Forår 2016

02324 – Videregående programmering

Gruppe nr.: 19

Afleveringsfrist: Lørdag den 09/04-2016 kl. 05:00

[PROJEKTOPGAVE – CDIO2]

Silas El-Azm Stryhn
s143599

Ramyar Hassani
s143242

Mikkel Hansen
s143603

Frank Thomsen
s124206

Martin Rødgaard
s143043

Denne rapport er afleveret via Campusnet

Denne rapport indeholder 12 sider inkl. denne side.

Timeregnskab

Timeregnskab *(Hele antal timer)*					
	Silas	**Martin**	**Mikkel**	**Frank**	**Ramyar**
Analyse	**1**	**1**	**1**	**3**	**1**
Implementering	**2**	**1**	**3**	**2**	**1**
Test	**0**	**1**	**1**	**0**	**0**
Dokumentation	**1**	**2**	**2**	**1**	**1**
I alt	**4**	**5**	**7**	**6**	**3**

Indholdsfortegnelse

Resumé

Vores rapport indeholder et analyseafsnit, hvor der kort er beskrevet om hvordan vi har analyseret de forskellige kommandoer, og så har vi lavet en kravspecifikation som sætter kravene fra kunden op i vigtighed. Så har vi også lavet et Use Case diagram, som viser hvordan vores vægt fungere med de forskellige kommandoer og den er beskrevet i use case beskrivelsen. Så har vi i vores implementations del valgt at fokusere på de forskellige kommandoer og beskrevet hvad de gør, og hvilke overvejelser vi ellers havde til at løse opgaven. Vi har i denne rapport ikke lavet nogen test del som sådan, men sat vægtens output på de forskellige kommandoer op i mod vores simulators output, og fået dem til at matche. I sidste ende endte vi ud med en simulator, som fungerer og kan det, som den fysiske vægt også kan af kommandoer.

Indledning

Denne rapport indeholder udviklingen af en vægt simulator. Denne simulator simulerer en Mettler BBK vægt, ved hjælp af et konsol-brugerinterface (Det sorte dos vindue i kommandofortolkeren). For at lave denne simulator har vi fået givet et program skelet og skulle selv implementere de forskellige kommandoer, så de simulere en Mettler BBK vægt. Denne proces har vi løst ved at analysere alle funktioner/kommandoer af på en Mettler BBK vægt, og derefter implementere en kopi af kommandoerne på vores simulator.
Rapportens formål er at give en god forståelse af vores tankegang igennem forløbet, hvilke overvejelser vi har gjort og hvordan vægt simulatoren er implementeret.

Hovedafsnit

Analyse

Vi har analyseret kundens krav til vægt simulatoren ved en kravspecifikation, som fortæller, hvilke dele som skal implementeres og hvad vigtigheden af disse er. Samtidig har vi analyseret de forskellige kommandoer som en Mettler BBK vægt har, ved fysisk at afprøve en sådan vægt. Vi har også lavet et Use Case diagram for at vise kunden hvordan vores simulator virker.

Kravspecifikation

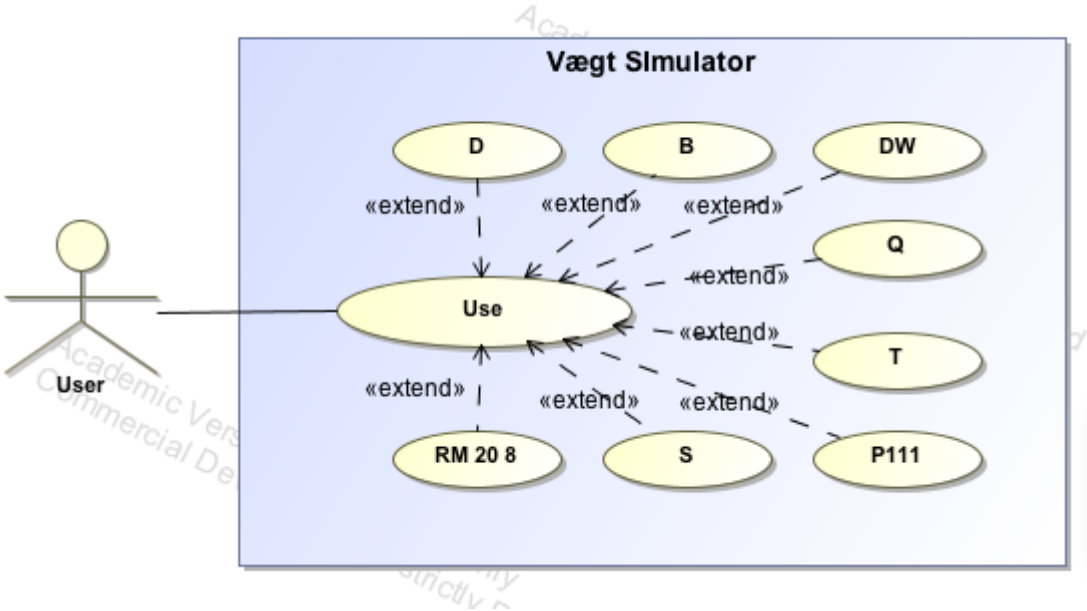
Vi skal lave et program som simulerer en Mettler BBK vægt og har derfor udarbejdet en kravspecifikation ud fra opgaveformuleringen. I denne kravspecifikation inddeler vi først vores krav i funktionelle krav og ikke funktionelle krav for derefter at bruge MoSCoW metoden på dem og kommentere vores valg.

Programmet skal have et konsol-brugerinterface.	Funktionelt	Must have	Vi synes dette krav er vigtigt, fordi man har brug for et sted at se, hvad der sker på vægten.
Simulatoren skal lytte på port 8000. Ved opstart fra kommandolinjen skal denne default-værdi kunne overskrives med et vilkårligt portnummer.	Ikke funktionelt	Could have	Vi synes ikke det er vigtigt at man skal kunne ændre portnummeret.
Programmet skal bestå af 8 kommandoer.	Funktionelt	Must have	Da det er en simulator er det selvfølgelig vigtigt at efterligne den så meget som mulig. Dog er nogle kommandoer, i vores øjne, vigtigere end andre.
En kommando B som skal kunne ændre vægtens brutto-belastning.	Funktionelt	Must have	Da det er en vægt vi skal simulere synes vi dette er en af de vigtigste ting at få med.
En kommando Q som skal kunne	Ikke	Should	Vi synes denne kommando er god at

afslutte simulatoren.	funktionelt	have	have, men den er selvfølgelig ikke nødvendig for at programmet kan fungere.
En kommando T som skal kunne tarere vægten.	Funktionelt	Must have	Vi synes denne kommando er vigtig da vi antager at man gerne vil kunne veje flere ting uden evt. at skulle genstarte programmet.
En kommando D som skal kunne udskrive en besked på vægtens display.	Ikke funktionelt	Should have	Vi synes denne kommando er vigtig men ikke nødvendig for at vægten kan fungere.
En kommando DW som skal kunne rense teksten på vægtens display.	Ikke funktionelt	Should have	Hvis man implementere muligheden for at kunne få vist tekst på displayet. Synes vi selvfølgelig også det er godt med en kommando til at fjerne det igen.
En kommando S som skal kunne udskrive den nuværende bruttovægt og tekst.	Ikke funktionelt	Should have	Denne kommando synes vi er god at have selvom vægten allerede automatisk opdateres efter hver kommando.
En kommando P111 som skal kunne udskrive en besked på vægtens display.	Ikke funktionelt	Should have	Denne kommando har på nuværende tidspunkt samme funktion som D og er derfor lige så vigtig. På den virkelige vægt er der forskel på hvordan teksten bliver vist, men det har vi på nuværende tidspunkt ikke tager højde for.
En kommando RM20 8 som skal kunne sende en besked. Det skal efterfølgende være muligt at give et svar til denne besked.	Ikke funktionelt	Should have	Vi synes ikke at denne funktion er specielt vigtig på nuværende tidspunkt, men at den vil blive det og derfor også er vigtig at få med.

Use Case diagram

Vores Use Case diagram giver et overblik over hvilke Use Cases vores system indeholder. Diagrammet viser os overordnet hvad systemet indeholder og hvilke muligheder aktøren har. I dette tilfælde skal diagrammet illustrerer, hvilke muligheder man har, når man benytter vores vægtsimulator.



Use Case beskrivelse

Use Case Name	Use
Use Case ID	1
Actors	User
Description	User bruger vægten
Preconditions	Vægten er startet
Postconditions	Kommandoer er blevet brugt
Main Flow	1. Vægt startet op 2. Port valg 3. Kommando 3.1. D 3.1.1. Skriver en String på instruktions displayet 3.2. DW 3.2.1. Nulstiller hvad der er blevet skrevet på instruktionsdisplayet 3.3. B 3.3.1. Indtast en vægt 3.4. S 3.4.1. Udskriver det sidst vejede 3.5. T 3.5.1. Tarerer vægten 3.6. P111 3.6.1. Skriver String/Strings i instruktions displayet 3.7. RM20 8 3.7.1. Skriver en besked på instruktionsdisplayet (3 Strings) 3.7.2. Modtager en besked tilbage fra vægten
Alternative Flow	Q - Lukker vægtsimulatoren

Design

Vi har i denne projektopgave ikke noget at tilføje til designafsnittet

Implementering

Se bilag for vejledning til installation og systemkrav, samt installation af den telnet-klient der benyttes til simulatoren.

Vi har i vores implementering fået sat et program skelet til rådighed, som vi har benyttet os af, og så implementeret de resterende dele af koden. Vi har igennem forløbet arbejdet over Github, hvor vi har lavet et repository, så alle fra gruppen kan arbejde på det samme projekt og se hvor langt vi er nået. Fordelen ved et repository er at hvis der sker en fejl, så kan man nemt gå tilbage til et tidligere punkt, hvor fejlen ikke var blevet implementeret endnu.
Vores implementering er hovedsageligt at få alle kommandoer til at fungere og ændre syntaks en smule, så vores simulator bliver en kopi af en Mettler BBK vægt.

Kommandoer

B double: Denne kommando er i stedet for vægtens veje egenskab. Her skriver man B, efterfulgt af det tal som en given genstand vejer. Det vil så komme til at stå som "Netto" i vores simulator. Her svarer vægten "DB"

D "String": Udskriver en string på instruktionsdisplayet. På den fysiske vægt vil denne kommando skrive henover selve vægtdisplayet. Her svarer vægten "DB"

DW: "Nulstiller" hvad der står i instruktionsdisplayet. På den fysiske vægt vil den fjerne den tekst der overskygger selve vægtdisplayet. Her svarer vægten "DB"

S: Udskriver hvad der sidst er blevet "vejet" med B-kommandoen i kommando prompt.

T: Tarerer vægten. Det vil sige, hvis man har noget på vægt og skriver T, vil den vise 0.0kg. Det bruges f.eks. til at afveje et eller andet som man er nødt til at putte i en beholder hvor man så tarerer efter at have stillet beholderen på vægten. Her svarer vægten "T S 0.0 kg".

Q: Lukker programmeret og alle forbindelser ned.

RM 20 8 "String" "String" "String": Den tager imod tre String, som bliver vist på hele vægtens display. Her svarer vægten i første omgang RM20 B og åbner for muligheden så brugeren kan svare på via vægten. Når brugeren har svaret noget vil vægten svare "RM20 A "String"".

P111 "String": Viser en string på instruktionsdisplayet. Hertil svarer vægten "P111 A".

Løsnings overvejelser

Inden vi kom frem til vores løsning havde vi en del andre overvejelser blandt andet at bruge multi threading. Her var vores tanke at have en circular buffer i stil med den vi lavede i en af undervisningsgangene, men i stedet for blot at have en producer og en consumer (altså en der kan lægge ting ind i den og en der kan tage ting ud), ville vi have to til at lægge ting ind. Her ville det altså være muligt både at tage imod input via netværket, samt keyboardet på samme tid. Dog ville det stadig kun være muligt for vores simulator at læse én ting af gangen.
Løsningen vi endte med at vælge var dog blot at udfylde det program skelet der var blevet givet til selve opgaven.

Test

Vi har ikke lavet nogen testscripts i denne omgang, men har i stedet brugt den fysiske vægt til at sammenligne hvordan vores simulator håndterer de forskellige kommandoer i forhold til.

Konklusion

Vi har fået lavet vores vægt simulator, og implementeret de kommandoer, som kunden ønskede, så den simulere en Mettler BBK vægt.

Bilag

Git

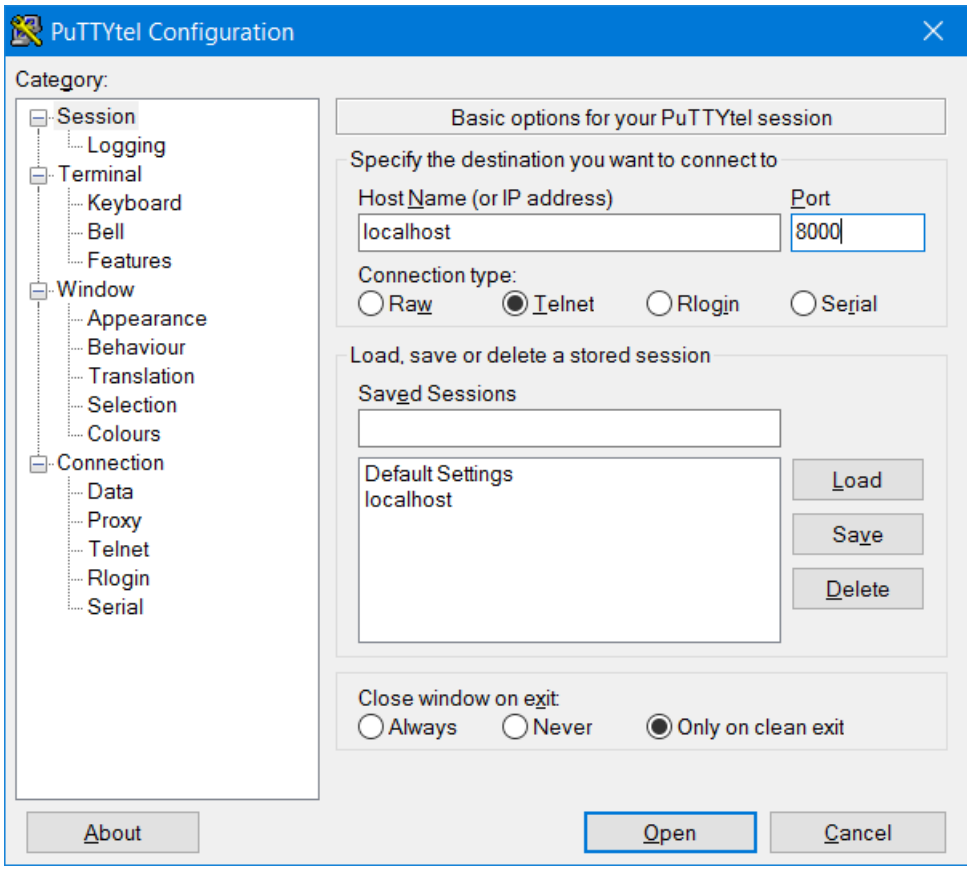
For at få fat i vores projekt, kan man importere det direkte til Eclipse vha. linket https://github.com/Chaaaaap/19_CDIO2 . Dette gøres ved at trykke File → Import → Git → Projects som Git → Clone URI → Indtaster linket i URI-feltet → Next → Finish. Når dette er gjort henter Eclipse selv hele projektet ned. Når først man har hentet projektet ned til Eclipse, skal man blot åbne klassen "main" og køre denne, for at starte spillet.
Man har dog også mulighed for at hente projektet ned som en .ZIP fil, der kan importeres i Eclipse som et eksisterende projekt. Dette gøres ved at trykke File → Import → General → Existing projects into workspace.

Systemkrav

Vores spil kan køres på alle maskiner der som minimum kører med Windows XP. Det eneste systemkrav der er, er at man skal have installeret JAVA 1.8, for at kunne bruge det library som vi har anvendt (jre 1.8.0_60). Man kan dog godt bruge en nyere udgave, så skal man blot ind og ændre i, hvilke libraries der er tilknyttet projektet. Det er ikke noget større problem, men hvis man ikke føler sig sikker på hvordan man tilføjer / fjerner libraries vil vi ikke anbefale det.

Telnet-klient

Når man første gang starter programmet, vil den stå og vente på en connection, som vi benytter en telnet-klient ved navn PuTTY til at give. Denne klient hentes fra hjemmesiden http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, hvor man vælger den version der hedder ”PuTTYtel.exe”. Mens at vægtsimulatoren venter på connection, startes PuTTYtel.exe op, og denne konfigureres til at modtage svar fra vægtsimulatoren. Et vejledende billede nedenfor viser, hvordan det skal se ud. I Host Name indtastes ”localhost” og i Port indtastes ”8000”.



Når denne er korrekt konfigureret, trykkes ”Open” og der vil nu forekomme en kommando promt, som kan tage imod kommandoerne beskrevet i implementeringsafsnittet.