

**Projektnavn:** CDIO 2

**Gruppenummer:** 18

**Afleveringsfrist:** Mandag d. 30/3 - 2015

*Denne rapport er afleveret via Campusnet (der skrives ikke under)*

*Denne rapport indeholder sider inkl. denne side.*



**Studienummer:**

s144839

**Navn:**

Liban Jama Awil Dirir

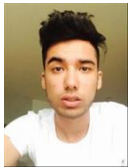


**Studienummer:**

s144859

**Navn:**

Kenneth Skovsgaard Frandsen



**Studienummer:**

s144841

**Navn:**

Nael Rashdeen



**Studienummer:**

s144849

**Navn:**

Delvaux Paulin Richard Niyonzima



**Studienummer:**

s133275

**Navn:**

Peter Asmussen



**Studienummer:**

s123503

**Navn:**

Thomas Liljegren



**Studienummer:**

s123172

**Navn:**

Martin Vieth

[Indledning](#)

[Kravspecifikation](#)

[Implementering](#)

[Simulator \(Peter\)](#)

[Flere inputs på samme tid \(Thomas\)](#)

[Threads i systemet \(Kenneth\)](#)

[Lukning af ressourcer\(Martin\)](#)

[Vægt kommandoer \(Peter\)](#)

[Dokumentation](#)

[RM20\\_8 \(Thomas\)](#)

[Tara](#)

[Q \(Quit\) \(Peter\)](#)

[Brutto \(Peter\)](#)

[Klassediagram \(Kenneth\)](#)

[Bilag](#)

[Bilag 1.](#)

## Indledning

CDIO del 2 handler om opkobling til vægt simulatoren. Vi har i dette projekt arbejdet med at simulere vægten og opkobling af brugere til denne simulator.

Vi har fået udleveret et program skellet for en vægt simulator, som vi har bygget videre på. Vi kobler os op til simulatoren via telnet og udfører kommandoerne igennem terminalen.

Simulatoren har forskellige funktioner som følger SISC protokollen.

Vi har i denne opgave lagt vægt på at arbejde med threads og gøre vores simulator istand til at behandle mange brugere på samme tid samt korrekt lukning af programmet.

Git repo: [https://github.com/ProjectGroup-CDIO/18\\_CDIO2](https://github.com/ProjectGroup-CDIO/18_CDIO2)

# Kravspekifikation

Der skal laves en vægt simulator med konsol-brugerinterface. Programmet skal simulere en Mettler BBK vægt som skal kunne modtage ordre fra for eksempel en telnet applikation.

Systemet skal overholde følgende krav:

- En vægt simulator
- 6 kommandoer fra den vedlagte (SISC\_protokol.xlsx)
- Implementer B som kommando til at ændre belastning på vægten
- Implementer Q til at afslutte simulator
- Implementer T som kommando til at tarere vægten.
- Skal lytte på port 8000 samt ved opstart fra kommandolinjen skal man kunne overskrive port 8000 da den er default
- Tilføje funktionalitet og give mulighed for at kunne ændre brutto belastning
- Implementere mulighed for at indtaste svar efter at vægten har modtaget "RM20 8" ordren.
- Sikre at programmet lukkes ordentligt ned når det afsluttes
- Skal kunne køre fra kommandolinjen.

# Implementering

## Simulator (Peter)

Simulator-klassen fungerer som bruger-grænsefladen for vores vægt-applikation.

```
public static void printmenu(){
    for (int i=0;i<25;i++) System.out.println(" ");
    System.out.println("*****");
    System.out.println("Netto: " + (brutto-tara)+ " kg" );
    System.out.println("Weightdisplay: " + weightDisplay);
    System.out.println("Instruktionsdisplay: " + instruktionsDisplay );
    System.out.println("*****");
    System.out.println(" ");
    System.out.println(" ");
    System.out.println("Debug info: ");
    System.out.print("Hooked up to: ");
    try {
        for(ClientInput client : clientList) {
            System.out.println("\n"+client.getSocket().getInetAddress());
        }
    } catch (NullPointerException e) {
        System.out.println("Hooked up to n/a");
    } catch (Exception e) {
        //e.printStackTrace();
    }
    System.out.println("Brutto: " + (brutto)+ " kg" );
    System.out.println(" ");
    System.out.println("Denne vegt simulator lytter på ordrene ");
    System.out.println("D, RM20 8, S, T, B, Q ");
    System.out.println("paa kommunikationsporten. ");
    System.out.println("*****");
    System.out.println("Tast T for tara (svarende til knaptryk på vægt)");
    System.out.println("Tast B for ny brutto (svarende til at belastningen på vægt ændres)");
    System.out.println("Tast Q for at afslutte program program");
    System.out.println("Indtast (T/B/Q for knaptryk / brutto ændring / quit)");
    System.out.print ("Tast her: ");
}
```

Her ses hovedmenuen som bliver printet ud, når brugeren åbner for programmet. Brugeren vil først få en del information, eksempelvis nettovægten og den vægt som displayet viser.

I midten af koden ses en try{ / catch{. Hvis brugeren er koblet op, vil systemet printe IP-adressen fra den socket, klienten er forbundet til - dvs. at det er vægts/simulatorens IP der bliver printet.

Hvis brugeren er forbundet til localhost, vil man få udskriften "Hooked up to n/a" i stedet for. Efterfølgende vil brugeren få en række muligheder, eksempelvis at sætte tara og ny brutto.

## Flere inputs på samme tid (Thomas)

For at få programmet til at tage imod flere inputs, fra flere forskellige kilder på samme tid, har vi gjort brug af Javas Threads. Threads bruges ved at extend Thread eller, hvis der allerede arves fra en klasse, implementere Runnable. Når det er gjort, skal klassen have en `run()` metode, der sættes i gang i det `thread.start()` kaldes.

Der er lavet en `SimInput`, og en `ClientInput` til at tage imod input fra henholdsvis simulator og client. Når programmet startes op, bedes brugeren om at specificere en port til serverens socket. I terminalen ser det sådan ud:

```
Thomas:desktop Thomas$ java -jar WeightSim.jar
Indtast ønsket port# eller tryk ENTER for port 8000
5450
Server venter på forbindelse på port 5450
-
```

Herefter stiller server-threaden sig til at lytte på den givne port. Samtidig kører en thread med `SimInput`, der er input fra simulatorens konsol. Nedenfor ses et udsnit af `SimInputs run()` metode:

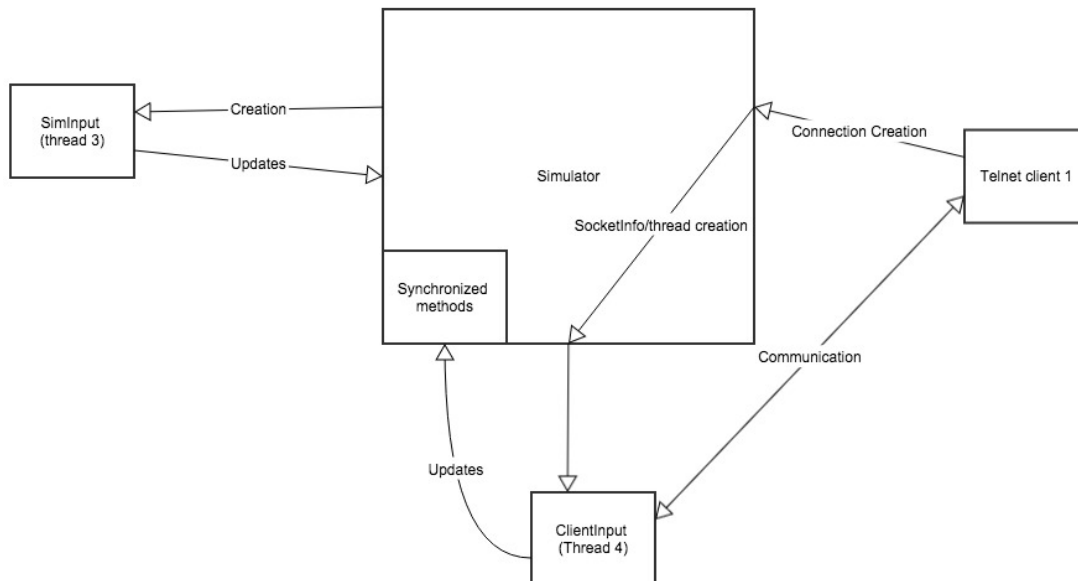
```
· else if(input.toUpperCase().startsWith("B")) {
    String temp = input.substring(1, input.length()).trim();
    if(temp.length() < 7) {
        double tempDouble;
        try {
            tempDouble = Double.parseDouble(temp);
            Simulator.setBrutto(tempDouble);
        } catch (NumberFormatException e) {
            System.out.println("Error: "+e.getMessage());
            e.printStackTrace();
        }
        Simulator.printmenu();
    }
}
```

Der tjekkes om input overholder reglerne for længde, hvorefter 'B' og mellemrum trimmes af, og der parses til double. I tilfælde af, at input f.eks. ikke indeholder tal fanges `NumberFormatException`, og løkken starter forfra.

På [bilag 1](#) kan man se en figur der hjælper med at vise hvordan programmet opretter threads til klienter der forbinder igennem f.eks. en telnet klient.

## Threads i systemet (Kenneth)

Versionen for én client.



Ovenstående billede er hvordan programmet connecter med de forskellige klasser i vores vægt simulator. Når programmet kører, og der er en klient forbundet, er der 4 threads i gang. De 2 første threads i vores system er Main og Garbage collection, som altid er 2 threads der kører i Java, andre programmeringssprog vil de ikke altid forefindes på samme måde.

Thread 3 i vores program er SimInput

SimInput, som er det input "vægten" giver i form af vægt "brutto"vægt.

Thread 4 - thread  $n$  i vores program er ClientInput som vist ovenfor.

ClientInput "oprettes" det antal gange som der er bruger logget ind, men vi opretter kun en i vores tilfælde som er vist på ovenstående billede. Hvis man opretter flere ville de se ud som vores [Bilag 1](#).

Ved at vi kører Thread giver det mulighed for vi kan have flere bruger på den samme vægt dette kan dog max være 64511 da 0-1024 er reserveret og der er i alt 65535 port man kan bruge. [Kilde](#)

## Lukning af ressourcer(Martin)

I dette program bruger vi threads til at udføre opgaver samtidigt. Hver af disse threads bruger en række af ressourcer som ved afslutning af programmet skal lukkes ned så de ikke kører i baggrunden efter afslutning af programmet.

Ved kommando Q i `ClientInput` vil der blive kørt en række af kommandoer som lukker de ressourcer som denne thread bruger samt lukning af `Siminput` ved `stopGracefully()` metoden.

```
else if ((inline.equals("Q"))){
    System.out.println("");
    System.out.println("Program stoppet Q modtaget");
    SimInput.stopGracefully();
    Simulator.stopLoop();
    Simulator.closedSockets();
    instream.close();
    outstream.close();
    System.in.close();
    System.out.close();
    System.exit(0);
}
```

For at sikre at alle igangværende threads bliver lukket ned har vi opsat vores program til at lukke threads når de ikke længere bruges korrekt eller har udført deres formål.

`ClientInput` threads lukkes når der ikke længere er en forbindelse mellem brugeren via telnet forbindelse eller lignende og simulatoren. Vi lukker den givne thread ved at se efter en `NullPointerException`.

```
}catch(NullPointerException e1){
    //when a client terminates his connection i.e closes his computer or con
    //the connection is set to null -> this means that we have to handle the
    System.out.println("\nConnection has been terminated, closing thread");
    break;
}
```

Det der afslutter den givne thread er break linjen, denne lukker threaden ved at gå ud af while løkken og dermed afsluttes threaden.

`SimInput` threaden afsluttes ved at bruge `stopGracefully()`, denne gør at while løkken ikke længere kører og dermed afslutter threads.



## Vægt kommandoer (Peter)

Når en bruger kører programmet, vil brugeren få en lang række muligheder for at udføre en masse kommandoer. Disse er beskrevet nedenunder billedet.

```
System.out.println("Denne vægt simulator lytter på ordrene");
System.out.println("D, RM20 8, S, T, B, Q");
System.out.println("på kommunikationsporten.");
System.out.println("*****");
System.out.println("Tast T for tara (svarende til knaptryk på vægt)");
System.out.println("Tast B for ny brutto (svarende til at belastningen på vægt ændres)");
System.out.println("Tast Q for at afslutte program program");
System.out.println("Indtast (T/B/Q for knaptryk / brutto ændring / quit)");
System.out.print ("Tast her: ");
```

### D

- Syntaks: D "String" (maks 7 karakterer)
- Ændrer vægtens display-tekst.
- Eksempel på gyldig kommando:

- D "HEJ"

### S

- Syntaks: S
- Printer vægtens pålagte vægt, brutto-tara = S.

### T

- Syntaks: T
- Nulstiller vægtens nettovægt, ved at sætte Tara til at være lig brutto

- T

### B

- Syntaks: B <double>
- Giver brugeren mulighed for at ændre bruttovægten.
- Det skal være et tal, minimum 3 karakterer, maks 7 karakterer.
- Eksempler på gyldige inputs:
  - 0.001
  - 1000
  - 10.1
  - 10.01

## Q

- Syntaks: Q
- Lukker vægt-simulatoren totalt

```
else if ((inline.startsWith("Q"))){  
    System.out.println("");  
    System.out.println("Program stoppet Q modtaget paa com port");  
    System.in.close();  
    System.out.close();  
    instream.close();  
    ostream.close();  
    System.exit(0);  
}
```

## RM20 8

- Syntaks: RM20 8 "string" "string" "string" (maks 30 karakterer)
- Ændrer vægtens instruktions display med den indtastede tekst
- Eksempel:
  - RM20 8 "This" "is" "cool"

## P111

- Syntaks: P111 "string" (maks 35 karakterer)
- Ændrer vægtens instruktions display med den indtastede tekst
- Eksempel:
  - P111 "This is cool"

## Fejlindtastninger

- Ved fejlindtastning (at man giver simulatoren et input, som vi ikke har defineret i koden), bliver der sendt beskeden S.

## Dokumentation

### RM20\_8 (Thomas)

Den ene RM20 kommando vi har implementeret i dette projekt, er 'RM20 8' kommandoen. Kommandoen skal sende en besked til vægtsens instruktions display, og derefter skal der være mulighed for at svare på den fra vægtsimulatorens terminal/konsol. For at checke, at hver kommando følger protokollens krav, bruges `checkRM20` metoden. Herunder skrives en kommando der opfylder protokollens krav:

```
Thomas:~ Thomas$ telnet localhost 8000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
RM20 8 "hej" "med" "dig"
```

og den går igennem som den skal,

```
*****
Netto: 0.0 kg
Weightdisplay:
Instruktionsdisplay: "HEJ" "MED" "DIG"
*****

Debug info:
Hooked up to n/a
Brutto: 0.0 kg
Streng modtaget: null

Denne vegt simulator lytter på ordrene
D, RM20 8, S, T, B, Q
paa kommunikationsporten.
*****
Tast T for tara (svarende til knaptryk på vægt)
Tast B for ny brutto (svarende til at belastningen på vægt ændres)
Tast Q for at afslutte program program
Indtast (T/B/Q for knaptryk / brutto ændring / quit)
Tast her:
Besked modtaget. Se instruktionsdisplay.
Tryk ENTER og derefter dit svar
Hej hej
|
```

hvor beskeden vises i det ønskede felt og svaret kommer tilbage som det skal:

```
Thomas:~ Thomas$ telnet localhost 8000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
RM20 8 "hej" "med" "dig"
RM20 A "Hej hej"
-
```

Hvis ikke protokollen overholdes, svarer simulatoren med fejlbeskeden 'S'.

```
RM20 8 "hej" "med" "dig"
S
RM20 8 hej "meD" D"dig"
S
RM20 8 "hej" "med" "dig"
S
RM20 8 "hej" "med" "dig"
S
-
```

## Tara

Tara kommandoen skal kunne vise det antal kilo som er på vægten. Desuden har vi implementeret for at vi kan starte på en bestemt vægt. Men når man opretter forbindelse til vægten og indtaster T i terminalen er dette outputtet:

```
Libans-MacBook-Pro:~ LibanDirir$ telnet localhost 8000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
T
T S      0.0kg
█
```

Menneskelige fejl kan altid ske. Menneskelige fejl i form af at man har indtastet forkert er outputtet følgende:

```
Libans-MacBook-Pro:~ LibanDirir$ telnet localhost 8000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
T
T S      0.0kg
T ABC
S
█
```

Outputtet til denne fejl er S. Som det står i kravspecifikationen. Man kan desuden se, at denne fejl altid vil forekomme så længe T er det første input i terminalen.

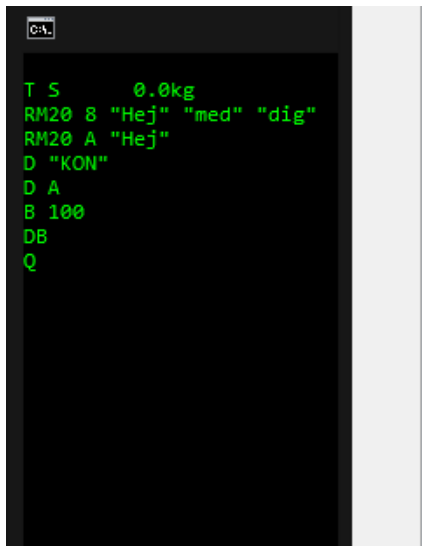
```

Libans-MacBook-Pro:~ LibanDirir$ telnet localhost 8000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
T
T S      0.0kg
T ABC
S
TT
S
█

```

## Q (Quit) (Peter)

Q er kommandoen, som stopper forbindelsen mellem klienten og vægten.



```

T S      0.0kg
RM20 8 "Hej" "med" "dig"
RM20 A "Hej"
D "KON"
D A
B 100
DB
Q

```

Debug info:

Hooked up to n/a

Brutto: 100.0 kg

Streng modtaget: null

Denne vegt simulator lytter på ordrene  
D, RM20 8, S, T, B, Q  
paa kommunikationsporten.

\*\*\*\*\*

Tast T for tara (svarende til knaptryk på vægt)  
Tast B for ny brutto (svarende til at belastningen på vægt ændres)  
Tast Q for at afslutte program program  
Indtast (T/B/Q for knaptryk / brutto ændring / quit)  
Tast her:  
Program stoppet Q modtaget

Connection has been terminated, closing thread  
# of clients connected: 1

Efter at have indtastet en række kommandoer, kan man efterfølgende skrive kommandoen **Q**, og forbindelsen vil blive lukket.

## Brutto (Peter)

Brutto kommandoen skal kunne sætte vægts bruttovægt fra klienten. Dette sker ved, som der står beskrevet i kommando-forklaringerne i afsnittet ovenfor, at man skriver **B**, efterfulgt af den vægt, man kunne ønske sig. Vi har i koden defineret, at hvis kommandoen går igennem, svarer simulatoren med et "DB", og et "S" hvis der er sket en fejl.

Ved korrekt indtastning af kommando, kunne det se således ud:

```
Telnet localhost

S
B 100
DB
T
T S      100.0kg
B 10.32
DB
T
T S      10.32kg
B 190.23
DB
T
T S      190.23kg
```

For at være sikker på, at inputtet gik igennem, printer vi vægten med Tara-kommandoen.

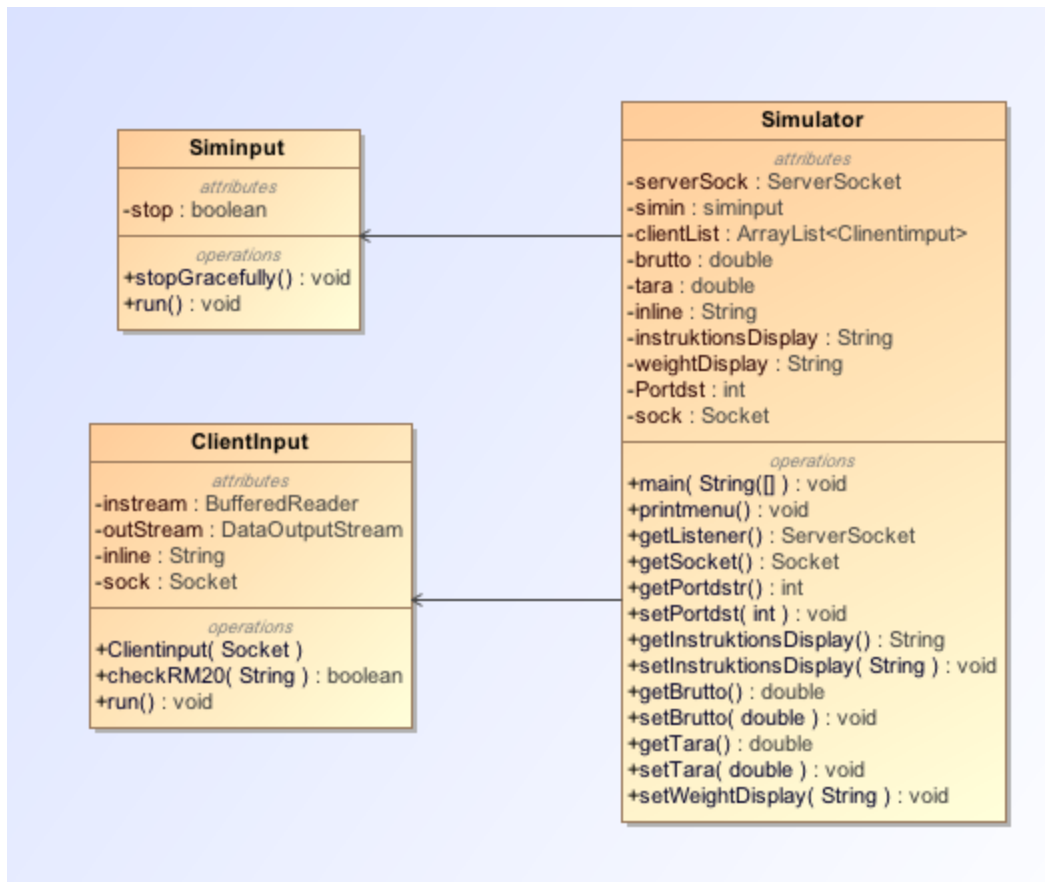
Hvis man ikke skriver brutto-kommandoen korrekt, vil man få et "S", som betyder at man har lavet en fejl:

```
Telnet localhost

S S      0.0 kg
B 10A23
S
B 10.0
DB
B 100
DB
DB
T
T S      100.0kg
```

Der ses, at første kommando er forkert, og vi får svaret "S" fra simulatoren. En lille fejl vi dog ikke kan få rettet er, at hvis vi i kommandoen skriver en integer i stedet for en double, vil vi få svaret "DB" to gange. Dette betyder dog ikke noget, da simulatoren stadigvæk tager imod kommandoen og opdaterer brutto-vægten, som også er vist vha. Tara-kommandoen.

## Klassediagram (Kenneth)



Herover ses vores KlasseDiagram, som viser hvad de forskellige klasser indeholder i vægt simulatoren.

SimInput er den klasse der simulerer selve vægt-delen, hvor man kan øge eller reducere, for at ændre totalvægt på "vægten". SimInput bliver der kun "oprettet" en af, så hvis der er flere brugere logget på vægten på samme tid, ville de begge påvirke den samme vægt, dog køre det med synchronize som gør, at kun en kan kommunikerer med vægten ad gangen.

Clientinput er klassen der "oprettes" sammen med den bruger der er logget på Simulatoren, her kan der være flere brugere som kan påvirke den samme vægt.

Vores Main klasse er Simulator klassen, som er den der kører main metoden og som holder styr på vores program og gør at programmet bliver kørt i rigtig rækkefølge.

## Fejl og mangler

Ved afslutning af programmet fra en client med kommando Q bliver programmet først lukket.

Efter et input er sendt til en ventende scanner nextline i `simInput`, For at afslutte programmet helt skal man i terminalen trykke enter.

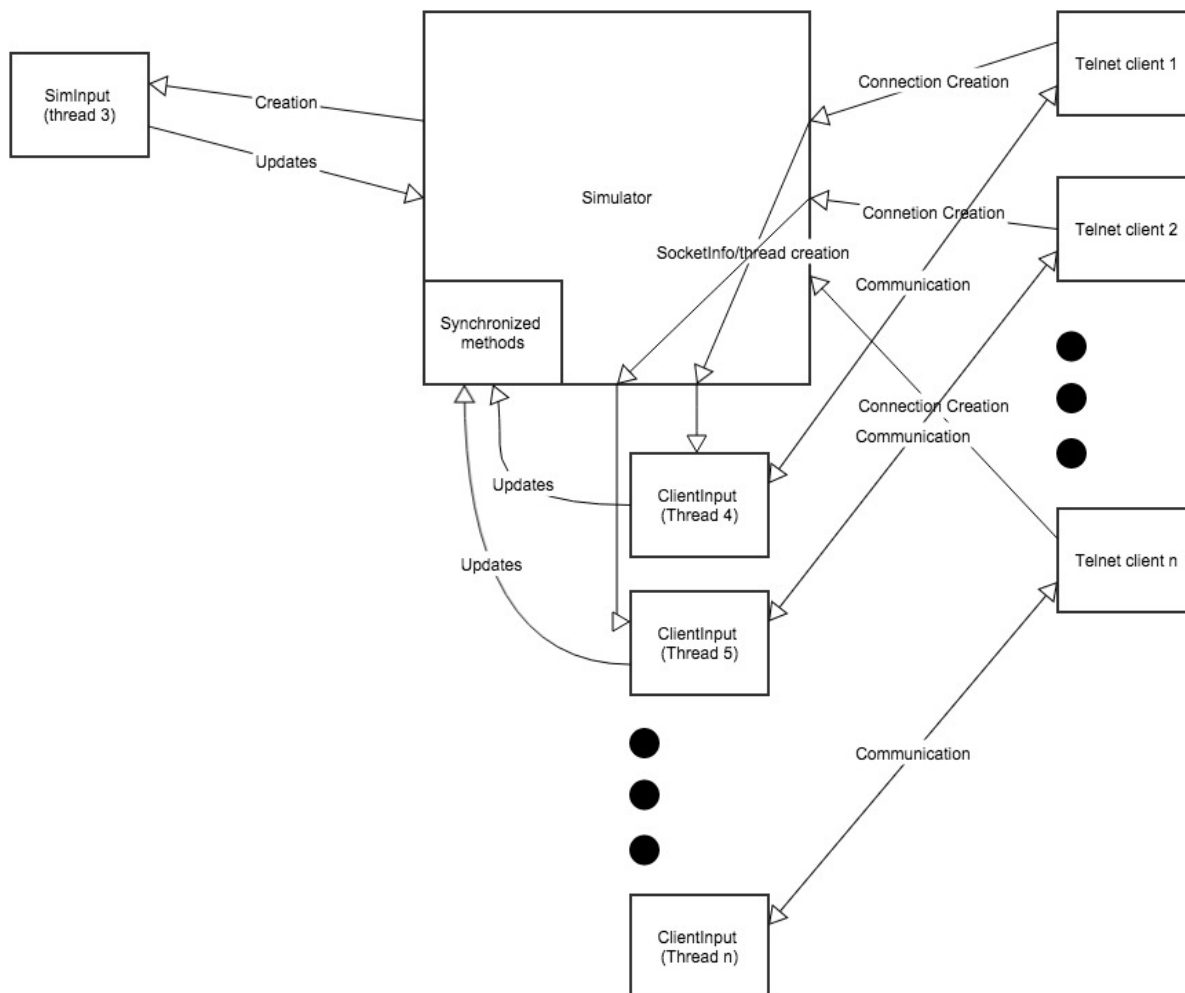
Ved indtastning af brutto som fx 100 eller 10 vil den skrive 2 gange at den er accepteret.

Dette er en mindre fejl som ikke har nogen effekt på programmet.



## Bilag

### Bilag 1.



Figuren viser hvordan simulatoren interagerer med  $n$  telnet klienter.