete from DB
97     userData.deleteUser(user100.getUserId());
98 }
99
100 @Test
101 public void testGetUser() throws DALException {
102     userData.createUser(user200);
103     int expected = 200;
104     int actual = this.user200.getUserId();
105     Assert.assertEquals(expected, actual);
106
107     // Perform the action to be tested
108     userData.getUser(actual);
109
110     expected = 200;
111     actual = userData.getUser(user200.getUserId()).getUserId();
112     Assert.assertEquals(expected, actual);
113
114     // Delete from DB
```



```
115     userData.deleteUser(user200.getUserId());
116 }
117
118 @Test
119 public void testGetUserList() throws DALException {
120     // create list of users for testing
121     List<UserDTO> expected = new ArrayList<UserDTO>();
122
123     // Put users in DB
124     userData.createUser(user300);
125     userData.createUser(user400);
126
127     // Put users in List
128     expected.add(user300);
129     expected.add(user400);
130
131     List<UserDTO> actual = userData.getUserList();
132
133     // Run through the list and check if elements match
134     for (int i = 0; i < expected.size(); i++) {
135         expected.equals(actual);
136     }
137     // Delete after use
138     userData.deleteUser(user300.getUserId());
139     userData.deleteUser(user400.getUserId());
140     expected.clear();
141     actual.clear();
142 }
143
144 @Test
145 public void testUpdateUser() throws DALException {
146     // Put user in the DB
147     userData.createUser(user500);
148
149     // expected and actual values before the test
150     String expected = "5555555555";
151     String actual = this.user500.getCpr();
152     Assert.assertEquals(expected, actual);
153
154     // Perform the action to be tested
155     user500.setCpr("111111112");
156     userData.updateUser(user500);
157
158     // expected and actual values after the test
159     expected = "111111112";
160     actual = userData.getUser(user500.getUserId()).getCpr();
161
162     Assert.assertEquals(expected, actual);
163     userData.deleteUser(user500.getUserId());
164 }
165
166 @Test
167 public void testDeleteUser() throws DALException {
168     // create list of users for testing
169     List<UserDTO> expected = new ArrayList<UserDTO>();
```

```
170
171 // Put user in the DB
172 userData.createUser(user600);
173
174 // Put users in List
175 expected.add(user600);
176
177 // Perform the action to be tested
178 userData.deleteUser(user600.getUserId());
179
180 // List after deletion
181 List<UserDTO> actual = userData.getUserList();
182
183 // Run through the list an check if elements match
184 if (expected.isEmpty() && actual.isEmpty()) {
185     expected.equals(actual);
186 }
187
188 // clear lists after use
189 expected.clear();
190 actual.clear();
191 }
192
193 }
```

## A.3 simulator

Al kode fra simulator pakken

### A.3.1 simulator.boundary

Koden i vores **simulator** data pakke

#### GUI

```
1 package simulator.boundary;
2
3 import java.awt.BorderLayout;
4 import java.awt.Color;
5 import java.awt.GridLayout;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.awt.event.KeyEvent;
9 import java.awt.event.KeyListener;
10
11 import javax.swing.Box;
12 import javax.swing.BoxLayout;
13 import javax.swing.JButton;
14 import javax.swing.JFrame;
15 import javax.swing.JPanel;
16 import javax.swing.JTextPane;
17
18 import simulator.data.IProgramState;
19
20 /**
21  * Class to create the Graphical User Interface
22  *
23  * @author Gruppe 53
24  *
25  */
26 public class GUI implements IBoundary {
27
28     private long lastRefresh = 0;
29     IProgramState programState;
30
31     // set up frames buttons and panels
32     JFrame f = new JFrame("Vægtsimulator");
33     JPanel viewPanel = new JPanel();
34     JPanel buttonPanel = new JPanel();
35     JPanel mainPanel = new JPanel();
36     JPanel taraPanel = new JPanel();
37     JButton b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, clear, enter, tara,
        cancel;
38     JTextPane toweight, fromweight, digits;
39     // JSpinner enterWeight = new JSpinner();
40     JTextPane enterWeight = new JTextPane();
41     EventHandler handler = new EventHandler();
42
43     @Override
44     public void closeResources() {
```

```
45     f.dispose();
46 }
47
48 /**
49  * Constructor that creates GUI components and their attributes
50  *
51  * @param programState
52  *         is used when a client changes the program state
53  */
54 public GUI(IProgramState programState) {
55     this.programState = programState;
56
57     // define components
58     b0 = new JButton("0");
59     b1 = new JButton("1");
60     b2 = new JButton("2");
61     b3 = new JButton("3");
62     b4 = new JButton("4");
63     b5 = new JButton("5");
64     b6 = new JButton("6");
65     b7 = new JButton("7");
66     b8 = new JButton("8");
67     b9 = new JButton("9");
68     clear = new JButton("CLEAR");
69     enter = new JButton("ENTER");
70     tara = new JButton("<T>");
71     cancel = new JButton("CANCEL");
72     toweight = new JTextPane();
73     fromweight = new JTextPane();
74     digits = new JTextPane();
75     enterWeight = new JTextPane();
76     // enterWeight = new JSpinner(new SpinnerNumberModel(0.0, 0.0,
77     // 100000.0,
78     // 1.0));
79
80     // define attributes on components
81     clear.setToolTipText("Push to clear input");
82     enter.setToolTipText("Push to send");
83     tara.setToolTipText("Push to tara weight");
84     cancel.setToolTipText("Push to cancel");
85     toweight.setBackground(Color.black);
86     toweight.setForeground(Color.green);
87     toweight.setFocusable(false);
88     toweight.setToolTipText("Upper display");
89     fromweight.setBackground(Color.black);
90     fromweight.setForeground(Color.green);
91     fromweight.setToolTipText("Bottom display");
92     fromweight.setFocusable(false);
93     digits.setBackground(Color.black);
94     digits.setForeground(Color.green);
95     digits.setFocusable(true);
96     digits.setToolTipText("Displays input from numpad on the weight");
97     enterWeight.setToolTipText("Enter brutto weight here");
98     enterWeight.setBackground(Color.black);
99     enterWeight.setForeground(Color.green);
```

```
99
100 // set layoutmanagers
101 viewPanel.setLayout(new BorderLayout(viewPanel, BorderLayout.Y_AXIS));
102 buttonPanel.setLayout(new GridLayout(4, 4));
103 mainPanel.setLayout(new BorderLayout());
104 taraPanel.setLayout(new BorderLayout(taraPanel, BorderLayout.X_AXIS));
105
106 // add components
107 buttonPanel.add(b1);
108 buttonPanel.add(b2);
109 buttonPanel.add(b3);
110 buttonPanel.add(b4);
111 buttonPanel.add(b5);
112 buttonPanel.add(b6);
113 buttonPanel.add(b7);
114 buttonPanel.add(b8);
115 buttonPanel.add(b9);
116 buttonPanel.add(clear);
117 buttonPanel.add(b0);
118 buttonPanel.add(enter);
119 mainPanel.add(buttonPanel, BorderLayout.EAST);
120 mainPanel.add(viewPanel, BorderLayout.CENTER);
121 mainPanel.add(enterWeight, BorderLayout.NORTH);
122 mainPanel.add(taraPanel, BorderLayout.SOUTH);
123 viewPanel.add(digits);
124 viewPanel.add(Box.createVerticalStrut(2));
125 viewPanel.add(toweight);
126 viewPanel.add(Box.createVerticalStrut(2));
127 viewPanel.add(fromweight);
128 viewPanel.add(Box.createVerticalStrut(2));
129 // viewPanel.add(taraPanel);
130 taraPanel.add(cancel);
131 taraPanel.add(tara);
132 taraPanel.add(enterWeight);
133
134 // add eventhandling
135 b0.addActionListener(handler);
136 b1.addActionListener(handler);
137 b2.addActionListener(handler);
138 b3.addActionListener(handler);
139 b4.addActionListener(handler);
140 b5.addActionListener(handler);
141 b6.addActionListener(handler);
142 b7.addActionListener(handler);
143 b8.addActionListener(handler);
144 b9.addActionListener(handler);
145 enter.addActionListener(handler);
146 clear.addActionListener(handler);
147 cancel.addActionListener(handler);
148 tara.addActionListener(handler);
149 // enterWeight.addChangeListener(handler);
150 enterWeight.addKeyListener(handler);
151
152 }
153
```

```

154  @Override
155  public void run() {
156
157      f.setVisible(true);
158
159      while (programState.isRunning()) {
160          if (programState.hasDisplayUpdated(lastRefresh - 10)) {
161              lastRefresh = System.currentTimeMillis();
162              printGui();
163          }
164          try {
165              this.wait(100);
166          } catch (Exception e) {
167              }
168      }
169  }
170
171  /**
172   * Method to print the GUI on screen
173   */
174  public void printGui() {
175
176      String adress = "null";
177      try {
178          adress = programState.getAddress().getHostAddress();
179      } catch (Exception e) {
180      }
181
182      toweight.setText("*****\n
183          + "Netto: " + programState.getNet() + " kg\n"
184          + "Instruktionsdisplay: " + programState.getDisplayText()
185          + "\n*****\n\n"
186          + "Debug info: \n" + "Hooked up to " + adress + "\nBrutto: "
187          + programState.getGross() + " kg" + "\nStreng modtaget: "
188          + programState.getNetString()
189          + "\n\nDenne vægt simulator lytter på ordrene "
190          + "\nD, DW, S, T, B, Q , P111 og RM20_8 "
191          + "\nPå kommunikationsporten\n" + "*****\n"
192          + "Tast T for tara\n"
193          + "Tast værdi nederst for ny brutto (svarende til at "
194          + "belastningen på vægt ændres)\n"
195          + "Klik på \"x\" i hjørnet for at afslutte program program\n");
196      fromweight.setText(programState.getBotDisplay());
197      digits.setText("");
198
199      // add mainPanel to Contentpane
200      f.getContentPane().add(mainPanel);
201      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
202      f.setSize(800, 600); // set to SVGA res
203      f.setVisible(true);
204      f.repaint();
205
206  }
207

```

```
208  /**
209  * Class to handle events in GUI
210  *
211  * @author Gruppe 53
212  *
213  */
214  private class Eventhandler implements ActionListener, KeyListener {
215
216      @Override
217      public void actionPerformed(ActionEvent e) {
218
219          String inputtext = digits.getText();
220
221          if (e.getSource() == b0) {
222              digits.setText(inputtext + b0.getText());
223          } else if (e.getSource() == b1) {
224              digits.setText(inputtext + b1.getText());
225          } else if (e.getSource() == b2) {
226              digits.setText(inputtext + b2.getText());
227          } else if (e.getSource() == b3) {
228              digits.setText(inputtext + b3.getText());
229          } else if (e.getSource() == b4) {
230              digits.setText(inputtext + b4.getText());
231          } else if (e.getSource() == b5) {
232              digits.setText(inputtext + b5.getText());
233          } else if (e.getSource() == b6) {
234              digits.setText(inputtext + b6.getText());
235          } else if (e.getSource() == b7) {
236              digits.setText(inputtext + b7.getText());
237          } else if (e.getSource() == b8) {
238              digits.setText(inputtext + b8.getText());
239          } else if (e.getSource() == b9) {
240              digits.setText(inputtext + b9.getText());
241          } else if (e.getSource() == clear) {
242              digits.setText("");
243              inputtext = digits.getText();
244          } else if (e.getSource() == tara) {
245              programState.tare();
246          } else if (e.getSource().equals(enter)) {
247              programState.setUserInput(digits.getText());
248              programState.setConfirmed(true);
249              digits.setText("");
250          } else if (e.getSource() == cancel) {
251              programState.setConfirmed(false);
252              digits.setText("");
253              programState.setUserInput("");
254          }
255      }
256
257      @Override
258      public void keyTyped(KeyEvent e) {
259      }
260
261      @Override
262      public void keyPressed(KeyEvent e) {
```

```

263         if (e.getSource() == enterWeight) {
264             if (e.getKeyCode() == KeyEvent.VK_ENTER)
265                 try {
266                     programState.setGross(Double.parseDouble(enterWeight
267                         .getText()));
268                 } catch (Exception ex) {
269
270                 }
271             }
272
273     }
274
275     @Override
276     public void keyReleased(KeyEvent e) {
277         // TODO Auto-generated method stub
278
279     }
280
281     // @Override
282     // public void stateChanged(ChangeEvent ce) {
283     //
284     //     if (ce.getSource() == enterWeight) {
285     //         programState.setGross(Double.parseDouble(enterWeight.getValue()
286     //             .toString()));
287     //     }
288     // }
289 }
290 }

```

### IBoundary

```

1  package simulator.boundary;
2
3  /**
4   * Interface with methods for all Boundary classes
5   *
6   * @author Gruppe 53
7   *
8   */
9  public interface IBoundary extends Runnable {
10
11      /**
12       *
13       * The method to run the input from the user thread.
14       *
15       */
16      public void run();
17
18      /**
19       * closes resources used by the boundary
20       */
21      public void closeResources();
22
23 }

```

### InputBoundary



```
1 package simulator.boundary;
2
3 import java.io.BufferedReader;
4 import java.io.InputStreamReader;
5
6 import simulator.data.IProgramState;
7
8 /**
9  * Class to take input from a client
10  *
11  * @author Gruppe 53
12  *
13  */
14 public class InputBoundary implements IBoundary {
15
16     BufferedReader consoleReader;
17     IProgramState programState;
18
19     /**
20      * Constuctor that makes the program ready for user inputs
21      *
22      * @param in
23      *         is a scanner that recieves keyboardinput
24      * @param programState
25      *         changes state of the program depending on the input
26      */
27     public InputBoundary(IProgramState programState) {
28         this.programState = programState;
29         consoleReader = new BufferedReader(new InputStreamReader(System.in));
30     }
31
32     @Override
33     public void closeResources() {
34         try {
35             consoleReader.close();
36             System.in.close();
37         } catch (Exception e) {
38         }
39     }
40
41     @Override
42     public void run() {
43         while (programState.isRunning()) {
44             try {
45                 if (!consoleReader.ready())
46                     continue;
47                 String userInput = consoleReader.readLine();
48                 if (userInput.equalsIgnoreCase("T")) {
49                     programState.tare();
50                 } else if (userInput.equalsIgnoreCase("B")) {
51                     System.out.println("Indtast brutto vægt.");
52                     while (!consoleReader.ready()) {
53                         if (!programState.isRunning())
54                             return;
55                     }
56                 }
57             }
58         }
59     }
60 }
```

```

56         userInput = consoleReader.readLine();
57         try {
58             programState.setGross(Double.parseDouble(userInput));
59         } catch (Exception e) {
60             System.out.println("Indtastet kan ikke genkendes "
61                 + "som tal.");
62         }
63     } else if (userInput.equalsIgnoreCase("Q")) {
64         programState.quit();
65         return;
66     } else
67         System.out.println("user input :\" + userInput + "\"");
68     programState.setUserInput(userInput);
69 } catch (Exception e) {
70 }
71 }
72 }
73 }

```

### NetworkIOBoundary

```

1  package simulator.boundary;
2
3  import java.io.BufferedReader;
4  import java.io.DataOutputStream;
5  import java.io.InputStreamReader;
6  import java.net.ServerSocket;
7  import java.net.Socket;
8
9  import simulator.data.IProgramState;
10
11 /**
12  * Class used to send/receive network input and output
13  *
14  * @author Gruppe 53
15  *
16  */
17 public class NetworkIOBoundary implements IBoundary {
18     private ServerSocket listener;
19     private Socket sock;
20     private DataOutputStream outstream;
21     private BufferedReader instream;
22     private IProgramState programState;
23     private boolean needResponse = false;
24     private long lastRequest;
25
26     public NetworkIOBoundary(IProgramState programState) {
27         this.programState = programState;
28     }
29
30     @Override
31     public void closeResources() {
32         try {
33             // lukning af serversocket i tilfælde af at vi stadig venter på
34             // connection
35             listener.close();

```

```
36     } catch (Exception e) {
37     }
38     try {
39         // lukning af socket
40         sock.close();
41         instream.close();
42         outstream.close();
43     } catch (Exception e) {
44     }
45 }
46
47 @Override
48 public void run() {
49
50     // Start af Socket
51     try {
52         listener = new ServerSocket(programState.getPort());
53         System.out.println("Venter på connection på port "
54             + programState.getPort());
55         System.out.println("Indtast eventuel portnummer som 1. argument");
56         System.out.println("på kommando linien for andet portnr");
57         sock = listener.accept();
58         // indtil videre regner vi kun med n forbindelse.
59         listener.close();
60         programState.setAddress(sock.getInetAddress());
61         instream = new BufferedReader(new InputStreamReader(
62             sock.getInputStream()));
63         outstream = new DataOutputStream(sock.getOutputStream());
64     } catch (Exception e) {
65         if (programState.isRunning())
66             System.out.println("Exception: " + e.getMessage());
67         programState.quit();
68         return;
69     }
70
71     // håndtering af input fra socket
72     try {
73         String netString;
74         while (programState.isRunning()) {
75
76             if (needResponse && programState.haveNewUserInput(lastRequest)) {
77                 String prefix = programState.getPrefix();
78                 if (programState.getConfirmed()) {
79                     outstream.writeBytes(prefix + " A \n"
80                         + programState.getUserInput() + "\\r\\n");
81                     needResponse = false;
82                 } else {
83                     if (prefix.equals("RM20"))
84                         outstream.writeBytes("RM20 C\\r\\n");
85                     else
86                         outstream.writeBytes("RM30 A 6\\r\\n");
87                 }
88             }
89
90             if (!instream.ready())
```

```
91         continue;
92
93     if ((netString = instream.readLine().toUpperCase()).isEmpty())
94         continue;
95
96     programState.setNetString(netString);
97
98     if (netString.startsWith("RM20")) {
99         programState.setPrefix("RM20");
100         String argString;
101         String args[];
102         try {
103             argString = netString.substring(8);
104             args = argString.split("\\ \\");
105         } catch (Exception e) {
106             outstream.writeBytes("RM20 L\r\n");
107             continue;
108         }
109         // if (args.length == 3 && args[2].equals("&3\\")) {
110         if (args.length == 3) {
111             outstream.writeBytes("RM20 B\r\n");
112             programState.setDisplayText(args[0]);
113             needResponse = true;
114             lastRequest = System.currentTimeMillis();
115         } else {
116             outstream.writeBytes("RM20 L\r\n");
117             outstream.writeBytes("Programmet understøtter kun "
118                 + "følgende version af RM20_8:\r\n");
119             outstream.writeBytes("RM20_8_\"<display text>\"_\"
120                 + "<placeholder text>\"_\"&3\\\"r\n");
121         }
122     } else if (netString.startsWith("RM39 1")) {
123         programState.setPrefix("RM30");
124         needResponse = true;
125     } else if (netString.startsWith("RESET")) {
126         programState.reset();
127         outstream.writeBytes("du har nulstillet programmet\r\n");
128     } else if (netString.startsWith("P111")) {
129         if (netString.length() <= 37) {
130             programState.setBotDisplay(netString.substring(6,
131                 netString.length()));
132             outstream.writeBytes("P111 A\r\n");
133         } else {
134             outstream.writeBytes("P111 L\r\n");
135         }
136     } else if (netString.startsWith("P110")) {
137         programState.setBotDisplay("");
138         outstream.writeBytes("P110 A \r\n");
139     } else if (netString.equals("DW")) {
140         programState.setDisplayText("");
141         outstream.writeBytes("DW A\r\n");
142     } else if (netString.startsWith("D ")) {
143         if (!netString.matches("D \\\".+\\\"")) {
144             outstream.writeBytes("D L\r\n");
145             continue;
```

```

146     }
147     int end = netString.indexOf("\n", 3);
148     netString = netString.substring(3, end);
149     programState.setDisplayText(netString);
150     ostream.writeBytes("D A\r\n");
151 } else if (netString.equals("T")) {
152     programState.tare();
153     ostream.writeBytes("T S " + (programState.getGross())
154         + " kg " + "\r\n");
155 } else if (netString.equals("S")) {
156     ostream.writeBytes("S S " + (programState.getNet())
157         + " kg " + "\r\n");
158 } else if (netString.startsWith("B ")) {
159     try {
160         double weight = Double.parseDouble(netString
161             .substring(2));
162         programState.setGross(weight);
163         ostream.writeBytes("B A\r\n");
164     } catch (Exception e) {
165         ostream.writeBytes("B L\r\n");
166     }
167 } else if (netString.equals("Q")) {
168     System.out
169         .println("Program stoppet Q modtaget p  com port");
170     ostream.writeBytes("program stoppet Q modtaget på com "
171         + "port");
172     programState.quit();
173 }
174 }
175 } catch (Exception e) {
176     System.out.println("Exception: " + e.getMessage());
177 }
178 }
179 }
180 }

```

### OutputBoundary

```

1 package simulator.boundary;
2
3 import simulator.data.IProgramState;
4
5 public class OutputBoundary implements IBoundary {
6
7     private long lastRefresh = 0;
8     private IProgramState programState;
9
10    public OutputBoundary(IProgramState programState) {
11        this.programState = programState;
12    }
13
14    @Override
15    public void closeResources() {
16        try {
17            System.out.close();
18        } catch (Exception e) {

```

```

19     }
20 }
21
22 @Override
23 public void run() {
24     while (programState.isRunning()) {
25         if (programState.hasDisplayUpdated(lastRefresh)) {
26             lastRefresh = System.currentTimeMillis();
27             printmenu();
28         }
29         try {
30             this.wait(100);
31         } catch (Exception e) {
32         }
33     }
34 }
35
36 /**
37  * Method to print console menu
38  */
39 public void printmenu() {
40     // for (int i = 0; i < 25; i++)
41     // System.out.println(" ");
42
43     String adress = "null";
44     try {
45         adress = programState.getAddress().getHostAddress();
46     } catch (Exception e) {
47     }
48
49     System.out.println("*****");
50     System.out.println("Netto: " + programState.getNet() + " kg");
51     System.out.println("Instruktionsdisplay: "
52         + programState.getDisplayText());
53     System.out.println("*****");
54     System.out.println(" ");
55     System.out.println(" ");
56     System.out.println("Debug info: ");
57     System.out.println("Hooked up to " + adress);
58     System.out.println("Brutto: " + programState.getGross() + " kg");
59     System.out.println("Streng modtaget: " + programState.getNetString())
60     ;
61     System.out.println(" ");
62     System.out.println("Denne vægt simulator lytter på ordrene ");
63     System.out.println("D, DN, S, T, B, Q ");
64     System.out.println("På kommunikationsporten ");
65     System.out.println("*****");
66     System.out.println("Tast T for tara (svarende til knaptryk på vægt)");
67     ;
68     System.out
69         .println("Tast B for ny brutto (svarende til at belastningen "
70             + "på vægt ændres)");
71     System.out.println("Tast Q for at afslutte program program");

```

```

70     System.out.println("Indtast (T/B/Q for knaptryk / brutto ændring / "
71         + "quit)");
72     System.out.print("Tast her: ");
73 }
74 }

```

### A.3.2 simulator.controller

Koden i vores **simulator** controller pakke

#### ConnectionSetup

```

1  package simulator.controller;
2
3  import java.net.ServerSocket;
4  import java.net.Socket;
5
6  import simulator.data.IProgramState;
7
8  /**
9   * Class to create Socket connection
10  *
11  * @author Gruppe 53
12  *
13  */
14  public class ConnectionSetup implements Runnable {
15      ServerSocket listener;
16      Socket socket;
17      IProgramState programState;
18
19      ConnectionSetup(ServerSocket listenSocket, IProgramState programState)
20      {
21          listener = listenSocket;
22          this.programState = programState;
23      }
24
25      @Override
26      public void run() {
27          try {
28              System.out.println("Venter på connection på port "
29                  + programState.getPort());
30              System.out.println("Indtast eventuel portnummer som 1. argument");
31              System.out.println("på kommando linien for andet portnr");
32              socket = listener.accept();
33              // indtil videre regner vi kun med n forbindelse.
34              listener.close();
35              programState.setAddress(socket.getInetAddress());
36          } catch (Exception e) {
37              if (programState.isRunning())
38                  System.out.println("Exception: " + e.getMessage());
39              programState.quit();
40              return;
41          }
42      }
43

```

```
44  /**
45   * Method to get the socket
46   *
47   * @return the socket
48   */
49  public Socket getSocket() {
50      return socket;
51  }
52
53 }
```

### Main

```
1  package simulator.controller;
2
3  import java.io.IOException;
4
5  import simulator.boundary.GUI;
6  import simulator.boundary.IBoundary;
7  import simulator.boundary.InputBoundary;
8  import simulator.boundary.NetworkIOBoundary;
9  import simulator.boundary.OutputBoundary;
10 import simulator.data.IProgramState;
11 import simulator.data.ProgramState;
12
13 /**
14  * The Main class that runs the program
15  *
16  * @author Gruppe 53
17  *
18  */
19 public class Main {
20
21     /**
22      * Runs the program
23      *
24      * @param args
25      * @throws IOException
26      *         thrown if port number is invalid
27      */
28     public static void main(String[] args) throws IOException {
29         int portdst;
30         IProgramState programState = new ProgramState();
31
32         if (args.length > 0)
33             try {
34                 portdst = Integer.parseInt(args[0]);
35             } catch (NumberFormatException e) {
36                 System.out.println("Port argument ugyldigt. "
37                     + "Bruger default 8000.");
38                 portdst = 8000;
39             }
40         else {
41             System.out.println("Bruger default port 8000.");
42             portdst = 8000;
43         }
44     }
45 }
```



```
44
45 // input setup and run
46 IBoundary input = new InputBoundary(programState);
47 Thread inputThread = new Thread(input);
48 inputThread.start();
49
50 // output setup and run
51 IBoundary output = new OutputBoundary(programState);
52 Thread outputThread = new Thread(output);
53 outputThread.start();
54
55 // gui setup and run
56 GUI gui = new GUI(programState);
57 Thread guiThread = new Thread(gui);
58 guiThread.start();
59
60 // Network setup and run
61 programState.setPort(portdst);
62 NetworkIOBoundary network = new NetworkIOBoundary(programState);
63 Thread networkThread = new Thread(network);
64 networkThread.start();
65
66 // wait until program is closed
67 while (programState.isRunning()) {
68     try {
69         Thread.sleep(100);
70     } catch (Exception e) {
71     }
72 }
73
74 network.closeResources();
75 input.closeResources();
76 output.closeResources();
77 gui.closeResources();
78 // System.exit(0);
79
80 }
81 }
```

### A.3.3 simulator.data

Koden i vores **simulator** data pakke

#### IProgramState

```
1 package simulator.data;
2
3 import java.net.InetAddress;
4
5 /**
6  * Interface with methods for ProgramState classes
7  *
8  * @author Gruppe 53
9  *
10 */
11 public interface IProgramState {
12     /**
13      *
14      * Sets the bottom display to show in the GUI recieves a string from
15      * the
16      * p111 command in the console
17      *
18      * @param botDisplay
19      */
20     public void setBotDisplay(String botDisplay);
21
22     /**
23      *
24      * return the bottom display that is currently stored in the GUI as a
25      * String
26      *
27      * @return
28      */
29     public String getBotDisplay();
30
31     /**
32      * Stores the latest string received from the network
33      *
34      * @param netString
35      *         the string to be stored
36      */
37     public void setNetString(String netString);
38
39     /**
40      * Fetch the latest string received from the network
41      *
42      * @return the stored string
43      */
44     public String getNetString();
45
46     /**
47      * Are there any updates to the information used for the display
48      *
49      * @param since
```

```
48     *           the system time in milliseconds from when we're
49     *           interested in
50     *           updates
51     * @return true if there are updates more recent than the specified
52     *           time
53     */
54     public boolean hasDisplayUpdated(Long since);
55     /**
56     * Are there any new user input
57     * @param since
58     *           the system time in milliseconds from when we're
59     *           interested in
60     *           input
61     * @return true if there are input more recent than the specified time
62     */
63     public boolean haveNewUserInput(Long since);
64     /**
65     * Stores the port the program is listening on
66     * @param port
67     *           the port number to be stored
68     */
69     public void setPort(int port);
70     /**
71     * Get the port number the program is listening on
72     * @return the port number the program is listening on
73     */
74     public int getPort();
75     /**
76     * Store the address of the connected client
77     * @param address
78     *           the address of the connected client
79     */
80     public void setAddress(InetAddress address);
81     /**
82     * retrieve the address of the connected client
83     * @return address the address of the connected client
84     */
85     public InetAddress getAddress();
86     /**
87     * Sets the tare weight to current load on the scale
88     * @param tweight
89     */
90     public void tare();
```

```
100
101  /**
102   * set the current load on the weight
103   *
104   * @param weight
105   */
106  public void setGross(double weight);
107
108  /**
109   * @return the current load on the scale
110   */
111  public double getGross();
112
113  /**
114   * @return the net weight on the scale (gross - tare)
115   */
116  public double getNet();
117
118  /**
119   * @return the current text on the display
120   */
121  public String getDisplayText();
122
123  /**
124   * Set the text on the scale display
125   *
126   * @param text
127   *         the text to be set on the display
128   */
129  public void setDisplayText(String text);
130
131  /**
132   * Retrieve the latest stored user input
133   *
134   * @return the last string that was stored as user input
135   */
136  public String getUserInput();
137
138  /**
139   * Store user input for later use
140   *
141   * @param text
142   *         the string to be stored
143   */
144  public void setUserInput(String text);
145
146  /**
147   * reset the display text, load and tare weight
148   */
149  public void reset();
150
151  /**
152   * set the program state to close the program
153   */
154  public void quit();
```

```
155
156  /**
157   * Tests if the program is still (supposed to be) running
158   *
159   * @return <code>true</code> until <code>quit()</code> has been
        executed and
160   *         <code>false</code> after <code>quit()</code> has been
        executed.
161   */
162  public boolean isRunning();
163
164  public void setConfirmed(boolean confirmed);
165
166  public boolean getConfirmed();
167
168  public void setPrefix(String prefix);
169
170  public String getPrefix();
171 }
```

### ProgramState

```
1  package simulator.data;
2
3  import java.net.InetAddress;
4
5  /**
6   * Class used to change the program state
7   *
8   * @author Gruppe 53
9   *
10  */
11  public class ProgramState implements IProgramState {
12      double tare = 0;
13      double gross = 0;
14      String display = "";
15      String netString = "";
16      String userInput = "";
17      String botDisplay = "";
18      String prefix = "";
19
20      int port = 0;
21      InetAddress address;
22
23      boolean confirmed;
24      boolean exit = false;
25      long lastUpdate = 0;
26      long lastInput = 0;
27
28      public ProgramState() {
29          lastUpdate = System.currentTimeMillis();
30      }
31
32      @Override
33      public void setBotDisplay(String botDisplay) {
34          this.botDisplay = botDisplay;
35      }
36  }
```

```
35     lastUpdate = System.currentTimeMillis();
36 }
37
38 @Override
39 public String getBotDisplay() {
40     return botDisplay;
41 }
42
43 @Override
44 public void setNetString(String netString) {
45     this.netString = netString;
46     lastUpdate = System.currentTimeMillis();
47 }
48
49 @Override
50 public String getNetString() {
51     return netString;
52 }
53
54 @Override
55 public boolean hasDisplayUpdated(Long since) {
56     return lastUpdate > since;
57 }
58
59 @Override
60 public void setPort(int port) {
61     this.port = port;
62     lastUpdate = System.currentTimeMillis();
63 }
64
65 @Override
66 public int getPort() {
67     return port;
68 }
69
70 @Override
71 public void setAddress(InetAddress address) {
72     this.address = address;
73     lastUpdate = System.currentTimeMillis();
74 }
75
76 @Override
77 public InetAddress getAddress() {
78     return address;
79 }
80
81 @Override
82 public boolean isRunning() {
83     return !exit;
84 }
85
86 @Override
87 public void tare() {
88     tare = gross;
89     lastUpdate = System.currentTimeMillis();
```

```
90     }
91
92     @Override
93     public void setGross(double weight) {
94         gross = weight;
95         lastUpdate = System.currentTimeMillis();
96     }
97
98     @Override
99     public double getGross() {
100         return gross;
101     }
102
103     @Override
104     public double getNet() {
105         return gross - tare;
106     }
107
108     @Override
109     public String getDisplayText() {
110         return display;
111     }
112
113     @Override
114     public void setDisplayText(String text) {
115         display = text;
116         lastUpdate = System.currentTimeMillis();
117     }
118
119     @Override
120     public void reset() {
121         gross = 0;
122         tare = 0;
123         display = "";
124         lastUpdate = System.currentTimeMillis();
125     }
126
127     @Override
128     public void quit() {
129         exit = true;
130     }
131
132     @Override
133     public boolean haveNewUserInput(Long since) {
134         return lastInput > since;
135     }
136
137     @Override
138     public String getUserInput() {
139         return userInput;
140     }
141
142     @Override
143     public void setUserInput(String text) {
144         userInput = text;
```

```
145     lastInput = System.currentTimeMillis();
146 }
147
148 @Override
149 public void setConfirmed(boolean confirmed) {
150     this.confirmed = confirmed;
151 }
152
153 @Override
154 public boolean getConfirmed() {
155
156     return confirmed;
157 }
158
159 @Override
160 public void setPrefix(String prefix) {
161     // TODO Auto-generated method stub
162
163 }
164
165 @Override
166 public String getPrefix() {
167     return prefix;
168 }
169
170 }
```



### A.3.4 simulator.test

Koden i vores **simulator** test pakke

#### MainTest

```
1 package simulator.test;
2
3 public class MainTest {
4
5     public static void main(String[] args) {
6
7         Test test = new Test();
8
9         test.run();
10
11     }
12
13
14
15
16
17 }
```

#### Test

```
1 package simulator.test;
2
3 import java.io.BufferedReader;
4 import java.io.DataOutputStream;
5 import java.io.InputStreamReader;
6 import java.net.Socket;
7
8 public class Test implements Runnable {
9
10     @Override
11     public void run() {
12
13         String sentence;
14         String modifiedSentence;
15         try {
16
17             Socket clientSocket = new Socket("localhost", 8000);
18             String[] expected = new String[12];
19             expected[0] = "D A";
20             expected[1] = "S 0.0 ";
21             expected[2] = "P111 A";
22             expected[3] = "RM20 B";
23             expected[4] = "D A";
24             expected[5] = "D A";
25             expected[6] = "";
26             expected[7] = "D A";
27             expected[8] = "B A";
28             expected[9] = "RM20 L";
29             expected[10] = "D L";
30             expected[11] = "P111 L";
```

```
31
32     String[] inputs = new String[12];
33     inputs[0] = "D \"det skal vises\"";
34     inputs[1] = "S";
35     inputs[2] = "P111 \"vis det her i bunden\"";
36     inputs[3] = "RM20 8 \"<indtast her>\" \"<det overskrives>\"
37         + "\" \"&3\" ";
38     inputs[4] = "D \" nu vises det\"";
39     inputs[5] = "D \"Hej med dig\"";
40     inputs[6] = "P110";
41     inputs[7] = "D hej hej hej";
42     inputs[8] = "B 400";
43     inputs[9] = "RM20 8 dette her en alt for lang String så den går "
44         + "ikke indtast her nr ";
45     inputs[10] = "D \"det her er en alt for lang string til at vise "
46         + "i displayet\"";
47     inputs[11] = "P111 \"det her er også en alt for lang string at "
48         + "vise men nu ligger den nede i det nederste display.\"";
49
50     for (int i = 0; i < inputs.length; i++) {
51
52         System.out.println("input to server: " + inputs[i]);
53         System.out.println("expected output: " + expected[i]);
54         System.out.print("recieved: ");
55
56         DataOutputStream outToServer = new DataOutputStream(
57             clientSocket.getOutputStream());
58
59         BufferedReader inFromServer = new BufferedReader(
60             new InputStreamReader(clientSocket.getInputStream()));
61
62         sentence = inputs[i];
63         outToServer.writeBytes(sentence + "\r\n");
64         modifiedSentence = inFromServer.readLine();
65         System.out.println("FROM SERVER: " + modifiedSentence);
66         Thread.sleep(7000);
67     }
68
69     clientSocket.close();
70 } catch (Exception e) {
71     e.getMessage();
72 }
73
74 }
75
76 }
```



42 </body>

43 </html>

## commodity\_confirm\_delete\_boundary.jsp

```

1 <%@ page language="java" import="java.util.*,admin.data.* "
2 contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
   www.w3.org/TR/html4/loose.dtd">
4 <jsp:useBean id="error" class="java.lang.String" scope="request"/>
5 <jsp:useBean id="commodity" class="admin.data.CommodityDTO" scope="
   request"/>
6
7 boolean done = (Boolean) request.getAttribute("done");
8 %>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <title>Slet R vare</title>
13 <link rel="stylesheet" type="text/css" href="style.css">
14 </head>
15 <body>
16 <div class="delete_dialog">
17 <% if (!error.equals("")) { %>
18 <h1>R vare sletning</h1>
19 <% out.print(error); %>
20 <a href="commodity_admin">OK</a>
21 <% } else {
22     if (!done) {%>
23 <h1>Bekrft at du vil slette flgende r vare :</h1>
24 <%} else { %>
25 <h1>Denne r vare er blevet slettet</h1>
26 <%} %>
27 <table border="1"><tbody>
28 <tr>
29 <th>R vare ID</th>
30 <th>R varenavn</th>
31 <th>Leverand r</th>
32 </tr>
33 <tr>
34 <%
35     out.println("\t\t\t\t\t<td>" + commodity.getComId() + "</td>\n");
36     out.println("\t\t\t\t\t<td>" + commodity.getComName() + "</td>\n");
37     out.println("\t\t\t\t\t<td>" + commodity.getSupplier() + "</td>\n");
38 ;
39 %>
40 </tr>
41 </tbody></table>
42 <% if (!done) {%>
43 <div class="buttons">
44 <form method="post">
45 <input type="submit" value="Slet">
46 </form>
47 <a href="commodity_admin">Annuller</a>
48 </div>
49 <%} else { %>

```

```

49     <a href="commodity_admin">Tilbage</a>
50     <%>
51     } %>
52 </div>
53 </body>
54 </html>

```

**commodity\_edit\_boundary.jsp**[illegible]

```

45 <div class="error"><%out.print(majorError);%></div>
46 <% } else { %>
47 <h1>Indtast nye r vareoplysninger </h1>
48 <form method="post">
49   <% if (!idError.equals("")) { %>
50   <div class="error"><%out.print(idError);%></div>
51   <% } %>
52   <label for="comId"> R vare ID</label>
53   <input type="text" name="newId" id="comId"
54     value="<%out.print(newId);%>">
55
56   <% if (nameError != null && !nameError.equals("")) { %>
57   <div class="error"><%out.print(nameError);%></div>
58   <% } %>
59   <label for="comName"> R varenavn </label>
60   <input type="text" name="newName" id="comName"
61     value="<%out.print(commodity.getComName());%>">
62
63   <% if (supError != null && !supError.equals("")) { %>
64   <div class="error"><%out.print(supError);%></div>
65   <% } %>
66   <label for="supplier">Leverandør</label>
67   <input type="text" name="newSupplier" id="supplier"
68     value="<%out.print(commodity.getSupplier());%>">
69
70   <% if (create) { %>   <input type="submit" value="Opret">
71   <% } else { %> <input type="submit" value="Rediger"> <% } %>
72
73 </form>
74 <% } %>
75 <div class="buttons">
76   <a href="commodity_admin">Tilbage</a>
77   <a href="login?logout=true">Log ud</a>
78 </div>
79 </div>
80 </body>
81 </html>

```

### commodityBatch\_admin\_boundary.jsp

```

1 <%@ page language="java" import="java.util.*,admin.data.*"
2   contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4   www.w3.org/TR/html4/loose.dtd">
5 <%
6   List<CommodityBatchDTO> cBList = (List<CommodityBatchDTO>) request.
7     getAttribute("commodityBatchList");
8 %>
9 <html>
10 <head>
11   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12   <title> R varer Batch Administration</title>
13   <link rel="stylesheet" type="text/css" href="style.css">
14 </head>
15 <body>
16   <div class="adminDialog">

```







[illegible]

```

63         value="<%out.print(newComId);%>">
64
65     <% if (amountError != null && !amountError.equals("")) { %>
66     <div class="error"><%out.print(newComId);%></div>
67     <% } %>
68     <label for="amount">Mngde</label>
69     <input type="text" name="newAmount" id="amount"
70         value="<%out.print(newAmount);%>">
71
72     <% if (create) {%> <input type="submit" value="Opret">
73     <% } else { %><input type="submit" value="Rediger"> <% } %>
74
75 </form>
76 <%} %>
77 <div class="buttons">
78     <a href="commodityBatch_admin">Tilbage</a>
79     <a href="login?logout=true">Log ud</a>
80 </div>
81 </div>
82 </body>
83 </html>

```

### login\_boundary.jsp

```

1  <%@ page language="java" import="java.util.*,admin.data.*"
2     contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4     www.w3.org/TR/html4/loose.dtd">
5  <jsp:useBean id="error" class="java.lang.String" scope="request"/>
6  <jsp:useBean id="userId" class="java.lang.String" scope="request"/>
7  <jsp:useBean id="password" class="java.lang.String" scope="request"/>
8  <jsp:useBean id="redirect" class="java.lang.String" scope="request"/>
9  <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <title>Login</title>
13 <link rel="stylesheet" type="text/css" href="style.css">
14 </head>
15 <body>
16     <div class="dialog">
17         <h1>Intast bruger ID og kodeord </h1>
18         <% if (error != "") { %> <div class="error"> <% out.print(error); %> <
19             /div> <%} %>
20         <form method="post">
21             <label for="userid">Bruger ID</label>
22             <input type="text" name="userId" id="userid"
23                 value="<% out.print(userId); %>">
24             <label for="password">Kodeord</label>
25             <input type="password" name="password" id="password"
26                 value="<% out.print(password); %>">
27             <input type="hidden" name="redirect" id="redirect"
28                 value="<% out.print(redirect); %>">
29             <input type="submit" value="Log ind">
30         </form>
31     </div>

```

```

31  <%—Kun til test—%>
32  <%
33      out.print("Current Users <BR>");
34      IUsersReadOnly users = (IUsersReadOnly) request.getAttribute("users")
35          ;
36      List<UserDTO> operators = users.getUserList();
37      for(UserDTO op : operators){
38          out.print("ID: " + op.getUserId() + " PW: "
39              + op.getPassword() + " Brugertype: " + UserType.fromInt(op.
40                  getAccesLevel()) + "<BR>");
41      }
42  %>
43 </body>
44 </html>

```

### mainmenu\_boundary.jsp

```

1  <%@ page language="java" import="java.util.*,admin.data.*,admin.
    controller.*"
2      contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
    www.w3.org/TR/html4/loose.dtd">
4  <%
5      ArrayList<MenuOption> menuChoices = (ArrayList<MenuOption>) request.
        getAttribute("Menulist");
6  %>
7  <html>
8  <head>
9      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
10     <title>Mainmenu</title>
11     <link rel="stylesheet" type="text/css" href="style.css">
12 </head>
13 <body>
14 <div class="main_menu">
15     <h1>Hovedmenu</h1>
16 <%
17     for(MenuOption option: menuChoices){
18         out.print("<A href=\"\" + option.url + \"\">\" + option.name + "</A><BR>
19             \"");
20     }
21 %>
22 </div>
23 </body>
24 </html>

```

### password\_change\_boundary.jsp

```

1  <%@ page language="java" import="java.util.*,admin.data.*"
2      contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
    www.w3.org/TR/html4/loose.dtd">
4  <jsp:useBean id="newPword1" class="java.lang.String" scope="request"/>
5  <jsp:useBean id="newPword2" class="java.lang.String" scope="request"/>
6  <jsp:useBean id="password" class="java.lang.String" scope="request"/>

```

```

7 <jsp:useBean id="pwError" class="java.lang.String" scope="request"/>
8 <jsp:useBean id="npwError" class="java.lang.String" scope="request"/>
9 <%
10     Boolean success = (Boolean) request.getAttribute("success");
11 %>
12 <html>
13 <head>
14 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
15 <title>Test Program</title>
16 <link rel="stylesheet" type="text/css" href="style.css">
17 </head>
18 <body>
19
20     <div class="password_dialog">
21         <% if (success){%>
22             <h1>Dit password er dret </h1>
23         <%} else { %>
24             <h1>Her kan du dre password</h1>
25             <form method="post">
26                 <% if (pwError != null && !pwError.equals("")) { %>
27                     <div class="error"> <% out.print(pwError); %> </div> <%} %>
28                     <label for="pword">Nuvrende password</label>
29                     <input type="password" name="password" id="pword"
30                         value="<% out.print(password); %>">
31                 <% if (npwError != null && !npwError.equals("")) { %>
32                     <div class="error"> <% out.print(npwError); %> </div> <%} %>
33                     <label for="newPword1">Nyt password</label>
34                     <input type="password" name="newPword1" id="newPword1"
35                         value="<% out.print(newPword1); %>">
36                     <label for="newPword2">Gentag nyt password</label>
37                     <input type="password" name="newPword2" id="newPword2"
38                         value="<% out.print(newPword2); %>">
39                     <input type="submit" value="Skift Password">
40                 </form>
41             <%}%>
42             <div class="buttons">
43                 <A href="mainmenu">Hovedmenu</A>
44                 <A href="login?logout=true">Log ud</A>
45             </div>
46         </div>
47 </body>
48 </html>

```

### prescription\_admin\_boundary.jsp

```

1 <%@ page language="java" import="java.util.*,admin.data.*"
2     contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4     www.w3.org/TR/html4/loose.dtd">
5 <%
6     List<PrescriptionDTO> prescriptions = (List<PrescriptionDTO>) request.
7         getAttribute("prescriptionList");
8 %>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

```



[illegible]

```

70     <form method="post">
71         <input type="submit" value="Slet">
72     </form>
73     <%      }
74     } %>
75     <div class="buttons">
76         <A href="prescription_admin">Tilbage</A> <A href="login?logout=true
77             ">Log
78             ud</A>
79     </div>
80 </body>
81 </html>

```

### prescription\_edit\_boundary.jsp

```

1  <%@ page language="java" import="java.util.*,admin.data.*"
2     contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4     www.w3.org/TR/html4/loose.dtd">
5  <jsp:useBean id="majorError" class="java.lang.String" scope="request" />
6  <jsp:useBean id="idError" class="java.lang.String" scope="request" />
7  <jsp:useBean id="nameError" class="java.lang.String" scope="request" />
8  <jsp:useBean id="prescription" class="admin.data.PrescriptionDTO"
9     scope="request" />
10 <jsp:useBean id="newId" class="java.lang.String" scope="request" />
11 <%
12     boolean complete = (Boolean) request.getAttribute("complete");
13     boolean create = (Boolean) request.getAttribute("create");
14     List<PrescriptionCompDTO> components = (List<PrescriptionCompDTO>)
15         request
16         .getAttribute("components");
17 %>
18 <html>
19 <head>
20 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
21 <link rel="stylesheet" type="text/css" href="style.css">
22 <title>Edit Prescription</title>
23 </head>
24 <body>
25     <div class="dialog">
26         <% if (complete) {
27             if (!create) { %>
28             <h1>Dine dringer er gemt</h1>
29             <% } else { %>
30             <h1>Ny recept oprettet</h1>
31             <% } %>
32         <table border="1">
33             <tbody>
34                 <tr>
35                     <th>ID</th>
36                     <th>Navn</th>
37                 </tr>
38             </tbody>
39         </table>
40         out.println("\t\t\t\t\t<tr>\n");

```



```

38         out.println("\t\t\t\t\t<td>" + prescription.getId() + "</td>\n");
39     out.println("\t\t\t\t\t<td>" + prescription.getName()
40         + "</td>\n");
41     out.println("\t\t\t\t\t</tr>\n");
42     %>
43 </tbody>
44 </table>
45 <div class="compList">
46 Receptkomponenter:
47 <table>
48 <tr>
49 <th>R vare ID</th>
50 <th>Netto vgt</th>
51 <th>Tollerance</th>
52 </tr>
53 <%
54     int i = 0;
55     if (components != null && !components.isEmpty()) {
56         for (PrescriptionCompDTO comp : components) {%>
57 <tr>
58 <td>
59 <%out.print(comp.getCommodityId());%>
60 </td>
61 <td>
62 <%out.print(comp.getNomNetto());%>
63 </td>
64 <td>
65 <%out.print(comp.getTolerance());%>
66 </td>
67 </tr>
68 <%
69     i++;
70 } %>
71 </table>
72 </div>
73 <% } else if (!majorError.equals("")) {
74     if (create) { %>
75 <h1>Fejl under oprettelse af recept!</h1>
76 <% } else { %>
77 <h1>Fejl under redigering af recept!</h1>
78 <% } %>
79 <div class="error">
80 <% out.print(majorError); %>
81 </div>
82 <% } else { %>
83 <h1>Indtast nye receptoplysninger</h1>
84 <form id="main" method="post">
85
86 <% if (!idError.equals("")) { %>
87 <div class="error"> <% out.print(idError); %> </div>
88 <% } %>
89 <label for="userid">Recept ID</label>
90 <input type="text" name="newId"
91     id="userid" value="<%out.print(newId);%>">

```



```

92      <% if (nameError != null && !nameError.equals("")) { %>
93      <div class="error"> <% out.print(nameError); %> </div>
94      <% } %>
95      <label for="username">Receptnavn</label>
96      <input type="text"
97          name="newName" id="username"
98          value="<%out.print(prescription.getName());%>"
99      </form>
100     <div class="compList">
101         Receptkomponenter:
102         <table>
103             <tr>
104                 <th></th>
105                 <th>R vare ID</th>
106                 <th>Nettovgt</th>
107                 <th>Tollerance</th>
108             </tr>
109             <%
110                 int i = 0;
111                 if (components != null && !components.isEmpty()) {
112                     for (PrescriptionCompDTO comp : components){%>
113                     <tr>
114                         <td></td>
115                         <td>
116                             <% String comIdError = (String) request.getAttribute("
117                                 comIdError" + i);
118                             if (comIdError != null && !comIdError.equals("")){ %>
119                             <div class="error"> <% out.print(comIdError); %> </div>
120                             <% } %>
121                         </td>
122                         <td>
123                             <% String comNetError = (String) request.getAttribute("
124                                 comNetError" + i);
125                             if (comNetError != null && !comNetError.equals("")){ %>
126                             <div class="error"> <% out.print(comNetError); %> </div>
127                             <% } %>
128                         </td>
129                         <td>
130                             <% String comTolError = (String) request.getAttribute("
131                                 comTolError" + i);
132                             if (comTolError != null && !comTolError.equals("")){ %>
133                             <div class="error"> <% out.print(comTolError); %> </div>
134                             <% } %>
135                         </td>
136                     </tr>
137                     <tr>
138                         <td>
139                             <button name="button" value="delete<%out.print(i);%>"
140                                 type="submit" form="main">Slet</button>
141                         </td>
142                         <td>
143                             <input type="text" name="comId<%out.print(i);%>"
144                                 value="<%out.print(comp.getCommodityId());%>"
145                                 form="main">
146                         </td>
147                     </tr>
148                 }
149             %>
150         </table>
151     </div>
152 }
153 %>

```

```

144         <input type="text" name="netto"<%out.print(i);%>"
145             value="<%out.print(comp.getNomNetto());%>"
146             form="main">
147     </td>
148     <td>
149         <input type="text" name="tolerance"<%out.print(i);%>"
150             value="<%out.print(comp.getTolerance());%>"
151             form="main">
152     </td>
153 </tr>
154 <%
155     }
156 } %>
157 <tr>
158     <td>
159         <input type="hidden" name="compCount"
160             value="<%out.print(i);%>" form="main">
161         <button name="button" value="new"
162             type="submit" form="main">Ny</button>
163     </td>
164 </tr>
165 </table>
166 </div>
167 <%
168     if (create) {
169 %>
170 <input type="submit" value="Opret" form="main">
171 <%
172     } else {
173 %><input type="submit" value="Gem" form="main">
174 <%
175     }
176 %>
177 <input type="submit" name="button" value="Kontroller" form="main">
178 <%
179     }
180 %>
181 <div class="buttons">
182     <A href="prescription_admin">Tilbage</A> <A href="login?logout=true
183         ud</A>
184 </div>
185 </div>
186 </body>
187 </html>

```

### productBatch\_confirm\_delete\_boundary.jsp

```

1 <%@ page language="java" import="java.util.*,admin.data.*"
2     contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4     www.w3.org/TR/html4/loose.dtd">
5 <jsp:useBean id="error" class="java.lang.String" scope="request" />
6 <jsp:useBean id="product" class="admin.data.ProductBatchDTO"
7     scope="request" />
8 <%

```

```

boolean complete = (Boolean) request.getAttribute("complete");
List<ProductBatchCompDTO> components = (List<ProductBatchCompDTO>)
    request
        .getAttribute("components");
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<link rel="stylesheet" type="text/css" href="style.css">
<title>Slet Produktionsbatch</title>
</head>
<body>
<div class="dialog">
    <% if (complete) { %>
    <h1>Dette produktbatch er blevet slettet</h1>
    <% } else if (!error.equals("")) { %>
    <h1>Fejl under sletning af produktbatch!</h1>
    <div class="error"> <% out.print(error); %> </div>
    <% } else { %>
    <h1>Er du sikker p at du vil slette dette produktbatch ?</h1>
    <% }
    if (product != null){ %>
    <table border="1">
    <tbody>
    <tr>
    <th>Batch ID</th>
    <th>Recept ID</th>
    <th>Status</th>
    </tr>
    <tr>
    <% out.println("\t\t\t\t\t<td>" + product.getPbId() + "</td>\n")
    ;
    out.println("\t\t\t\t\t<td>" + product.getPrescriptionId() + "
    </td>\n");
    out.println("\t\t\t\t\t<td>" + product.getStatus().uiName() + "
    </td>\n"); %>
    </tr>
    </tbody>
    </table>
    <div class="compList">
    Foretagede afvejninger:
    <table>
    <tr>
    <th> R varebatch ID</th>
    <th>Operatr ID</th>
    <th>Nettovgt</th>
    <th>Tara vgt</th>
    </tr>
    <% if (components != null && !components.isEmpty()) {
    for (ProductBatchCompDTO comp : components){%>
    <tr>
    <td>
    <%out.print(comp.getCommoditybatch_id());%>
    </td>
    <td>

```

```

59         <%out.print(comp.getUser_id());%>
60     </td>
61     <td>
62         <%out.print(comp.getNetto());%>
63     </td>
64     <td>
65         <%out.print(comp.getTara());%>
66     </td>
67 </tr>
68 <%    }
69    } %>
70 </table>
71 </div>
72 <%    if (!complete && error.equals("")) { %>
73 <form method="post">
74     <input type="submit" value="Slet">
75 </form>
76 <%    }
77    } %>
78 <div class="buttons">
79     <A href="productBatch_admin">Tilbage</A>
80     <A href="login?logout=true">Log ud</A>
81 </div>
82 </div>
83 </body>
84 </html>

```

### productBatch\_edit\_boundary.jsp

```

1  <%@ page language="java" import="java.util.*,admin.data.*"
2     contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4     www.w3.org/TR/html4/loose.dtd">
5  <jsp:useBean id="majorError" class="java.lang.String" scope="request" />
6  <jsp:useBean id="idError" class="java.lang.String" scope="request" />
7  <jsp:useBean id="presIdError" class="java.lang.String" scope="request" />
8  <jsp:useBean id="statusError" class="java.lang.String" scope="request" />
9  <jsp:useBean id="userError" class="java.lang.String" scope="request" />
10 <jsp:useBean id="compError" class="java.lang.String" scope="request" />
11 <jsp:useBean id="product" class="admin.data.ProductBatchDTO"
12     scope="request" />
13 <jsp:useBean id="newId" class="java.lang.String" scope="request" />
14 <jsp:useBean id="newPresId" class="java.lang.String" scope="request" />
15 <jsp:useBean id="newStatus" class="java.lang.String" scope="request" />
16 <jsp:useBean id="newUserId" class="java.lang.String" scope="request" />
17 <%
18     boolean complete = (Boolean) request.getAttribute("complete");
19     boolean create = (Boolean) request.getAttribute("create");
20     List<ProductBatchCompDTO> components = (List<ProductBatchCompDTO>)
21         request
22             .getAttribute("components");
23 %>
24 <html>
25 <head>
26 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

```

[illegible]

```

78         }
79     }
80     %>
81     </table>
82 </div>
83 <%      } %>
84 <%
85     } else if (!majorError.equals("")) {
86         if (create) {
87     %>
88     <h1>Fejl under oprettelse af produktbatch!</h1>
89     <%
90         } else {
91     %>
92     <h1>Fejl under redigering af produktbatch!</h1>
93     <%
94         }
95     %>
96     <div class="error">
97         <%
98             out.print(majorError);
99         %>
100    </div>
101    <%
102        } else {
103    %>
104    <h1>Indtast nye oplysninger om produktbatch</h1>
105    <form method="post">
106        <% if (!idError.equals("")) { %>
107        <div class="error"> <% out.print(idError); %> </div>
108        <% } %>
109        <label for="prodId">Produktbatch ID</label>
110        <input type="text" name="newId"
111            id="prodId" value="<%out.print(newId);%>">
112
113        <% if (!presIdError.equals("")) { %>
114        <div class="error"> <% out.print(presIdError); %> </div>
115        <% } %>
116        <label for="presId">Recept ID</label> <input type="text"
117            name="newPresId" id="presId"
118            value="<%out.print(newPresId);%>">
119
120        <% if (!userError.equals("")) { %>
121        <div class="error"> <% out.print(userError); %> </div>
122        <% } %>
123        <label for="userId">Bruger ID</label> <input type="text"
124            name="newUserId" id="userId"
125            value="<%out.print(newUserId);%>">
126
127        <label for="date">Oprettet</label> <input type="text"
128            name="date" id="date" disabled
129            value="<%out.print(product.getCreationDate());%>">
130
131        <% if (!statusError.equals("")) { %>
132        <div class="error"> <% out.print(statusError); %> </div>

```

```

133     <% } %>
134     <label for="usertype">Status</label>
135     <select name="newStatus">
136         <option value="NEW" <% if (newStatus.equals("NEW")) out.print("
            selected");%>>
137         <%out.print(StatusType.NEW.uiName());%></option>
138         <option value="IN_PRODUCTION" <% if (newStatus.equals("
            IN_PRODUCTION")) out.print("selected");%>>
139         <%out.print(StatusType.IN_PRODUCTION.uiName());%></option>
140         <option value="PAUSED" <% if (newStatus.equals("PAUSED")) out.
            print("selected");%>>
141         <%out.print(StatusType.PAUSED.uiName());%></option>
142         <option value="FINISHED" <% if (newStatus.equals("FINISHED")) out
            .print("selected");%>>
143         <%out.print(StatusType.FINISHED.uiName());%></option>
144     </select>
145     <div class="compList">
146         Udfrte vejninger:
147         <% if (compError != null && !compError.equals("")) { %>
148         <div class="error"> <% out.print(compError); %> </div>
149         <% } %>
150         <table>
151             <tr>
152                 <th> R varebatch ID</th>
153                 <th>Operatr ID</th>
154                 <th>Netto vgt</th>
155                 <th>Tara vgt</th>
156             </tr>
157             <%
158                 if (components != null && !components.isEmpty()) {
159                     for (ProductBatchCompDTO comp : components) {
160                         %>
161                         <tr>
162                             <td> <% out.print(comp.getCommoditybatch_id()); %></td>
163                             <td> <% out.print(comp.getUser_id()); %></td>
164                             <td> <% out.print(comp.getNetto()); %></td>
165                             <td> <% out.print(comp.getTara()); %></td>
166                         </tr>
167                         <%
168                             } %>
169                     </table>
170                 </div>
171                 <% if (create) { %>
172                 <input type="submit" value="Opret" >
173                 <% } else { %>
174                 <input type="submit" value="Gem" >
175                 <% }
176                 %>
177             </form>
178             <div class="buttons">
179                 <A href="productBatch_admin">Tilbage</A>
180                 <A href="login?logout=true">Log ud</A>
181             </div>
182         </div>
183     </body>

```

184 &lt;/html&gt;

**productBatch\_print\_boundary.jsp**

```

1 <%@ page language="java" import="java.util.*,admin.data.*"
2   contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
   www.w3.org/TR/html4/loose.dtd">
4 <jsp:useBean id="error" class="java.lang.String" scope="request" />
5 <jsp:useBean id="date" class="java.lang.String" scope="request" />
6 <jsp:useBean id="sumTara" class="java.lang.String" scope="request" />
7 <jsp:useBean id="sumNetto" class="java.lang.String" scope="request" />
8 <jsp:useBean id="product" class="admin.data.ProductBatchDTO"
9   scope="request" />
10 <%
11   List<ProductListItem> components = (List<ProductListItem>) request
12     .getAttribute("components");
13 %>
14
15 <html>
16 <head>
17 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
18 <link rel="stylesheet" type="text/css" href="style.css">
19 <title>Rediger Produktbatch</title>
20 </head>
21 <body>
22 <div class="print">
23   <% if (!error.equals("")) { out.print(error); } %>
24   <table border="0"><tbody>
25     <tr>
26       <td>Udskrevet</td>
27       <td><% out.print(date);%></td>
28     </tr>
29     <tr>
30       <td>Produkt Batch ID</td>
31       <td><% out.print(product.getPbId());%></td>
32     </tr>
33     <tr>
34       <td>Recept ID</td>
35       <td><% out.print(product.getPrescriptionId());%></td>
36     </tr>
37   </tbody></table>
38   <table border="0"><tbody>
39     <% for (ProductListItem item : components) {%>
40       <tr><td><br><br></td></tr>
41       <tr><td>
42         <table border="0"><tbody>
43           <tr>
44             <td>R vare ID</td>
45             <td><% out.print(item.commodityId);%></td>
46           </tr>
47           <tr>
48             <td>R vare Navn</td>
49             <td><% out.print(item.commodityName);%></td>
50           </tr>
51         </tbody></table>

```



```

52     <hr>
53     <table border="0"><tbody>
54         <tr>
55             <td>Mngde</td>
56             <td>Tolerance</td>
57             <td>Tara</td>
58             <td>Netto</td>
59             <td>R vare Batch</td>
60             <td>Operat r</td>
61         </tr>
62         <tr>
63             <td><% out.print(item.ammount);%></td>
64             <td><% out.print(item.tolerance);%></td>
65             <td><% if (item.tara != 0 ) out.print(item.tara);%></td>
66             <td><% if (item.netto != 0 ) out.print(item.netto);%></td>
67             <td><% if (item.commodityBatch != 0 ) out.print(item.
68                 commodityBatch);%></td>
69             <td><% out.print(item.operator);%></td>
70         </tr>
71     </tbody></table>
72     <% } %>
73     <tr><td>
74         <BR><BR><BR>
75         <table border="0"><tbody>
76             <tr>
77                 <td>Sum Tara</td>
78                 <td><% out.print(sumTara);%></td>
79             </tr>
80             <tr><td><BR></td></tr>
81             <tr>
82                 <td>Sum Netto</td>
83                 <td><% out.print(sumNetto);%></td>
84             </tr>
85             <tr><td><BR><BR></td></tr>
86             <tr>
87                 <td>Status</td>
88                 <td><% out.print(product.getStatus().uiName());%></td>
89             </tr>
90             <tr>
91                 <td>Oprettelsesdato</td>
92                 <td><% out.print(product.getCreationDate());%></td>
93             </tr>
94         </tbody></table>
95     </td></tr>
96 </tbody></table>
97 <A href="productBatch_admin">Tilbage</A>
98 </div>
99 </body>
100 </html>

```

style

```

h1 {
    text-align: center;
    -webkit-margin-before: 0;

```

```
    color: white;
}

.dialog,
.adminDialog,
.main_menu,
.delete_dialog,
.password_dialog,
.product_admin{
    padding: 20px;
    margin: auto;
    margin-top: 5em;
    background: #77B;
    border: 4px outset #88C;
}

.main_menu{
    width: 20em;
}

.product_admin{
    width: 40em;
}

.dialog,
.password_dialog {
    width: 28em;
}

.adminDialog,
.delete_dialog{
    width: 35em;
}

.dialog label,
.password_dialog label{
    box-sizing: border-box;
    display: inline-block;
    margin: 4px 2px;
    padding: 2px 5px;
    width: 40%;
    background: #000;
    color: #fff;
}

.dialog input,
.dialog select,
.dialog button,
.password_dialog input{
    width: 58%;
    padding: 2px 5px;
    box-sizing: border-box;
}

.dialog input[type="submit"],
```

```
.dialog button {
    width: 70px;
}

.password_dialog input[type="submit"]
{
    width: 10em;
}

.error {
    margin: 4px 2px;
    text-align: center;
    color: white;
    background: red;
    border: 1px dashed white;
    box-sizing: border-box;
    padding: 2px 5px;
}

.buttons{
    width: 14em;
    margin: auto;
    margin-top: 2em;
}

.delete_dialog .buttons{
    width: 12em;
}

a,
button {
    text-align: center;
    color: black;
    padding: 2px;
    font-style: normal;
    text-decoration: none;
    width: 5em;
}

a,
input[type="submit"],
button {
    display: block;
    margin: auto;
    background: Gainsboro;
    border: 2px outset gray;
    font-family: "Times New Roman", Times, serif;
    font-size: 14px;
    margin-top: 1em;
}

tr a,
tr button{
    margin: auto;
}
```

```
.adminDialog a{
    display: block;
    width: 10em;
}

.product_admin a{
    display: block;
    width: 11em;
}

.adminDialog .buttons a,
.product_admin .buttons a{
    display: inline-block;
    margin: 0.5em;
    width: 6em;
}

.main_menu a{
    width: 14em;
    margin: auto;
}

.dialog .buttons a,
.password_dialog .buttons a{
    display: inline-block;
    margin: 0.5em;
    width: 6em;
}

.delete_dialog .buttons a,
.delete_dialog .buttons input{
    display: inline-block;
    width: 5em;
    margin: 0.5em;
    margin-top: 1em;
    padding: 2px;
}

.adminDialog table a,
.product_admin table a{
    width: 4em;
}

table{
    margin: auto;
    border-collapse: collapse;
}

td{
    text-align: center;
    padding: 0em 0.5em;
}

td a{
```

```

    margin: 0em -0.5em;
}

tr{
    border: 1px solid black;
    border-color: black transparent;
    vertical-align: middle;
}

.long_button{
    width: 10em;
}

.buttons form{
    display: inline-block;
}

.compList input {
    margin: auto;
}

.status{
    padding: 0em 1em;
}

input[disabled] {
    color: #000;
    background: transparent;
    border: 0px solid transparent;
}

.print tr{
    border: 0px solid transparent;
}

.print td{
    padding: 0em 1em;
    text-align: left;
}

.print table{
    margin: 0px;
}

.print a{
    margin: 0px;
    margin-top: 2em;
}

}

test_boundary.jsp

```

```

1 <%@ page language="java" import="java.util.*,admin.data.*"
2   contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
   www.w3.org/TR/html4/loose.dtd">

```

```

4 <jsp:useBean id="error" class="java.lang.String" scope="request"/>
5 <jsp:useBean id="bruto" class="java.lang.String" scope="request"/>
6 <jsp:useBean id="tara" class="java.lang.String" scope="request"/>
7 <%
8     double netto = (Double) request.getAttribute("netto");
9 %>
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
13 <title>Test Program</title>
14 <link rel="stylesheet" type="text/css" href="style.css">
15 </head>
16 <body>
17     <div class="dialog">
18         <h1>Intast bruto og tara vgt </h1>
19         <% if (error != "") { %> <div class="error"> <% out.print(error); %> <
20             /div> <%} %>
21         <form method="post">
22             <label for="bruto">Bruto vgt </label>
23             <input type="text" name="bruto" id="bruto"
24                 value="<% out.print(bruto); %>">
25             <label for="tara">Tara vgt </label>
26             <input type="text" name="tara" id="tara"
27                 value="<% out.print(tara); %>">
28             <label for="netto">Netto vgt (resultat)</label>
29             <input type="text" name="netto" id="netto" readonly
30                 value="<% out.print(netto); %>">
31             <input type="submit" value="Beregn">
32         </form>
33         <div class="buttons">
34             <a href="mainmenu">Hovedmenu</a>
35             <a href="login?logout=true">Log ud</a>
36         </div>
37     </div>
38 </body>
39 </html>

```

### user\_admin\_boundary.jsp

```

1 <%@ page language="java" import="java.util.*,admin.data.*"
2     contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4     www.w3.org/TR/html4/loose.dtd">
5 <%
6     List<UserInfo> users = (List<UserInfo>) request.getAttribute("userlist")
7     );
8 %>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
12 <title>Bruger Administration</title>
13 <link rel="stylesheet" type="text/css" href="style.css">
14 </head>
15 <body>
16     <div class="adminDialog">
17         <h1>Bruger Administration </h1>

```

```

16 <table border="1"><tbody>
17 <tr>
18 <th>ID</th>
19 <th>Initialer</th>
20 <th>Navn</th>
21 <th>CPR-nummer</th>
22 <th>Brugertype</th>
23 <th>Rediger</th>
24 <th>Slet</th>
25 </tr>
26 <%
27 for (UserInfo user : users){
28 out.println("\t\t\t\t\t<tr>\n");
29 out.println("\t\t\t\t\t<td>" + user.id + "</td>\n");
30 out.println("\t\t\t\t\t<td>" + user.ini + "</td>\n");
31 out.println("\t\t\t\t\t<td>" + user.name + "</td>\n");
32 out.println("\t\t\t\t\t<td>" + user.cpr + "</td>\n");
33 out.println("\t\t\t\t\t<td>" + user.access.uiName() + "</td>\n");
34 out.println("\t\t\t\t\t<td><A href=\"" + user.editURL + "\">
    Rediger</A></td>\n");
35 out.println("\t\t\t\t\t<td><A href=\"" + user.deleURL + "\">
    Slet</A></td>\n");
36 out.println("\t\t\t\t\t</tr>\n");
37 }
38 %>
39 </tbody></table>
40 <A href="user_edit?id=new">Opret ny bruger</A>
41 <div class="buttons">
42 <A href="mainmenu">Hovedmenu</A>
43 <A href="login?logout=true">Log ud</A>
44 </div>
45 </div>
46 </body>
47 </html>

```

**user\_confirm\_delete\_boundary.jsp**

```

1 <%@ page language="java" import="java.util.*,admin.data.*"
2     contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3 <!--DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4     www.w3.org/TR/html4/loose.dtd">
5 <jsp:useBean id="error" class="java.lang.String" scope="request"/>
6 <jsp:useBean id="userInfo" class="admin.data.UserInfo" scope="request"/>
7 <%
8     boolean done = (Boolean) request.getAttribute("done");
9 %>
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
13 <title>Delete User</title>
14 <link rel="stylesheet" type="text/css" href="style.css">
15 </head>
16 <body>
17     <div class="delete_dialog">
18         <% if (!error.equals("")) { %>

```

```

18      <h1>Bruger sletning</h1>
19      <% out.print(error); %>
20      <a href="user_admin">OK</a>
21      <% } else {
22          if (!done) {%>
23      <h1>Bekrft at du vil slette flgende bruger:</h1>
24      <%} else { %>
25      <h1>Denne bruger er blevet slettet</h1>
26      <%} %>
27      <table border="1"><tbody>
28          <tr>
29              <th>ID</th>
30              <th>Initialer</th>
31              <th>Navn</th>
32              <th>CPR-nummer</th>
33              <th>Administrator</th>
34          </tr>
35      <%
36          out.println("\t\t\t\t\t<tr>\n");
37          out.println("\t\t\t\t\t<td>" + userInfo.id + "</td>\n");
38          out.println("\t\t\t\t\t<td>" + userInfo.ini + "</td>\n");
39          out.println("\t\t\t\t\t<td>" + userInfo.name + "</td>\n");
40          out.println("\t\t\t\t\t<td>" + userInfo.cpr + "</td>\n");
41          out.println("\t\t\t\t\t<td>" + userInfo.access.userName() + "</td>\n");
42          ;
43      </tbody></table>
44      <% if (!done) {%>
45      <div class="buttons">
46          <form method="post">
47              <input type="hidden" name="confirmed" id="confirmed"
48                  value="true">
49              <input type="submit" value="Slet">
50          </form>
51          <a href="user_admin">Annullr</a>
52      </div>
53      <%} else { %>
54      <a href="user_admin">Tilbage</a>
55      <%}
56      } %>
57  </div>
58 </body>
59 </html>

```

### user\_edit\_boundary.jsp

```

1  <%@ page language="java" import="java.util.*,admin.data.*"
2      contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
4      www.w3.org/TR/html4/loose.dtd">
5  <jsp:useBean id="majorError" class="java.lang.String" scope="request"/>
6  <jsp:useBean id="idError" class="java.lang.String" scope="request"/>
7  <jsp:useBean id="accessError" class="java.lang.String" scope="request"/>
8  <jsp:useBean id="error" class="admin.data.UserInfo" scope="request"/>
9  <jsp:useBean id="info" class="admin.data.UserInfo" scope="request"/>
10 <jsp:useBean id="old" class="admin.data.UserInfo" scope="request"/>

```



```

10 <jsp:useBean id="newId" class="java.lang.String" scope="request"/>
11 <jsp:useBean id="oldId" class="java.lang.String" scope="request"/>
12 <jsp:useBean id="newPw" class="java.lang.String" scope="request"/>
13 <% boolean complete = (Boolean) request.getAttribute("complete"); %>
14 <% boolean create = (Boolean) request.getAttribute("create"); %>
15 <html>
16 <head>
17 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
18 <link rel="stylesheet" type="text/css" href="style.css">
19 <title>Edit user</title>
20 </head>
21 <body>
22
23
24 <div class="dialog">
25   <% if (complete){ %>
26     <% if (!create){ %>
27       <h1>Redigering foretaget</h1>
28       <table border="1"><tbody>
29         <tr>
30           <th>ID</th>
31           <th>Initialer</th>
32           <th>Navn</th>
33           <th>CPR-nummer</th>
34           <th>Brugertype</th>
35         </tr>
36       <%
37         out.println("\t\t\t\t\t<tr>\n");
38         out.println("\t\t\t\t\t<td>" + old.id + "</td>\n");
39         out.println("\t\t\t\t\t<td>" + old.ini + "</td>\n");
40         out.println("\t\t\t\t\t<td>" + old.name + "</td>\n");
41         out.println("\t\t\t\t\t<td>" + old.cpr + "</td>\n");
42         out.println("\t\t\t\t\t<td>" + old.access.userName() + "</td>\n");
43       %>
44       </tbody></table><br>
45       Er blevet rettet til:<br><br>
46     <% } else { %>
47       <h1>Brugeroprettelse foretaget</h1>
48     <% } %>
49     <table border="1"><tbody>
50       <tr>
51         <th>ID</th>
52         <th>Initialer</th>
53         <th>Navn</th>
54         <th>CPR-nummer</th>
55         <th>Brugertype</th>
56       </tr>
57     <%
58       out.println("\t\t\t\t\t<tr>\n");
59       out.println("\t\t\t\t\t<td>" + info.id + "</td>\n");
60       out.println("\t\t\t\t\t<td>" + info.ini + "</td>\n");
61       out.println("\t\t\t\t\t<td>" + info.name + "</td>\n");
62       out.println("\t\t\t\t\t<td>" + info.cpr + "</td>\n");
63       out.println("\t\t\t\t\t<td>" + info.access.userName() + "</td>\n");
64     %>

```

```

65     </tbody></table>
66     <% if (create){ %>
67         Brugerens kodeord er:<br>
68         <h1><%out.println(newPw); %></h1>
69
70     }
71     } else if (!majorError.equals("")){
72 %>
73 <% if (create) {%> <h1>Fejl under oprettelse af bruger!</h1>
74 <% } else { %><h1>Fejl under redigering af bruger!</h1> <% } %>
75 <div class="error"><%out.print(majorError);%></div>
76 <% } else { %>
77 <h1>Indtast nye brugeroplysninger</h1>
78 <form method="post">
79     <% if (!idError.equals("")){ %>
80     <div class="error"><%out.print(idError);%></div>
81     <% } %>
82     <label for="userid">Bruger ID</label>
83     <input type="text" name="newId" id="userid"
84         value="<%out.print(newId);%>">
85
86     <% if (error.ini != null){ %>
87     <div class="error"><%out.print(error.ini);%></div>
88     <% } %>
89     <label for="userini">Bruger initialer</label>
90     <input type="text" name="newIni" id="userini"
91         value="<%out.print(info.ini);%>">
92
93     <% if (error.name != null){ %>
94     <div class="error"><%out.print(error.name);%></div>
95     <% } %>
96     <label for="username">Brugernavn</label>
97     <input type="text" name="newName" id="username"
98         value="<%out.print(info.name);%>">
99
100     <% if (error.cpr != null){ %>
101     <div class="error"><%out.print(error.cpr);%></div>
102     <% } %>
103     <label for="usercpr">CPR</label>
104     <input type="text" name="newCPR" id="usercpr"
105         value="<%out.print(info.cpr);%>">
106
107     <% if (!accessError.equals("")){ %>
108     <div class="error"><%out.print(accessError);%></div>
109     <% }
110     String access = "OPERATOR";
111     if (info.access != null)
112         access = info.access.name();%>
113     <label for="usertype">Bruger Type</label>
114     <select name="newAccess">
115     <option value="OPERATOR" <% if (access.equals("OPERATOR")) out.print(
116         "selected");%>>
117     <%out.print(UserType.OPERATOR.uiName());%></option>
118     <option value="FOREMAN" <% if (access.equals("FOREMAN")) out.print(
119         "selected");%>>

```

```
118     <%out.print(UserType.FOREMAN.uiName());%></option>
119     <option value="PHARMACIST" <% if (access.equals("PHARMACIST")) out.
        print("selected");%>>
120     <%out.print(UserType.PHARMACIST.uiName());%></option>
121     <option value="ADMIN" <% if (access.equals("ADMIN")) out.print("
        selected");%>>
122     <%out.print(UserType.ADMIN.uiName());%></option>
123     <option value="INACTIVE" <% if (access.equals("INACTIVE")) out.print(
        "selected");%>>
124     <%out.print(UserType.INACTIVE.uiName());%></option>
125     </select>
126     <% if (create) {%> <input type="submit" value="Opret">
127     <% } else { %><input type="submit" value="Rediger"> <% } %>
128
129 </form>
130 <%} %>
131 <div class="buttons">
132     <A href="user_admin">Tilbage</A>
133     <A href="login?logout=true">Log ud</A>
134 </div>
135 </div>
136 </body>
137 </html>
```

## A.5 WEBINF

Al kode i WEBINF mappen

### web.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="
  http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun
  .com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
3   <display-name>CDIO</display-name>
4   <welcome-file-list>
5     <welcome-file>login</welcome-file>
6   </welcome-file-list>
7   <servlet>
8     <description>Login Page</description>
9     <display-name>LoginController</display-name>
10    <servlet-name>LoginController</servlet-name>
11    <servlet-class>admin.controller.LoginController</servlet-class>
12  </servlet>
13  <servlet-mapping>
14    <servlet-name>LoginController</servlet-name>
15    <url-pattern>/login</url-pattern>
16  </servlet-mapping>
17  <servlet>
18    <description>Test Program</description>
19    <display-name>Test</display-name>
20    <servlet-name>TestController</servlet-name>
21    <servlet-class>admin.controller.TestController</servlet-class>
22  </servlet>
23  <servlet-mapping>
24    <servlet-name>TestController</servlet-name>
25    <url-pattern>/test</url-pattern>
26  </servlet-mapping>
27  <servlet>
28    <description>Password Change</description>
29    <display-name>Change Password</display-name>
30    <servlet-name>PasswordChangeController</servlet-name>
31    <servlet-class>admin.controller.PasswordChangeController</servlet-
    class>
32  </servlet>
33  <servlet-mapping>
34    <servlet-name>PasswordChangeController</servlet-name>
35    <url-pattern>/password_change</url-pattern>
36  </servlet-mapping>
37  <servlet>
38    <description>the main menu</description>
39    <display-name>Main Menu</display-name>
40    <servlet-name>MainMenuController</servlet-name>
41    <servlet-class>admin.controller.MainMenuController</servlet-class>
42  </servlet>
43  <servlet-mapping>
44    <servlet-name>MainMenuController</servlet-name>
45    <url-pattern>/mainmenu</url-pattern>
```

```
46     </servlet-mapping>
47
48
49     <!-- User Administration -->
50     <servlet>
51         <description>User Administration</description>
52         <display-name>User Admin</display-name>
53         <servlet-name>UserAdminController</servlet-name>
54         <servlet-class>admin.controller.UserAdminController</servlet-class>
55     </servlet>
56     <servlet-mapping>
57         <servlet-name>UserAdminController</servlet-name>
58         <url-pattern>/user_admin</url-pattern>
59     </servlet-mapping>
60     <servlet>
61         <description>Confirm deletion of user</description>
62         <display-name>Delete User</display-name>
63         <servlet-name>UserConfirmDeleteController</servlet-name>
64         <servlet-class>admin.controller.UserConfirmDeleteController</servlet-
            class>
65     </servlet>
66     <servlet-mapping>
67         <servlet-name>UserConfirmDeleteController</servlet-name>
68         <url-pattern>/user_confirm_delete</url-pattern>
69     </servlet-mapping>
70     <servlet>
71         <description>Editing or creation of a user</description>
72         <display-name>Edit or new user</display-name>
73         <servlet-name>UserEditController</servlet-name>
74         <servlet-class>admin.controller.UserEditController</servlet-class>
75     </servlet>
76     <servlet-mapping>
77         <servlet-name>UserEditController</servlet-name>
78         <url-pattern>/user_edit</url-pattern>
79     </servlet-mapping>
80
81
82     <!-- Product Batch Administration -->
83     <servlet>
84         <description>Product Batch Administration</description>
85         <display-name>Product Batch Admin</display-name>
86         <servlet-name>ProductBatchAdminController</servlet-name>
87         <servlet-class>admin.controller.ProductBatchAdminController</servlet-
            class>
88     </servlet>
89     <servlet-mapping>
90         <servlet-name>ProductBatchAdminController</servlet-name>
91         <url-pattern>/productBatch_admin</url-pattern>
92     </servlet-mapping>
93     <servlet>
94         <description>Editing a product batch</description>
95         <display-name>Edit Product Batch</display-name>
96         <servlet-name>ProductBatchEditController</servlet-name>
97         <servlet-class>admin.controller.ProductBatchEditController</servlet-
            class>
```

```
98     </servlet>
99     <servlet-mapping>
100         <servlet-name>ProductBatchEditController</servlet-name>
101         <url-pattern>/productBatch_edit</url-pattern>
102     </servlet-mapping>
103     <servlet>
104         <description>Confirm deletion of product batch</description>
105         <display-name>Delete Product Batch</display-name>
106         <servlet-name>ProductBatchConfirmDeleteController</servlet-name>
107         <servlet-class>admin.controller.ProductBatchConfirmDeleteController</
            servlet-class>
108     </servlet>
109     <servlet-mapping>
110         <servlet-name>ProductBatchConfirmDeleteController</servlet-name>
111         <url-pattern>/productBatch_confirm_delete</url-pattern>
112     </servlet-mapping>
113     <servlet>
114         <description>Print of product batch</description>
115         <display-name>Print Product Batch</display-name>
116         <servlet-name>ProductBatchPrintController</servlet-name>
117         <servlet-class>admin.controller.ProductBatchPrintController</servlet-
            class>
118     </servlet>
119     <servlet-mapping>
120         <servlet-name>ProductBatchPrintController</servlet-name>
121         <url-pattern>/productBatch_print</url-pattern>
122     </servlet-mapping>
123
124
125     <!-- Prescription Administration -->
126     <servlet>
127         <description>Prescription Administration</description>
128         <display-name>Prescription Admin</display-name>
129         <servlet-name>PrescriptionAdminController</servlet-name>
130         <servlet-class>admin.controller.PrescriptionAdminController</servlet-
            class>
131     </servlet>
132     <servlet-mapping>
133         <servlet-name>PrescriptionAdminController</servlet-name>
134         <url-pattern>/prescription_admin</url-pattern>
135     </servlet-mapping>
136     <servlet>
137         <description>Editing a prescription</description>
138         <display-name>Edit Prescription</display-name>
139         <servlet-name>PrescriptionEditController</servlet-name>
140         <servlet-class>admin.controller.PrescriptionEditController</servlet-
            class>
141     </servlet>
142     <servlet-mapping>
143         <servlet-name>PrescriptionEditController</servlet-name>
144         <url-pattern>/prescription_edit</url-pattern>
145     </servlet-mapping>
146     <servlet>
147         <description>Confirm deletion of prescription</description>
148         <display-name>Delete Prescription</display-name>
```

```
149     <servlet-name>PrescriptionConfirmDeleteController</servlet-name>
150     <servlet-class>admin.controller.PrescriptionConfirmDeleteController</
        servlet-class>
151 </servlet>
152 <servlet-mapping>
153     <servlet-name>PrescriptionConfirmDeleteController</servlet-name>
154     <url-pattern>/prescription_confirm_delete</url-pattern>
155 </servlet-mapping>
156
157
158 <!-- Commodity Batch Administration -->
159 <servlet>
160     <description>Commodity Batch Administration</description>
161     <display-name>Commodity Batch Admin</display-name>
162     <servlet-name>BatchCommodityAdminController</servlet-name>
163     <servlet-class>admin.controller.BatchCommodityAdminController</
        servlet-class>
164 </servlet>
165 <servlet-mapping>
166     <servlet-name>BatchCommodityAdminController</servlet-name>
167     <url-pattern>/commodityBatch_admin</url-pattern>
168 </servlet-mapping>
169 <servlet>
170     <description>Editing a commodity batch</description>
171     <display-name>Edit Commodity Batch</display-name>
172     <servlet-name>BatchCommodityEditController</servlet-name>
173     <servlet-class>admin.controller.BatchCommodityEditController</servlet
        -class>
174 </servlet>
175 <servlet-mapping>
176     <servlet-name>BatchCommodityEditController</servlet-name>
177     <url-pattern>/commodityBatch_edit</url-pattern>
178 </servlet-mapping>
179 <servlet>
180     <description>Corfirm deletion of commodity batch</description>
181     <display-name>Delete Commodity Batch</display-name>
182     <servlet-name>BatchCommodityConfirmDeleteController</servlet-name>
183     <servlet-class>admin.controller.BatchCommodityConfirmDeleteController
        </servlet-class>
184 </servlet>
185 <servlet-mapping>
186     <servlet-name>BatchCommodityConfirmDeleteController</servlet-name>
187     <url-pattern>/commodityBatch_confirm_delete</url-pattern>
188 </servlet-mapping>
189
190
191 <!-- Commodity Administration -->
192 <servlet>
193     <description>Commodity Administration</description>
194     <display-name>Commodity Admin</display-name>
195     <servlet-name>CommodityAdminController</servlet-name>
196     <servlet-class>admin.controller.CommodityAdminController</servlet-
        class>
197 </servlet>
198 <servlet-mapping>
```

```
199     <servlet-name>CommodityAdminController</servlet-name>
200     <url-pattern>/commodity_admin</url-pattern>
201 </servlet-mapping>
202 <servlet>
203     <description>Editing a commodity </description>
204     <display-name>Edit Commodity </display-name>
205     <servlet-name>CommodityEditController</servlet-name>
206     <servlet-class>admin.controller.CommodityEditController</servlet-
        class>
207 </servlet>
208 <servlet-mapping>
209     <servlet-name>CommodityEditController</servlet-name>
210     <url-pattern>/commodity_edit</url-pattern>
211 </servlet-mapping>
212 <servlet>
213     <description>Corfirm deletion of commodity</description>
214     <display-name>Delete Commodity</display-name>
215     <servlet-name>CommodityConfirmDeleteController</servlet-name>
216     <servlet-class>admin.controller.CommodityConfirmDeleteController</
        servlet-class>
217 </servlet>
218 <servlet-mapping>
219     <servlet-name>CommodityConfirmDeleteController</servlet-name>
220     <url-pattern>/commodity_confirm_delete</url-pattern>
221 </servlet-mapping>
222 </web-app>
```

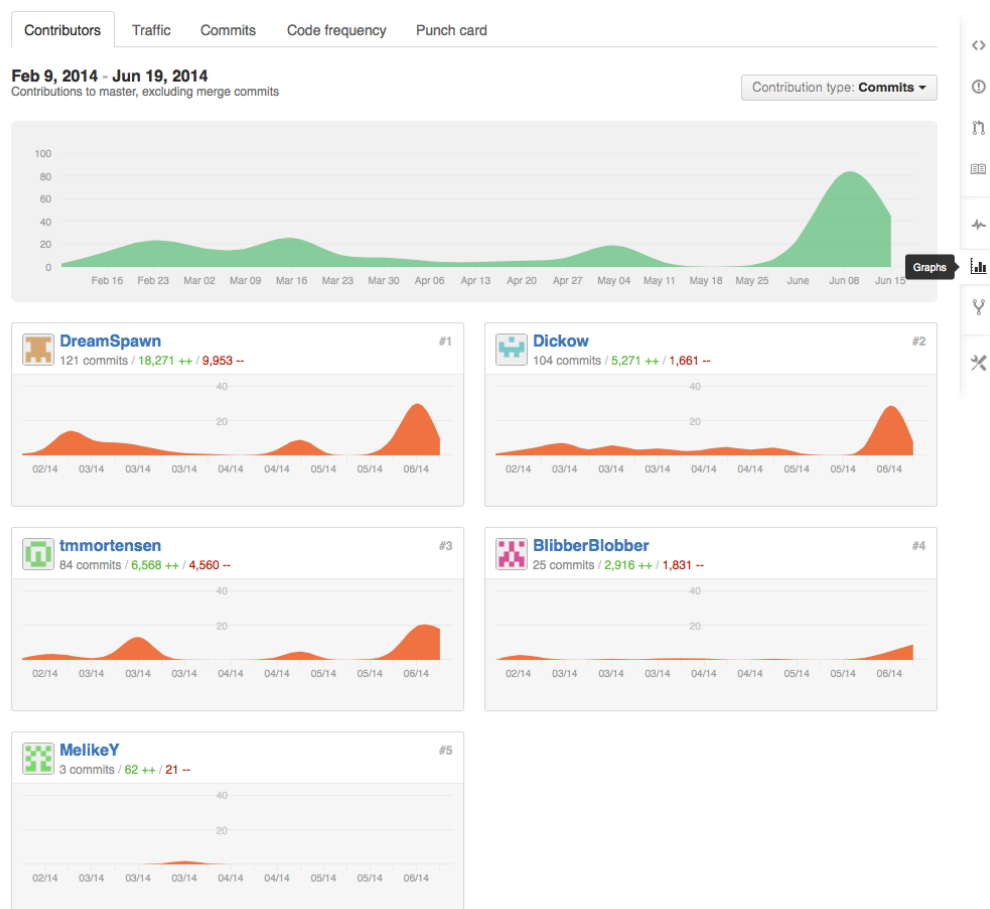


## B Arbejdsgang

Vi har i gruppen haft et ganske udemærket samarbejde. Gruppen har bestået af folk på et meget forskelligt niveau. En af os har utrolig meget erfaring med kodning, et par stykker har rodet meget lidt med noget ved siden af, og andre har kun haft med i bagagen, hvad de har lært i indledende programmering.

Desværre har vi haft to gruppemedlemmer, der ikke har virket synderligt interesserede i at være med. De har været der til alle forelæsninger og til grupperegning, og i 3-ugers projektet har de stort set været der alle de timer hjælpelærerne har været til rådighed. Deres arbejdsindsats derimod, har ikke været tilsvarende den tid de har været fremmødt.

For at illustrere arbejdsindsatsen fra gruppen medlemmer lidt mere retfærdigt, har vi medtaget dette diagram fra **Github**, som vi har brugt til at samle vores kode i. Dette diagram viser aktiviteten over hele perioden.



Figur B.1: Use-case diagram, der viser de udleverede use-cases

Diagrammet kræver en kort forklaring da ikke alle gruppemedlemmer figurerer derpå, og at aktiviteterne er ret ujævnt fordelt. Vi starter med at fortælle hvilke alias, der dækker over hvilke gruppemedlemmer: DreamSpawn er Mikkel. Dickow er Jeppe. tmmortensen er Thomas. BlibberBlobber er Morten. MelikeY er Melike. Nikobirk er Nikolaj og LidaW er Lida.

Hvis man kigger på grafen, vil man se at Nikolaj ikke er med på diagrammet. Dette er fordi vi havde en del problemer med at få hans github til at køre. Derfor har han committed en del af sit arbejde med Mikkel, Morten og Jeppe. Vi fik dog hans git til at virke. Han figurerer, bare stadig ikke på graferne.

Morten har også lavet lidt med Jeppe. Derfor burde hans egen graf måske vise et lidt større tal.

Samtidig skal det siges at både Nikolaj og Morten har vist interesse for at lære noget, og hele tiden stillet sig selv til rådighed.

Jeppe og Thomas har også lavet lidt sammen.

Mikkel har som sagt lavet noget med Nikolaj. Hans graf viser dog en væsentlig højere aktivitet end de andre medlemmer. Dette har noget med erfaringsniveau at gøre. Mikkel har det lettere ved mange af tingene end os andre, og har derfor kunnet nå en del mere. Hvilket også har resulteret i et super flot program inden for tidsrammen.

Thomas har som sagt lavet lidt sammen med Jeppe.

Melike og Lida har arbejdet en del sammen. Dog kan man se at det kun er Melike, der rent faktisk har committed noget i løbet af hele perioden. Det skal siges at Thomas har committed et par linjer kode for dem begge. Nok noget der svarer til det commit, de i forvejen har.

Derfor må vi sige at graferne generelt giver et meget godt billede af arbejdsindsatsen. Selvfølgelig har der været nogle niveau forskelle, der er nødt til at blive modregnet, men der er langt fra 62 tegn i koden og så bare op til den næste med 2.916.

## C Brugervejledning

### Brugervejledning for afvejnings procedure



Figur C.1: *titelPicture*

## Installationsvejledning

Her er installationsvejledningen til afvejningssystemet.

### Administrationsdelen

For at køre Web-interfacet til administration, er det nødvendigt at have en **Tomcat-server** på den server, der skal køre administrationsdelen. Samtidig skal der installeres en **MySQL** server til at indeholde data. Dette kan gøres på samme maskine, eller på en separat, efter behov. Installation af disse kan findes på <http://tomcat.apache.org> for **Tomcat**-server og <http://www.mysql.com> for en **MySQL** server. Herefter lægger man den medfølgende .war fil ind i undermappen *webapps* i **Tomcat** server mappen *apache-tomcat-x.x.xx*. *x*'erne definerer versionsnummer på **Tomcat** installationen.

### ASE

Afvejningsenheden (**ASE**), er let at gå til. Her skal man blot ligge jar-filen ASE.jar over på den computer man ønsker at køre den fra og herefter dobbeltklikke for at starte den. Herefter gennemgår man sin afvejningsprocedure på vægten.

### Vægt simulator

Præcis som **ASE**'en kommer vægtsimulatoren som en jar-fil. Denne ligger man over på den ønskede computer og kører ved at dobbeltklikke.

## logIn

Når programmet åbnes via en web browser vil det være følgende billede som møder brugeren, for at logge ind, skal brugeren skrive sit bruger-id samt det tilhørende password. hvis det accepteres vil brugeren blive automatisk givet videre til hovedmenuen.

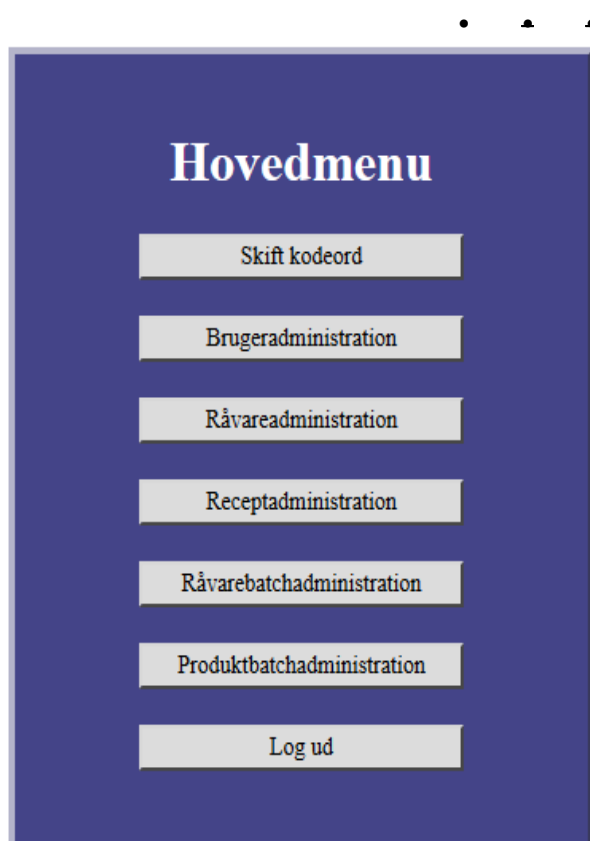


Figur C.2:

## Menu

Når man møder menuen, vil det ikke være det samme som møder brugeren, alt efter hvilken stilling man har i firmaet har man adgang til forskellige menu punkter.

Figur C.3:



- Admin
  - Skift kodeord
  - Brugeradministration
  - Råvareadministration
  - Receptadministration
  - Råvarebatchadministration
  - Produktbatchadministration
- Farmaceut
  - Skift kodeord
  - Råvareadministration
  - Receptadministration
  - Råvarebatchadministration
  - Produktbatchadministration
- Værkfører
  - Råvareadministration
  - Råvarebatchadministration
  - Produktbatchadministration
- Operatør
  - Skift kodeord

## Skift kodeord

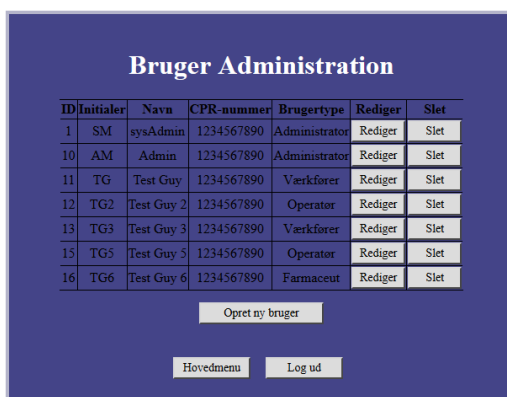
Ved valg af menupunktet skift kodeord, vil brugeren få vist følgende skærbillede, her skal brugeren indtaste sit gamle password, dette vil dog være auto udfyldt når man ankommer til siden. Det nye password skal overholde de gældende regler for passwords, for at blive accepteret



Figur C.4:

## bruger administration

Ved valg af bruger administration vil administratoren blive præsenteret med en menu, hvor det er muligt at se samtlige brugere, og der er mulighed for både at slette en bruger samt at redigere i en bruger. Administratoren har også mulighed for at oprette en ny bruger.



ID	Initialer	Navn	CPR-nummer	Brugertype	Rediger	Slet
1	SM	sysAdmin	1234567890	Administrator	Rediger	Slet
10	AM	Admin	1234567890	Administrator	Rediger	Slet
11	TG	Test Guy	1234567890	Værkfører	Rediger	Slet
12	TG2	Test Guy 2	1234567890	Operator	Rediger	Slet
13	TG3	Test Guy 3	1234567890	Værkfører	Rediger	Slet
15	TG5	Test Guy 5	1234567890	Operator	Rediger	Slet
16	TG6	Test Guy 6	1234567890	Farmaceut	Rediger	Slet

Figur C.5:

## bruger redigering/ bruger oprettelse

Hvis administratoren vælger at redigere en bruger vil han/hun blive præsenteret for en ny menu hvor alle informationer om den pågældende bruger er fyldt ind på forhånd, og det som ønskes at blive ændret kan bare blive overskrevet. Menuen for at oprette en ny bruger er identisk med den for at redigere, her vil ingen af værdierne være udfyldt på forhånd.

**Indtast nye brugeroplysninger**

Bruger ID	10
Bruger initialer	AM
Brugernavn	Admin
CPR	1234567890
Bruger Type	Administrator

Rediger

Tilbage Log ud

Figur C.6:

## recept liste

Når man vælger receptadministration, vil man først blive præsenteret for en liste af recepter, her har man så mulighed for at ændre i nuværende recepter, samt at slette recepter hvis de ikke har nogen receptkomponenter tilknyttet til deres id. Man kan også oprette nye recepter.

**Recept Administration**

ID	Navn	Rediger	Slet
20	vodkaJuice	Rediger	Slet

Opret ny recept

Hovedmenu Log ud

Figur C.7:

## recept

Vælger man at oprette eller ændre en nuværende recept vil menuen se ud som i følgende billede, her er det muligt for administratoren at tilføje flere recept komponenter og at ændre id'et på recepten, og også at ændre i allerede eksisterende recept komponenter.

**Indtast nye receptoplysninger**

Recept ID: 20

Receptnavn: vodkaJuice

Receptkomponenter:

	Råvare ID	Nettovægt	Tolerance
Slet	20	12.0	2.0
Ny			

Gem

Kontroller

Tilbage Log ud

Figur C.8: c

## Råvare administration

Hvis man vælger menuen Råvareadministration, vil man først blive præsenteret for en liste af alle råvarer i systemet, her vil der så være mulighed for at ændre en råvare, samt at oprette en ny, eller slette en råvare hvis den ikke længere bliver brugt.

**Råvarer Administration**

ID	Navn	Leverandør	Rediger	Slet
20	vodka	smirnoff	Rediger	Slet
21	vodka	karloff	Rediger	Slet
30	juice	rynkeby	Rediger	Slet

Opret ny råvare

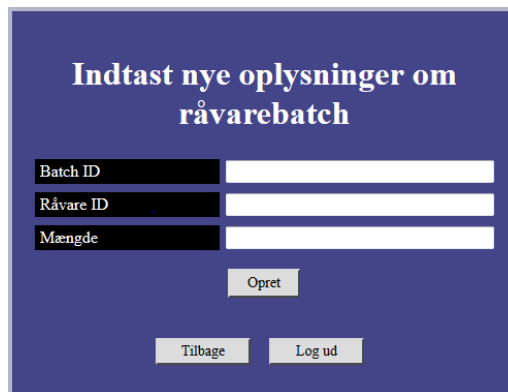
Hovedmenu Log ud

Figur C.9:



## Råvare batch

Når brugeren har valgt at oprette en ny råvarebatch, er det følgende menu som vil blive vist for brugeren, her kræves det bare at man giver et id, samt en eksisterende råvare fra systemet, og den mængde som der forefindes på lageret.



The screenshot shows a web form titled "Indtast nye oplysninger om råvarebatch" on a dark blue background. The form contains three input fields with labels "Batch ID", "Råvare ID", and "Mængde" in white text. Below the input fields is a yellow button labeled "Opret". At the bottom of the form are two yellow buttons labeled "Tilbage" and "Log ud".

Figur C.10:

## produkt batch

Når man vælger menu punktet produktbatch administration, vil brugeren blive præsenteret for en række af produktbatches som er oprettet i systemet, herfra vil brugeren have mulighed for at ændre i nuværende produktbatches samt at oprette nye.



The screenshot shows a web menu titled "Produktionsbatch Administration" on a dark blue background. At the top, there is a table header with three columns: "Batch ID", "Receipt ID", and "Status". Below the header is a yellow button labeled "Opret nyt produktbatch". At the bottom of the menu are two yellow buttons labeled "Hovedmenu" and "Log ud".

Figur C.11:

## Produkt batch (opret)

Når der skal oprettes et nyt produktbatch er det vigtigt at man ikke kan oprette en produktbatch hvor der ikke findes en tilsvarende receipt, men ellers er der mulighed for at tilføje produktbatches.

Figur C.12:

## afvejnings print

Et billede af et udprint som en operator vil modtage inden han/hun kan begynde afvejningen.

Udskrevet	2014-06-19
Produkt Batch ID	1
Receipt ID	1

Råvare ID	1
Råvare Navn	vand

Mængde	Tolerance	Tara	Netto	Råvare Batch	Operator
20.0	1.0				

Råvare ID	2
Råvare Navn	vand

Mængde	Tolerance	Tara	Netto	Råvare Batch	Operator
40.0	3.0				

Sum Tara	0.0
Sum Netto	0.0

Status	Opretet
Oprettelsesdato	2014-06-19

Tilbage

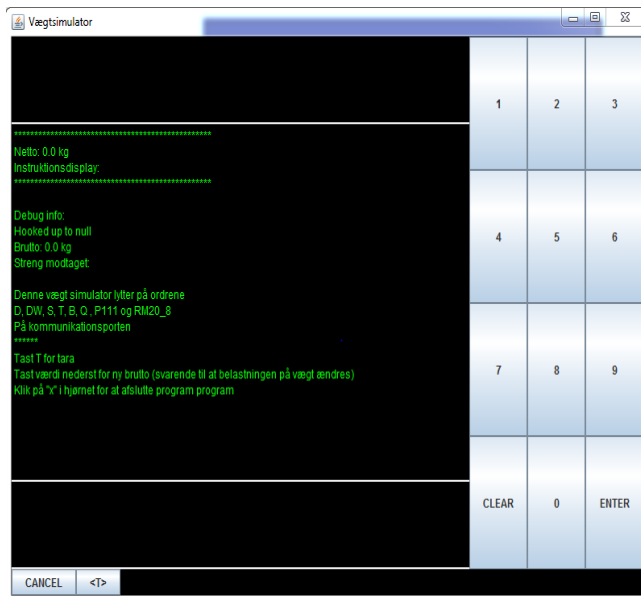
Figur C.13:

## D Simulator

Når man starter simulatoren bliver man mødt med et vindue, der viser informationer om hvilke ordre vægten lytter efter.

Figur D.1:

•    ▲    ▲    ▲    ▲ Øverste vindue



– Visning af input fra bruger

• Mellemvindue

- Forbindelses info
- Info om ordre
- Besked om afslutning
- Besked om vægt

• Nederste del

- Tara knap
- Input fra bruger
- Cancel knap

• Knapper

- Talcifre
- Clear input
- Enter godkender input

## E ASE

Bruges til at styre afvejningsproceduren. Det betyder at afvejningsproceduren fungerer på følgende måde:

- Operatøren har modtaget en produktionsforskrift på papir fra værkføreren.
- Operatøren vælger en afvejningsterminal.
- Operatøren indtaster operatør nr.
- Vægten svarer tilbage med operatørnavn som så godkendes.
- Operatøren indtaster produktbatch nummer.
- Vægten svarer tilbage med navn på recept der skal produceres (eks: saltvand med citron)
- Operatøren kontrollerer at vægten er ubelastet og trykker 'ok'
- Vægten tareres.
- Vægten beder om første tara beholder.
- Operatør placerer første tarabeholder og trykker 'ok'
- Vægten af tarabeholder registreres
- Vægten tareres.
- Vægten beder om raavarebatch nummer på første råvare.
- Operatøren afvejer op til den ønskede mængde og trykker 'ok'
- Pkt. 7 – 14 gentages indtil alle råvarer er afvejet.
- Systemet sætter produktbatch nummerets status til "Afsluttet".
- Det kan herefter genoptages af en ny operatør.