CDIO Del 1

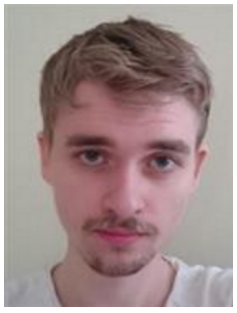
Afvejningssystem
Gruppe 15

(a) s153475
Lukas Bek

(b) s144850
Lars Quaade

(c) s154306
Sebastian Hoppe

(d) s153947
Magnus Haakonson



(e) s153356
William Wiberg

(f) s090830
Tanja Pajlina

Danmarks Tekniske Universitet
Softwareteknologi Diplom
Videregående programmering 02324
Stig Høgh

9. April 2016

Time Regnskab

Navn	Dok.	Design	Impl.	Test	Andet	I alt
Lukas Bek	4	5	10	0	0	19
Sebastian Hoppe	4	2	4	7	2	19
Magnus Haakonsson	6	3	10	0	0	19
Lars Quaade	3	5	7	4	0	19
William Wiberg	4	6	5	0	4	19
Tanja Pajlina	4	4	8	0	3	19

Indhold

Indledning

I denne CDIO-opgave har vores gruppe, fået til opgave at benytte vores viden fra kurset 02325 som hjælp til at udføre den obligatoriske opgave. Opgaven går ud på at programmere en vægtsimulator med en tilhørende konsol-bruger interface til fjernkontrol af vægten. Vi skulle kunne simulere en Mettler BBK vægt til hjælp af dette, har vi fået udleveret et program-skelet ved opgavens start.

Opgaven

Opgavebeskrivelsen kan findes på klassens fællesdrev i google-docs 02324 Tilgængeligt på CN:

`https://drive.google.com/folderview?id=0B1qlt-Xd6kaQRVpuYThCVUhCbkk&usp=sharing&tid=0B1qlt-Xd6kaQalk4REc5Yk5ZWWM`

Kravspecificering

Funktionelle krav:

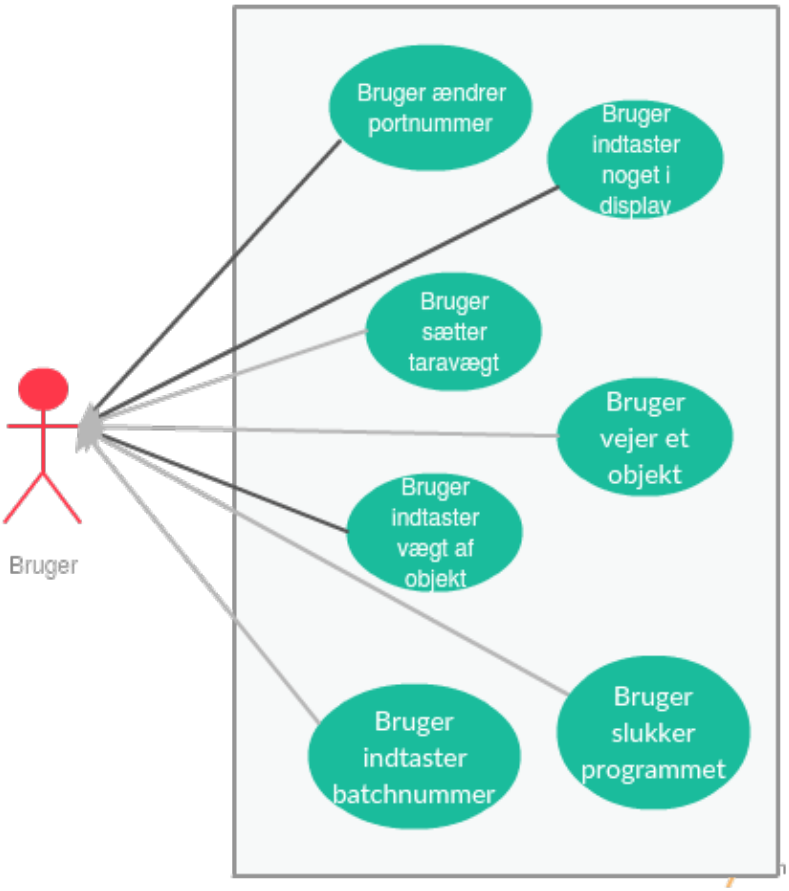
- Skal bruges med konsol-brugerinterface (sort DOS vindue)
- Simulatoren skal lytte på port 8000. Ved opstart fra kommandolinjen, skal denne kunne overskrives med et vilkårligt port-nummer.
- Programmet skal kunne startes og sluttes
- Vægten skal kunne modtage forskellige kommandoer:
  RM, D, DW, T, B, Q, S
  Disse står yderligere beskrevet under afsnittet "Vægten"i rapporten.
- Klienten skal kunne oprette forbindelse til vægten og sende input
- Vægtsimulatoren skal benytte protokol SISC
- Java Version 8 update 73
- Eclipse Version: Mars Release (4.5.1) ...

Non Funktionelle krav:

- Opbygges efter 3-lags modellen
- Kodes i Java
- Usecases skal benyttes til at beskrive programmet

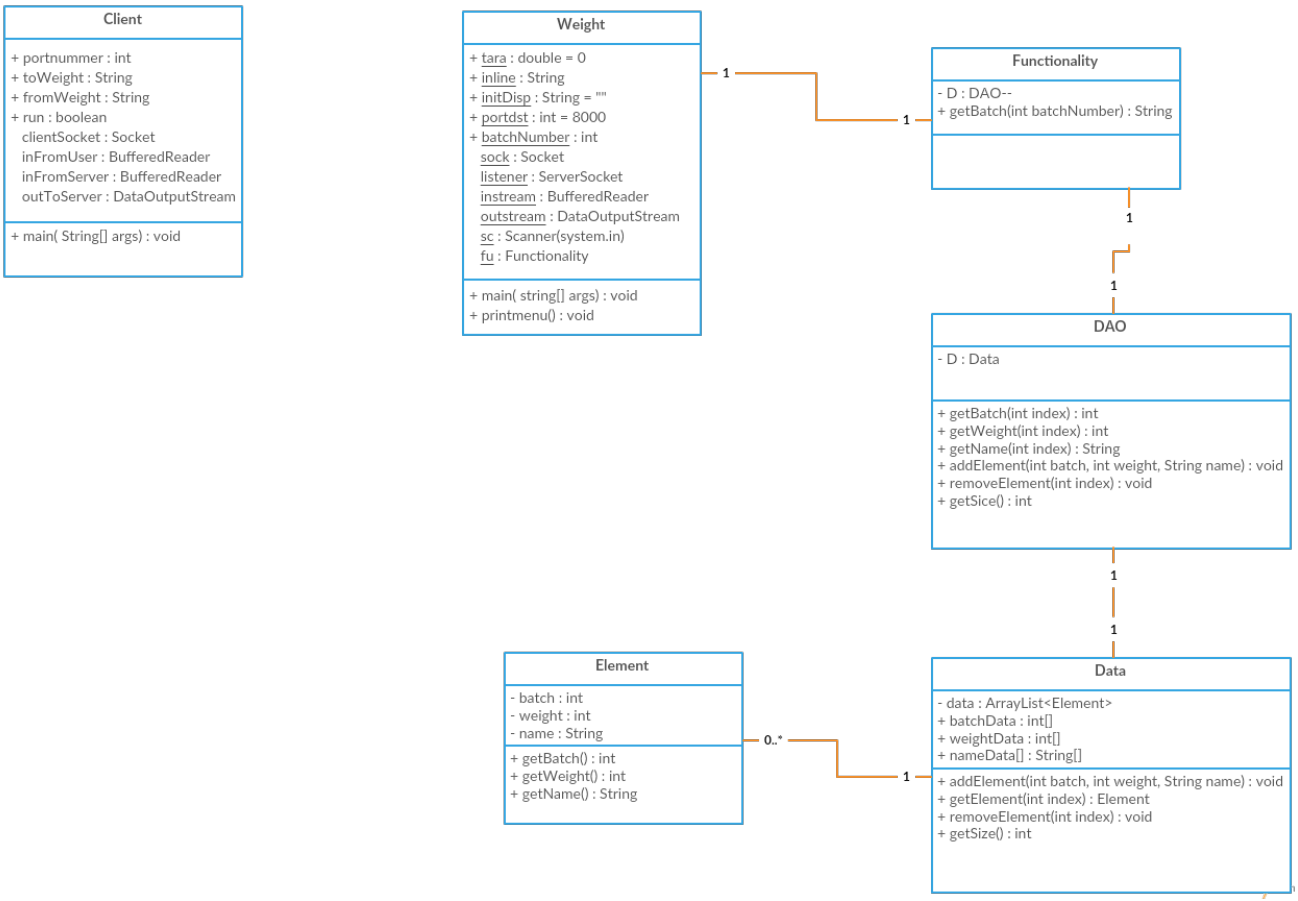
Design

UseCaseDiagram



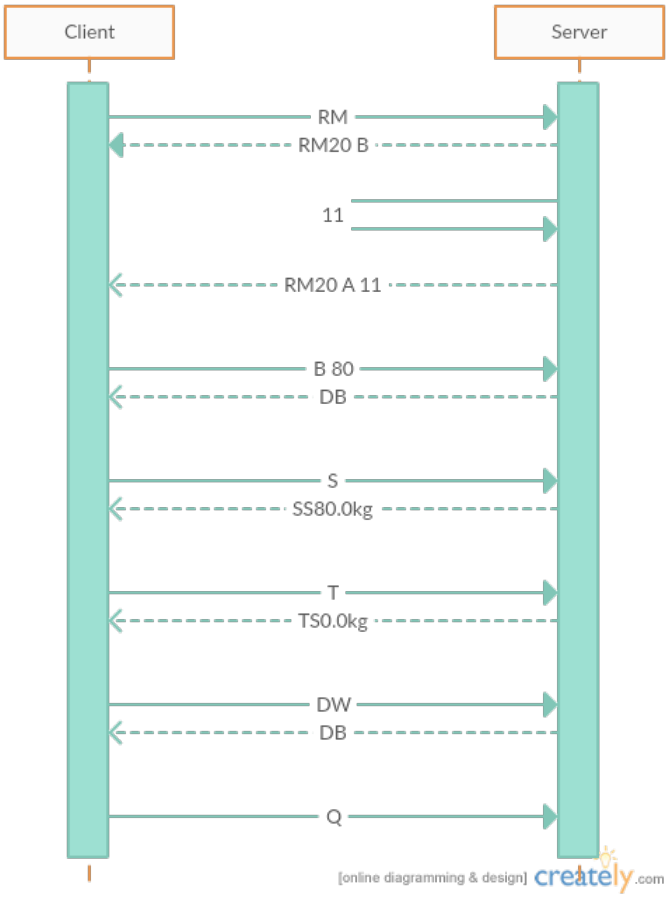
Ovenover ses et Use Case diagram, hvor brugeren er en aktør til alle use cases. Der er en Use Case for alle kommandoerne man kan bruge i klienten, som f.eks. 'D', som benyttes til at skrive en besked på vægtens display.

Klassediagram



Her ses et design-klasse-diagram over klasserne i programmet. Klienten står for sig selv, da den ikke direkte er med i vægtprogrammet, men kun forbinder sig til den. Vægten er sat op så alle funktioner, med undtagelse af "RM"ordren, køres i selve vægt-klassen. RM ordren åbner op for en dialog til en funktionalitets klasse, med formål på at hente data fra et datalager. I dette tilfælder henter funktionaliteten dataen op igennem en DAO klasse, som går direkte ind i datalaget, hvor dataet er lagret. Dataklassen har en innerclass kaldet Element, som definerer de elementer, der skal ligges i datalaget.

Systemsekvensdiagram



Her ses System Sekvens Diagrammet over et forløb i vores program. Diagrammet viser forholdet mellem Client og Server, og hvordan de enkelte beskeder sendes frem og hvad der så sendes tilbage. Det første forløb går fra RM til at RM20 A 11 bliver sendt tilbage, i mellemtiden er der blevet sendt RM20 B tilbage for at godkende, på selve vægten skrives det batchnummer der ønskes at arbejde med, når dette er sket sendes der RM20 A 11, som godkender at der er blevet fundet et nummer 11 i databasen. B 80 simulere at der stilles 80 kg. på vægten, der sendes DB tilbage for at godende. S sendes for at se hvad vægten har på sig af vægt, og der sendes SS80.0kg tilbage for godkendelse. T sendes for at tarere vægten - TS0.0kg er godkendelse for dette. DW fjerner det der står på displayet af vægten - godkendes med DB. Til sidst sendes Q for at lukke forbindelsen mellem Server og Client.

Trelags modellen

I denne CDIO-opgave skulle der bruges trelags modellen. Denne model er i sin simpelhed at der er forskel på grænselaget, funktionaliteten og dataen. Dermed skal funktionaliteten ikke have noget data liggende i sig, op datalaget skal ikke have nogle input / output i programmet. Dette gør at der er en klar separation mellem brugergrænsefladen, funktionaliteten og dataen. Hvilket gør at hvis disse laves på den rette måde, kan man erstatte fx. datalaget med et andet datalag, og dermed gøre at programmet har tilgang til en anden form for data. På samme måde ville man kunne erstatte funktionaliteten, hvis nu der skal regnes med empiriske enheder i stedet for metriske enheder. Til sidst ville man kunne erstatte den tekstuelle brugergrænseflade med en grafisk brugergrænseflade uden at der er brug for hverken at ændre noget i datalaget eller i funktionaliteteslaget.

Funktionalitets Laget

Funktionalitetslaget er det lag der har til opgave at sammensætte information fra vores DAO / Data klasser. Metoden getBatch tager et batchnummer, og så returneres en sammensætning af information der er relevant for det enkelte batchnummer.

Data Laget

Data laget er opbygget af to klasser, navnligt Data - der er den relle klasse til dataen, og DAO - der er vores Data Acces Object. Data klassens opgave er i denne opgave at simulere en database, men er i grunden bare opbygget af en enkelt ArrayList der indeholder et antal objekter af typen Element som bare indeholder private integers og en private String, til henholdsvis Batch nummer, vægt og navn. Disse kan så tilgås i den ArrayList der tidligere er beskrevet. Data klassen har så get metoder til hver af de tre parametre i Element objekterne. DAO klassen fører så disse metoder videre til resten af programmet. Dette gør, at når vi senere skal erstatte vores Data klasse med en reel database så burde dette være rimelig lige til.

Vægten

Hvad gør den?

Vægten til denne opgave skulle simulere en Mettler BBK vægt. Vægten lavet til dette projekt fungerer som en server, hvortil der kan oprettes forbindelse

fra en indkommende forbindelse. Den indkommende forbindelse vil derefter kunne bruge de funktionaliteter, som vægten tilbyder.

Når en forbindelse er blevet oprettet til vægten, vil den på vægten printe "vægtmenuen", som svarer til den fysiske vægt. Programmerings-mæssigt vil denne menu forblive oppe, så længe programmet kører, eller indtil at den får nedlukningskommandoen. Det fungerer ved hjælp af en while-loop, som kun kan afsluttes når den rette kommando modtages. Hver gang en kommando afsluttes i while-loopet bliver vægtmenuen printet igen - denne gang med de nye opdateringer. Dette svarer til at skærmen bliver opdateret på en reel vægt.

Tilgængelige kommandoer

De tilgængelige kommandoer, som vægten kan modtage, var opgivet på forhånd til at være: "RM", "D", "DW", "T", "S", "B"og "Q". Gennem klienten vil de forskellige kommandoer blive udskrevet.

Kommandoen RM er en kommando til at skifte displayet ud fra et batchnummer. Når "RM"modtages fra klienten vil der returneres "RM20 B"til klienten og der vil ikke modtages input fra klienten længere, men i stedet vente på input fra en, som står ved vægten. Her åbnes en scanner, som venter på input fra vægten og som beder dem om at indtaste et batchnummer. En person indtaster nu batchnummeret, hvorefter der søges efter dette batchnummer i et datalag. Et varenavn returneres efterfulgt af en vægt af batchen, som herefter vil komme op på displayet. Vægten vil nu ikke længere lytte på personen ved vægten, men på klienten igen, samt returnere "RM20 A"efterfulgt af inputtet fra personen ved vægten til vægten.

Kommandoen D bruges hvis der skal skrives noget i displayet på vægten. Den læser det første D og herefter hvad der kommer efter mellemrummet, vil blive skrevet i skærmen. Når dette er skrevet i displayet på vægten, vil der returneres "DB"til klienten.

Kommandoen DW resetter hvad der står i displayet og returnerer også "DB"til klienten.

Kommandoen T sætter vægten til 0 kg. Det vil sige hvis der allerede står noget på vægten vil den sætte den til 0. Kommandoen returnerer "TS"samt det antal kg, som blev tareret.

Kommandoen S "vejer"det som står på vægten. Den returnerer "SS"samt den vægt af det (utarerede) objekt, der står på vægten.

Kommandoerne B og Q er to kommandoer, som ikke normalt findes på en vægt, men er nødvendige for at få programmet til at virke (da det er en virtuel vægt). B efterfulgt af en vægt "sætter"et objekt på vægten. Det vil sige at det tal som kommer efter B, vil svare til vægten af det objekt, der stilles på vægten. Q bruges til at lukke vægten.

Skift af port

Det er muligt at ændre portnummeret fra den standard værdi som simulatoren lytter på, hvilket er port 8000. Denne værdi kan ændres til et vilkårligt portnummer med en værdi > 1024, når man opstarter fra kommandolinjen. Dette gøres ved at overskrive default-værdien, og vi har derfor også gjort det muligt, at ændre denne fra klientsiden.

Specifik vejledning:

- Åben kommandoprompten
- cd desktop // Stien til filen, hvis den eksporterede .jar fil ligger på desktop
- java -jar *filnavn*.jar *portnummer* // fx java -jar ServerD2.jar 3000

Dette gøres på serveren. Hvis Serveren er startet kan den samme fremgangsmåde for at skifte port bruges på klienten også.

Klienten

Klienten er lavet for at have en mulighed for at bruge vægten. Efter at den har oprettet forbindelse til vægten, vil der kunne tastes et input, som derefter sendes til vægten. Den bruges ikke til andet end kommunikation mellem klienten og vægten.

I klienten oprettes der forbindelse til serveren ved at lytte til port 8000. Denne værdi kan ændres i klienten, så der kan lyttes til et portnummer som kan have en værdi der er større end 1024. Der refererers eventuelt til kapitlets om vægten, hvor den specifikke kommando står beskrevet. Herefter vil de forskellige muligheder for klienten fremvises.

Konfigurations- og versionsstyring

Udviklings- og produktionsplatformen brugt til at udvikle dette projekt er ens og kan beskrives som en samlet platform. Den inkluderer følgende:

- OS: Windows 10
- Java Version 8 update 77
- Eclipse Version: Mars Release (4.5.1) ...

Produktets udvikling er dokumenteret og versioneret, ved hjælp af GitHub, og kan ses på nedenstående link:
`https://github.com/LukasBek/15_CDIO_D2.git`

Tests

Til at teste benytter vi JUnit tests. Vi har i dette program valgt at teste alle de forskellige kommandoer, som man kan benytte i clienten, inklusiv RM 20 funktionen. Vi har dog ikke testet Q funktionen, som skal lukke programmet ned, da man nemt kan teste det, ved at prøve det i programmet. For at køre testene kræver det, at man kører vægt-simulatoren før man kører testskriptsne. Vi har også valgt at lægge RM 20 tests i sit eget skript, da de var lidt mere vanskelige at lave. Her kræves det nemlig også at man taster et batchnummer i vægten, som i vores test skal være 2, for at gå igennem. Man kan selvfølgelig prøve med et andet nummer, hvor man så laver en negativ test.

Konklusion

Det har lykkedes vores gruppe at udvikle en vægtsimulator, som indeholder de metoder som opgaven stiller. Programmet kan simulere en Mettler BBK 462 vægt, som fungerer sammen med klienten.

Ved tilføjelse af funktionalitet, er det gennem programmet muligt at tarere vægten og ændre brutto vægten. Dette gøres gennem vores Vægt klasse, som vi fik udleveret og som vi har videreudviklet på, sådan at vægten har de "almindelige"funktioner en vægt har, navnligt at kunne veje og tarere.

Vi har opbygget vores program efter 3-lags-modellen, hvor funktionalitetslaget har til ansvar at finde de informationer der tilhører et givet batchnummer. Dette bliver fundet igennem vores Data Access Object (DAO)

som har forbindelse til vores Data klasse, der i dette tilfælde simulerer en database, med en ArrayList der indeholder forskellige "Data Objekter". Efter dataet er blevet fundet af funktionalitetsklassen, bliver dataet pakket ind og sendt videre op til vægten.

Derudover har vi lavet en metode, som gør det muligt at ændre standard port nummeret 8000, som simulatoren skal lytte på, både gennem klient og server.

Bilag

Use Cases

Bruger ændrer portnummer

ID: 01
Kort Beskrivelse: Bruger vil ændre portnummeret
Primær Aktør: Bruger
Sekundære Aktører: ingen
Forudsættelser: Brugeren har oprettet den eksterne .jar fil til at køre programmet
Primært flow: 1. Bruger åbner Kommandoprompten 2. Bruger finder stigen til den eksterne .jar fil 3. Bruger kører .jar filen fra Kommandoprompten og angiver et portnummer over 1024 4. Bruger får svar om at der nu bliver lyttet på det angivede portnummer i stedet for standardporten, 8000
Efterfølge: Programmet fortsætter med at vente på input fra klienten
Alternative Flows: 1. (a) Bruger åbner Kommandoprompten (b) Bruger finder stigen til den eksterne .jar fil (c) Bruger kører .jar filen fra Kommandoprompten og angiver et portnummer under 1024 (d) Bruger indtastede et portnummer under 1024 og får at vide at der i stedet bliver lyttet på standardporten, 8000 2. (a) Bruger åbner Kommandoprompten (b) Bruger finder stigen til den eksterne .jar fil (c) Bruger kører .jar filen fra Kommandoprompten og angiver et portnummer som ikke er en integer og får en fejlmeddelelse (d) Kommandoprompten går tilbage til stigen hvor .jar filen ligger og brugeren kan prøve at kører filen igen med et andet portnummer

Bruger indtaster noget i display

ID: 02

Kort Beskrivelse: Bruger vil vise besked i display

Primær Aktør: Bruger

Sekundære Aktører: ingen

Forudsættelser: Brugeren har oprettet den eksterne .jar fil til at køre programmet

Primært flow:

1. Bruger åbner kommandoprompten
2. Bruger finder stigen til den eksterne .jar fil
3. Bruger kører server filen fra kommandoprompten
4. Bruger åbner ny Kommandoprompt og kører klient filen
5. Bruger Skriver 'D' efterfulgt af en besked i klientens kommandoprompt
6. Beskeden bliver vist i serverens kommandoprompt

Efterfølge: Programmet fortsætter med at vente på input fra klienten

Alternative Flows:

1. (a) Bruger åbner kommandoprompten
   (b) Bruger finder stigen til den eksterne .jar fil
   (c) Bruger kører server filen fra kommandoprompten
   (d) Bruger åbner ny Kommandoprompt og kører klient filen
   (e) Bruger Skriver 'DW' i klientens kommandoprompt
   (f) Displayet på serveren bliver ryddet

Bruger sætter taravægt

ID: 03
Kort Beskrivelse: Bruger vil nulstille vægten
Primær Aktør: Bruger
Sekundære Aktører: ingen
Forudsættelser: Brugeren har oprettet den eksterne .jar fil til at køre programmet
Primært flow: 1. Bruger åbner kommandoprompten 2. Bruger finder stigen til den eksterne .jar fil 3. Bruger kører server filen fra kommandoprompten 4. Bruger åbner ny Kommandoprompt og kører klient filen 5. Bruger Skriver 'T' i klientens kommandoprompt 6. Den målte vægt bliver sat til 0
Efterfølge: Programmet fortsætter med at vente på input fra klienten
Alternative Flows:

Bruger vejer et objekt

ID: 04
Kort Beskrivelse: Bruger vil veje et objekt
Primær Aktør: Bruger
Sekundære Aktører: ingen
Forudsættelser: Brugeren har oprettet den eksterne .jar fil til at køre programmet
Primært flow: 1. Bruger åbner kommandoprompten 2. Bruger finder stigen til den eksterne .jar fil 3. Bruger kører server filen fra kommandoprompten 4. Bruger åbner ny Kommandoprompt og kører klient filen 5. Bruger Skriver 'S' i klientens kommandoprompt 6. Vægten vejer objektet
Efterfølge: Programmet fortsætter med at vente på input fra klienten
Alternative Flows:

Bruger indtaster vægt af objekt

ID: 05
Kort Beskrivelse: Bruger vil indtaste vægt af objekt
Primær Aktør: Bruger
Sekundære Aktører: ingen
Forudsættelser: Brugeren har oprettet den eksterne .jar fil til at køre programmet
Primært flow: 1. Bruger åbner kommandoprompten 2. Bruger finder stigen til den eksterne .jar fil 3. Bruger kører server filen fra kommandoprompten 4. Bruger åbner ny Kommandoprompt og kører klient filen 5. Bruger Skriver 'B' efterfulgt af objektet vægt i klientens kommandoprompt 6. Serveren modtager den skrevede vægt
Efterfølge: Programmet fortsætter med at vente på input fra klienten
Alternative Flows:

Bruger slukker programmet

ID: 06
Kort Beskrivelse: Bruger vil slukke programmet
Primær Aktør: Bruger
Sekundære Aktører: ingen
Forudsættelser: Brugeren har oprettet den eksterne .jar fil til at køre programmet
Primært flow: 1. Bruger åbner kommandoprompten 2. Bruger finder stigen til den eksterne .jar fil 3. Bruger kører server filen fra kommandoprompten 4. Bruger åbner ny Kommandoprompt og kører klient filen 5. Bruger Skriver 'Q' i klientens kommandoprompt 6. Klienten og serveren lukker ned
Efterfølge:
Alternative Flows:

Bruger indtaster batchnummer

ID: 07
Kort Beskrivelse: Bruger vil indtaste et batchnummer
Primær Aktør: Bruger
Sekundære Aktører: ingen
Forudsættelser: Brugeren har oprettet den eksterne .jar fil til at køre programmet
Primært flow: 1. Bruger åbner kommandoprompten 2. Bruger finder stigen til den eksterne .jar fil 3. Bruger kører server filen fra kommandoprompten 4. Bruger åbner ny Kommandoprompt og kører klient filen 5. Bruger Skriver 'RM' i klientens kommandoprompt 6. Serveren beder en person om et batchnummer 7. Bruger indtaster et batchnummer i serverens kommandoprompt 8. Serveren godkender batchnummer
Efterfølge:
Alternative Flows: 1. Bruger åbner kommandoprompten 2. Bruger finder stigen til den eksterne .jar fil 3. Bruger kører server filen fra kommandoprompten 4. Bruger åbner ny Kommandoprompt og kører klient filen 5. Bruger Skriver 'RM' i klientens kommandoprompt 6. Serveren beder en person om et batchnummer 7. Bruger indtaster et batchnummer i serverens kommandoprompt