

CDIO 2

Vægt Simulator

af: Gruppe 12



Projekt i Videregående Programmering
DTU 2015

Lavet af:



Jonas Praëm (s144883)



Mads Justesen (s134834)



Alexander V. Pedersen (s145099)



Andreas Strange (s082853)



Charles Mathiesen (s974489)



Teeba Al-Dulaimi (s124178)



Tanja Hersborg (s090830)

1 Indledning

I denne del af CDIO 2 har vi fået til opgave, at skulle programmere en vægt simulator med en tilhørende konsol-brugerinterface til fjernkontrol af vægten.

Vægtsimulateren skal kunne simulere en Mettler BBK vægt. For hjælp til dette har vi ved opgavens start, modtaget et vedlagt program-skellet.

2 Projektplaner

2.1 Timeregnskab

2.2 Arbejdsgang

Vi har gennem udvikling af programmet gjort brug af **Git-hub** til at dele koden, og **Google Docs** til rapportskrivning. Gennem disse har det været muligt for alle gruppemedlemmer at følge med i hvor langt vi var i processen, både i form af kodning og rapport.

Til udarbejdelse af **UML**, har vi benyttet **Keynote**.

3 Kravspecifikation

3.1 Funktionelle krav

- Vægt Simulatorens skal benytte protokol SISC
- Simulatorens skal lytte på port 8000, Ved opstart fra kommandolinjen, skal denne default værdi kunne ændres til en vilkårlig værdi.
- Programmet skal indeholde samme kommandoer som Mettler toledos vægt
 1. RM20 8 2.
 2. P111
 3. S
 4. D "text"
- Der skal kunne indtastes en bruttovægt og tarere på vægt simulatoren
- Man skal i server klienten kunne indtaste svar efter at vægten har modtaget "RM20 8" ordren.
- Vægt simulatorens resurser skal være sikret korrekt lukkelse.

3.2 Ikke funktionelle krav

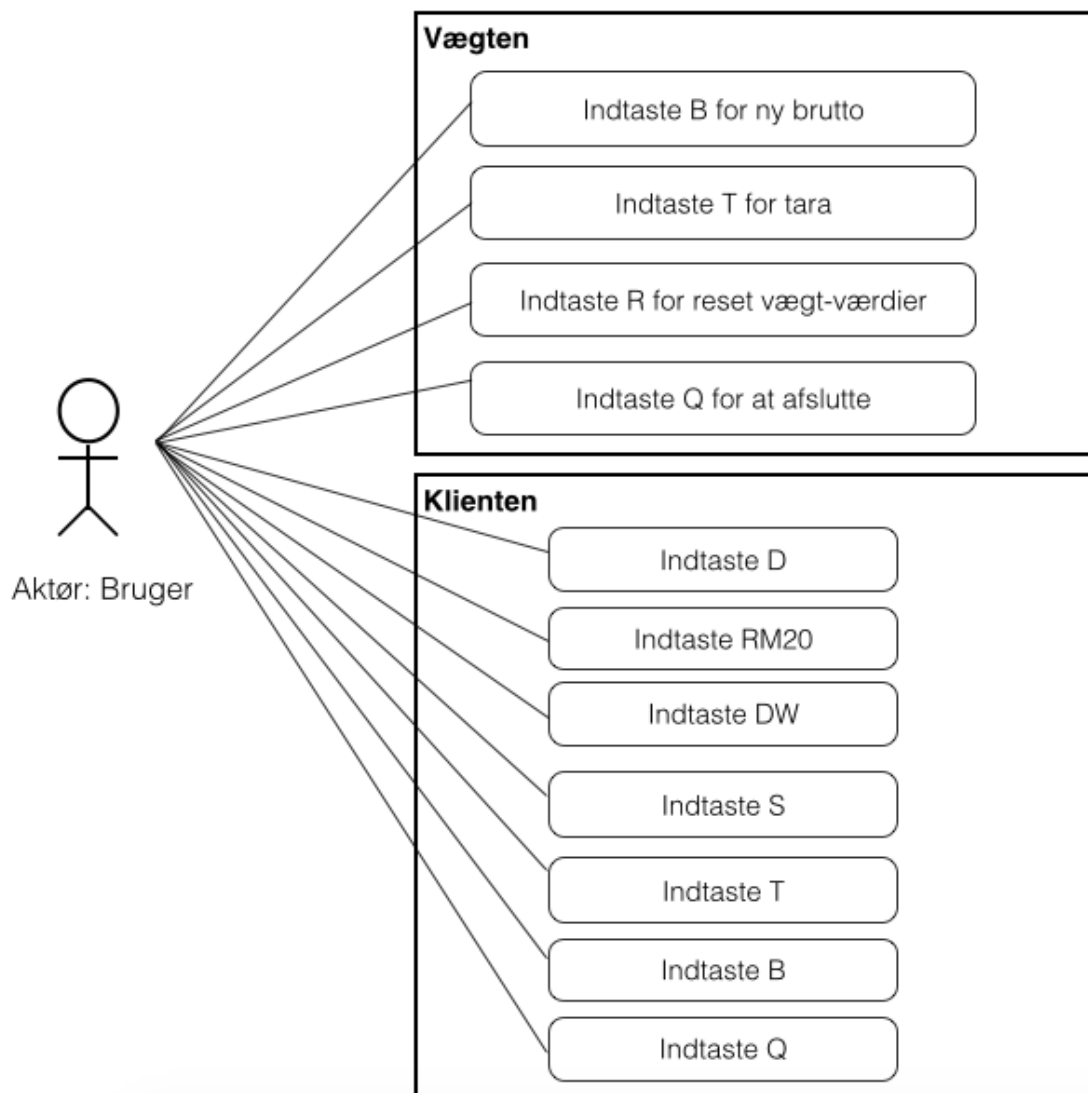
- Kodes i Java
- Efter eget valg skal programmet dokumenteres ved hjælp af UML artefakter.

- Opbygning efter 3-lags-modellen

4 Analyse

4.1 Use Case

Vi gennem projektet arbejde efter **UP** principperne, hvor en udarbejdelse af Use Case er væsentlig. Deraf blev der udarbejdet nedenstående Use Case diagram, der beskriver de handlinger der kan foretages i programmet.



4.2 Use case beskrivelser

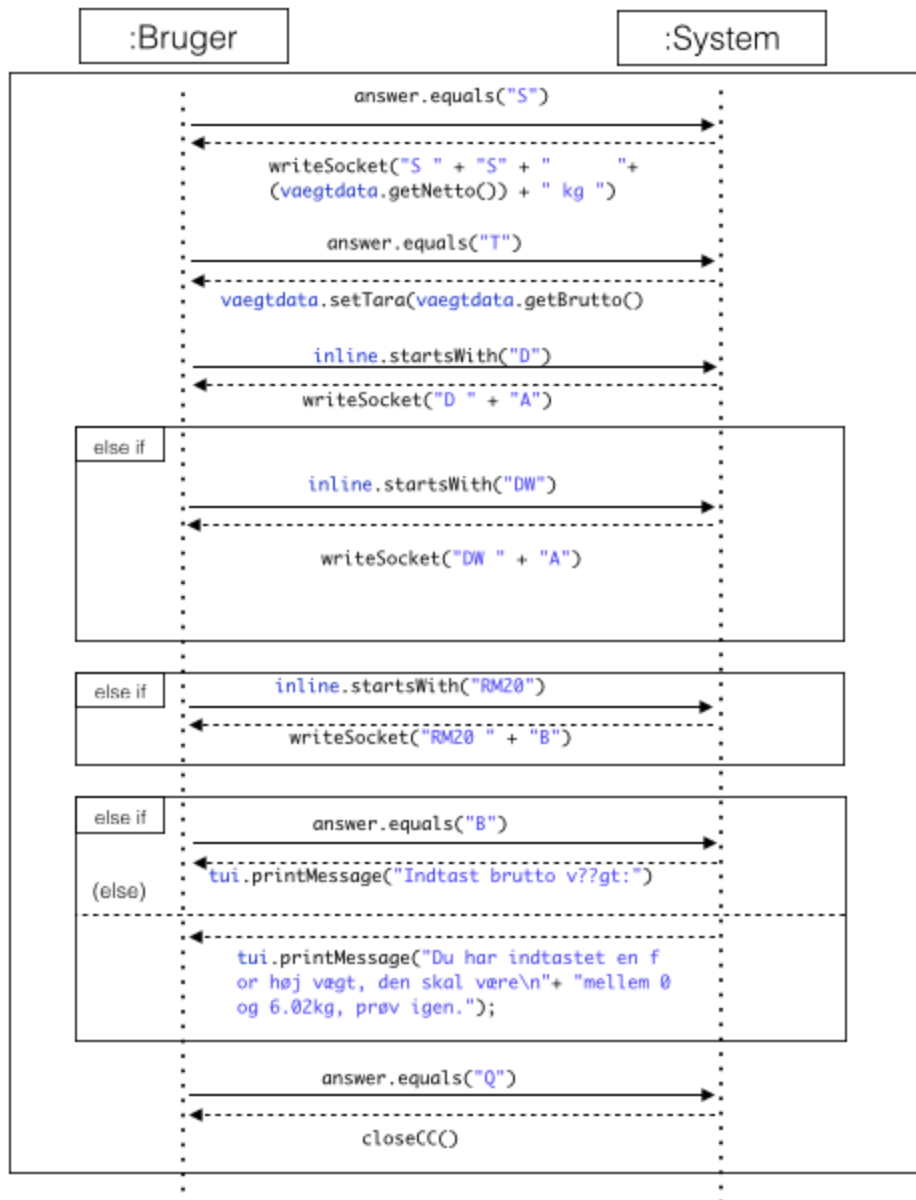
Da vores færdiggjorte produkt i dette projekt skulle være en server klient, bliver vi nødt til at se dette fra et admin perspektiv, da det er denne aktør som står for opretholdelse af en server klient.

Navn:	Bruger skriver kommando i telnet/anden klient
Aktører:	Bruger
Frekvens:	Hver gang bruger skal bruge information fra vægt
Startbetingelser:	Telnet klient, brugeren har kendskab til IP/port adressen på server klienten kører på.
Beskrivelse:	Når brugeren skal have fat i information fra vægten, skal brugeren benytte sig af et sæt af kommandoer til at kontakte vægten med. Heriblandt har vi en kommando til at aflæse vægtens nuværende værdi og ud over dette har vi en kommando til tara, altså nulstille vægten og gemme den værdi som vægten nu har fået fjernet, således at hvis brugeren fjerner en genstand fra vægten, vil vægten fremkomme som havende en negativ vægt.
Alternative flow:	Brugeren indtaster en kommando som ikke er eksisterende, altså en kommando som ikke er implementeret i vægten. Denne kommando vil kaste en exception, således at brugeren vil få en fejlbesked, at denne kommando ikke er eksisterende.
Illustrationer:	
Slutbetingelser:	Brugeren har modtaget den information som var ønsket.

4.2 Design sekvensdiagram

Her ses et design sekvensdiagram over det flow der forløber når brugeren indtaster de forskellige kommandoer. Eksempelvis vil systemet, når brugeren indtaster kommandoen "S", returnere en værdi i enheden kg.

Metoden *answer.equals()* tager imod indtastning af kommandoer, og metoden *writeSocket()* returnere værdien af kommandoerne. Det hele kører i et loop som er vist ved hjælp af en firkant rundt om alle flows. Else if, er de loop der bliver kørt inde i det store loop, som er vist ved hjælp af firkanter rundt om et enkelt flow for de dertilhørende metoder.



5 Implementering

5.1 Grænseflade

5.1.1 TUI

Dette er vores konsol grænseflade til brugeren. TUI'en er implementeret vha. en `InputStreamReader` omkranset af en `BufferedReader` på `system.in` til at opsnappe kommandoer tastet lokalt på "vægten" og sende dem videre til `ClientController`. Vores TUI har to modes at vise vægt-menuen med, en hvor det kun er vægt og tastatur, og en hvor det er vægt, debug og tastatur.

Der er indført en `clearScreen()` metode som sørger for vi har samme skærbillede i vores kommandoprompt hele tiden uanset hvilken mode brugeren har valgt at køre vægt-simulator som.

5.2 Funktionalitetslaget

Her beskriver vi de forskellige klasser vi har i vores controller pakke.

5.2.1 ClientController

Dette er kommandocentralen for vægtens menu, den sikrer at TUI får de korrekte værdier og der modtages brugerinput fra vægten fra TUI.

Herudover benyttes der en instans af `IOController` til at svare på RM20 beskeder modtaget fra den tilkoblede bruger.

5.2.2 IOController

Her bliver fjerntilslutning etableret og sendt videre til `WeightController` laget, det er også her der bestemmes hvilken port der skal aktiveres med `ServerSocket` ud fra hvad der er skrevet i kommandoprompt (hvis noget).

5.2.3 WeightController

Opfangelse af den fjerntilkoblede bruger indsamles og analyseres i dette lag, ud fra modtagne værdier modtager bruger en besked prompte baseret på hvad der forespurgtes om og hvis dette kunne accepteres som korrekt syntaks.

5.3 Datalaget

5.3.1 WeightData

Her opbevares samtlige data om vægtens nuværende tilstand og belastning, der opbevares også beskeder som skal sendes til brugeren af vægten, herunder de 4 tilstande displayet kan være i. 1) kun vægt, 2) vægt og RM20 besked, 3) vægt, RM20 og P111 besked, og tilsidst 4) 7 karakters tekst istedet for vægt, RM20 og P111, herudover kan der selvfølgelig forekomme kombinationer af disse muligheder.

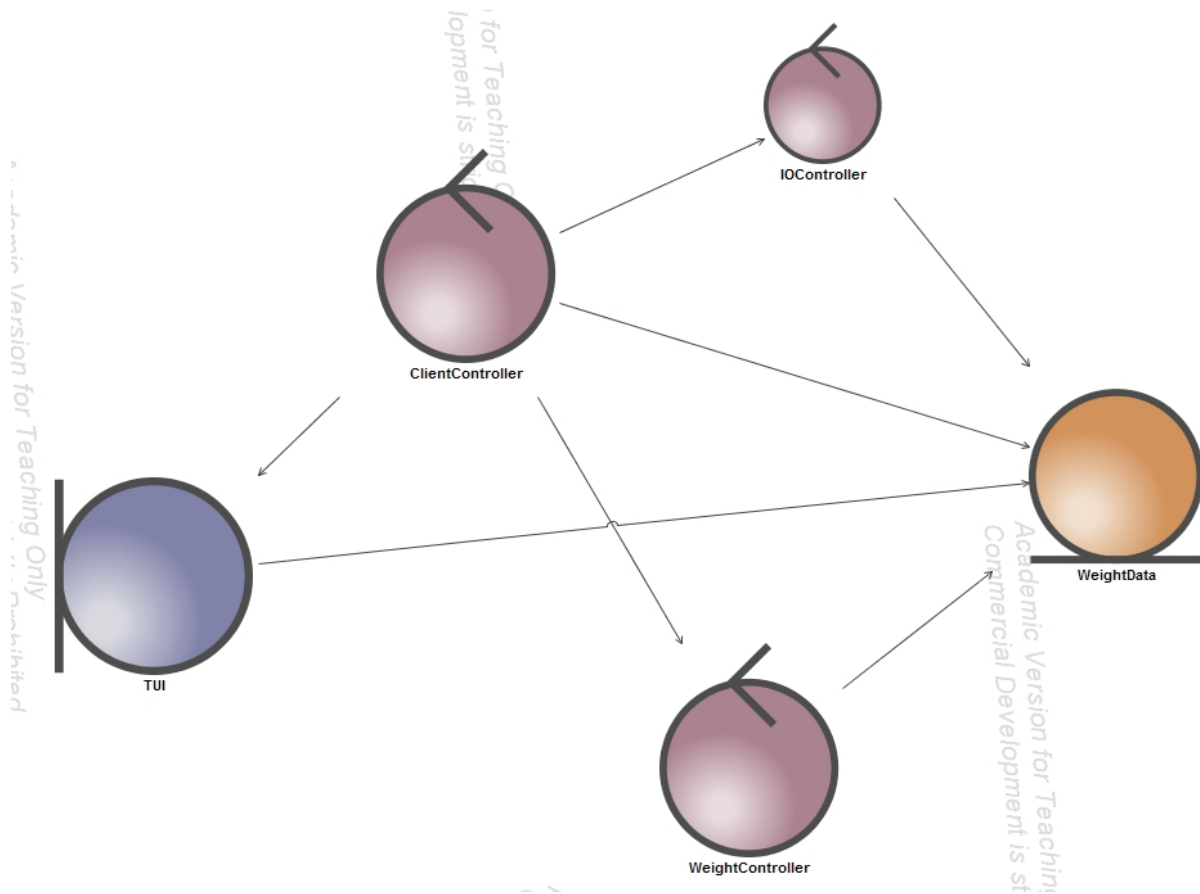
Der opbevares også valg foretaget ved eksekvering af programmet fra kommandoprompten, til brug i TUI og i `WeightController` laget.

6 Design

Design-Klasse diagram



BCE-Model



Client Controlleren er den overordnede controller, der håndterer alt input fra server klienten, Samtidig ved denne controller hvornår programmet skal lukke (når det repræsentative input fra brugeren, skal udføre en afslutning af programmet).

IOControlleren holder styr på input fra andre klienter. Denne controller fejlbehandler også input fra disse klienter.

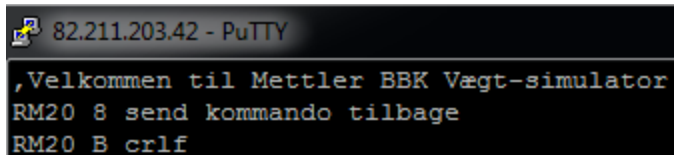
WeightControlleren sørger for at vægt simulatoren starter ordentligt op, og lukker korrekt ned. (lukker og åbner streams).

WeightData klassen holder styr på vægtsimulatorens virtuelle værdier, og sørger for, at de altid stemmer overens, uanset hvilken klient der tilgår variableerne.

7 Test

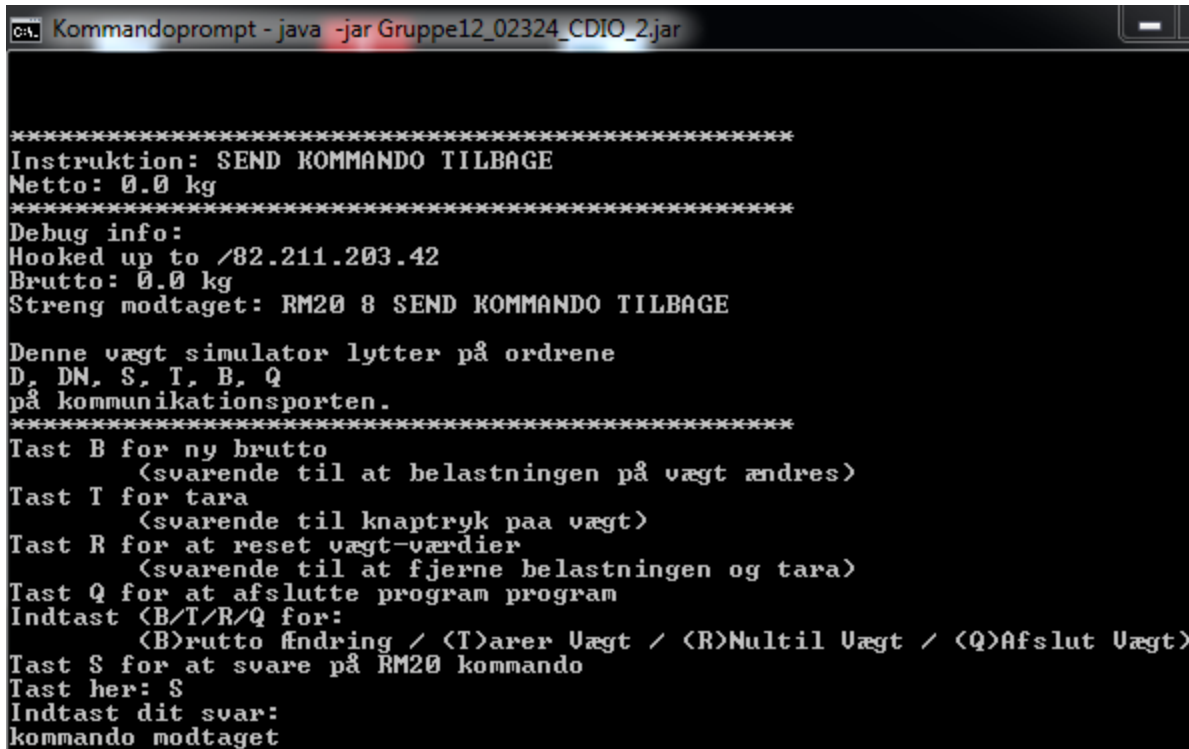
RM20.

Vi connecter først til vægten ved hjælp af putty og sender derefter en RM20 kommando.



```
82.211.203.42 - PuTTY
,Velkommen til Mettler BBK Vægt-simulator
RM20 8 send kommando tilbage
RM20 B crlf
```

RM20 B indikere at serveren har modtaget kommandoen.



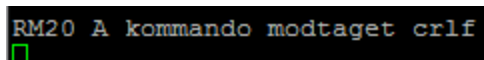
```
Kommandoprompt - java -jar Gruppe12_02324_CDIO_2.jar

*****
Instruktion: SEND KOMMANDO TILBAGE
Netto: 0.0 kg
*****
Debug info:
Hooked up to /82.211.203.42
Brutto: 0.0 kg
Streng modtaget: RM20 8 SEND KOMMANDO TILBAGE

Denne vægt simulator lytter på ordrene
D, DN, S, T, B, Q
på kommunikationsporten.
*****
Tast B for ny brutto
    (svarende til at belastningen på vægt ændres)
Tast T for tara
    (svarende til knaptryk paa vægt)
Tast R for at reset vægt-værdier
    (svarende til at fjerne belastningen og tara)
Tast Q for at afslutte program program
Indtast (B/T/R/Q for:
    (B)rutto ændring / (T)arer Vægt / (R)Nullil Vægt / (Q)Afslut Vægt)
Tast S for at svare på RM20 kommando
Tast her: S
Indtast dit svar:
kommando modtaget
```

I toppen kan vi se den instruktion som vi sendte fra putty.

Vi indtaster nu S og derefter vores svar.



```
RM20 A kommando modtaget crlf
█
```

putty har nu modtaget svaret fra vores server.

Vi prøver nu at sende en forkert rm20 kommando.



```
RM20
ES
```

der vil blive sendt en error besked tilbage og programmet vil vente på en ny kommando.

ændring af Tara og Brutto på vægt.

```
CHL Kommandoprompt - java -jar Gruppe12_02324_CDIO_2.jar

*****
Netto: 0.0 kg
*****
Debug info:
Hooked up to null
Brutto: 0.0 kg
Streng modtaget:
```

vi kan se på dette screenshot at vægten pt ikke har noget stående på sig.

Den er heller ikke tilkoblet en klient som kan sende kommandoer til den.

vi prøver nu at bruge vores indbygget kommandoer i vægten som tillader os at ændre vægten.

```
*****
Tast B for ny brutto
      <svarende til at belastningen på vægt ændres>
Tast T for tara
      <svarende til knaptryk paa vægt>
Tast R for at reset vægt-værdier
      <svarende til at fjerne belastningen og tara>
Tast Q for at afslutte program program
Indtast <B/T/R/Q for:
      <B>rutto ændring / <T>arer Vægt / <R>Nulltil Vægt / <Q>Afslut Vægt>
Tast her:
```

vi starter med at bruge B for at ændre brutto vægten.

```
Tast her: B
Indtast brutto vægt:
2
```

og kan derefter se at netto vægten har ændret sig.

```
*****
Netto: 2.0 kg
*****
Debug info:
Hooked up to null
Brutto: 2.0 kg
Streng modtaget:
```

vi Tara nu vægten ved at bruge kommandoen T.

```
*****
Netto: 0.0 kg
*****
Debug info:
Hooked up to null
Brutto: 2.0 kg
Streng modtaget:
```

vi kan prøve at tilføje mere vægt ved brug af kommandoen B igen.

```
*****
Netto: 4.0 kg
*****
Debug info:
Hooked up to null
Brutto: 6.0 kg
Streng modtaget:
```

som det kan ses på det sidste billede har vi tilføjet 4 kilo til vægten hvilket sammen med vores tara vægt på 2 bliver til brutto på 6. Til sidst kan vi resette vægten ved at indtaste kommandoen R.

```
*****  
Netto: 0.0 kg  
*****  
Debug info:  
Hooked up to null  
Brutto: 0.0 kg  
Streng modtaget:
```

Du kan manuelt fra vægten indtaste ny vægt og ændre tara. det skal dog nævnes at vægt simulatoren, lige som den rigtige vægt, har et grænse for hvad vægten kan være.

```
Tast her: B  
Indtast brutto vægt:  
6.7  
Du har indtastet en for høj vægt, den skal være  
mellem 0 og 6.02kg, prøv igen.
```

8 Manual

For at starte program i kommandoprompt findes følgende valgmuligheder:

- 1) java -jar Gruppe12_02324_CDIO_2.jar
- 2) java -jar Gruppe12_02324_CDIO_2.jar portnummer
- 3) java -jar Gruppe12_02324_CDIO_2.jar portnummer menuvalg

Portnummer bør være mellem 1025-65000

Menuvalg:

0 = kun vægtdisplay

1 = vægtdisplay og tastatur

2 = vægtdisplay, tastatur og debuginfo

Efter programmet er startet kan vægten styres fra en fjern-tilkoblet bruger eller lokalt fra vægten selv.

Som fjern-tilkoblet bruger har man følgende kommandoer til vægten:

D	skriv en besked på max 7 tegn til vægtens "vægt"-display
DW	fjern besked i "vægt"-display så der vises vægt igen.
B	tilføj en Bruttovægt mellem 0 og 6.02 kg.
S	nuværende Netto-vægt på vægten.
T	tarer vægten
RM20 8	send en besked til bruger der påkræver et svar fra brugeren.
P111	send en besked til det tredje display på vægten.
Q	afslut programmet

Lokal bruger på vægten har følgende muligheder:

B	indtast en ny bruttovægt.
T	tarer vægten
R	nulstil vægt (fjern vægten helt)
Q	afslut programmet
S	når en RM20 kommando er modtaget, benyttes S til at svare på beskeden.

Konklusion

Det er lykkedes at lave en vægt simulator der kan opereres lokalt og som kan tilgås udefra via en klient. Alle ønskede funktioner er implementeret. Programmet er opbygget efter 3 - lags modellen. I grænseflade laget findes klassen TUI der implementerer IUI. I funktionalitets laget findes alle controllere, IOController, WeightController, ClientController, der også implementerer deres respektive Interfaces. I datalaget findes klassen WeightData der implementerer IWeightData. Til at fange fejl er der oprettet 3 exception-klasser, hhv. InputLengthException, UnknownInputException, og UnsupportedWeightException. Programmet kan køres gennem kommandoprompt hvor man efter ønske kan tilføje argumenter der vælger port nr og om programmet skal køres med menu. Der køres 4 tråde i dette program, Main, garbage collector, IOController og ClientController.