Technical University of Denmark

02324 Videregående Programmering

CDIO delopgave 2




Joakim	Milishia	Mette	Semi
S. Mortensen	Moradi	L.B. Andersen	Seitovski
s175179	s175193	s172840	s175181

March 16, 2018

1 Abstract

The purpose of this assignment, CDIO 1, is to use the three layer model to implement the user-admin function in a scaling system. The implementation will be described throughout this paper with analysis, design and implementation. Several models will be used to give a clear picture of how the problems of this assignment will be handled.

Contents

2 Indledning

Til denne CDIO del 2 skal der udvikles et program, som kan kommunikere med en vægt. Programmet kommunikere over et netværk som styrer afvejningen på en Mettler vægt. De udleverede vægte skal agere som servere, hvor programmet skal kunne hente data fra vægten og dermed også fungere som klient. For at kunne løse opgaven vil der blandt andet blive implementeret en Javaklasse som skal indeholde kommandoerne S, T, D, DW, P111 og RM20 8, så man i stedet for at skulle kommunikere med vægten via et terminalprogram, kan gøre det igennem et java-program.

3 Analyse

Analyse delen er trukket med videre fra cdio 1, da denne her opgave cdio 2 er en udvidelse af systemet der blev implemeteret i cdio 1.

4 Design

I følgende fase varetager vi selve designet af vores system. Her udarbejdes designet til systemet ved hjælp af UML artefakter, såsom Design klassediagram,fFlowchart.

4.1 Designklassediagram

I følgende diagram vises en statisk struktur af systemet som inkludere klasserne og deres attributter, metoder og og relationerne mellem objekterne.

Den arkitektur, som repræsenterer vores Designklasse diagram har vi taget udgangspunkt i 3 lagsmoddellen og MVC modellen (Model, View og Controller modellen)

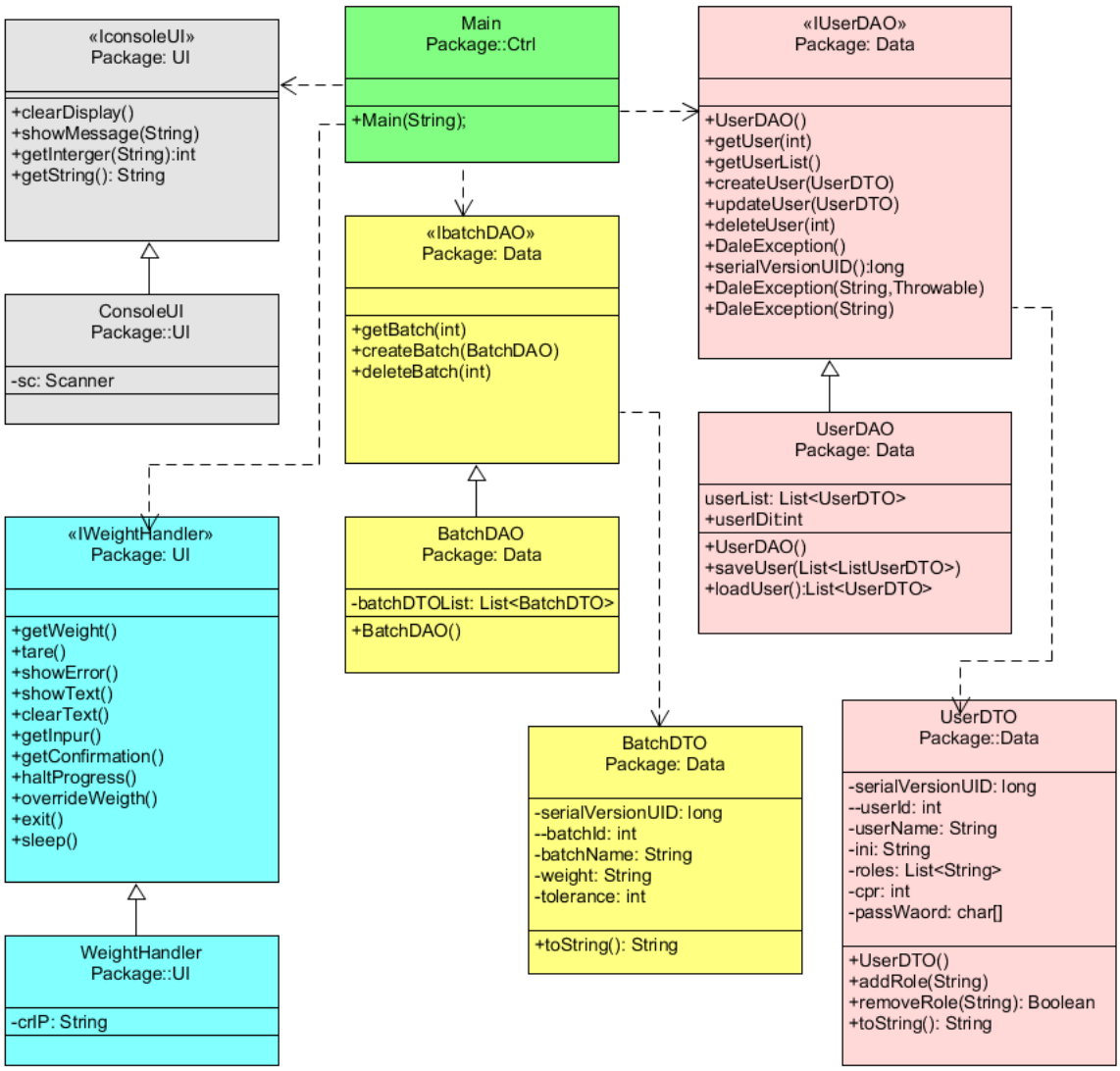


Fig. 1. Design Klasse Diagram

4.2 Designet

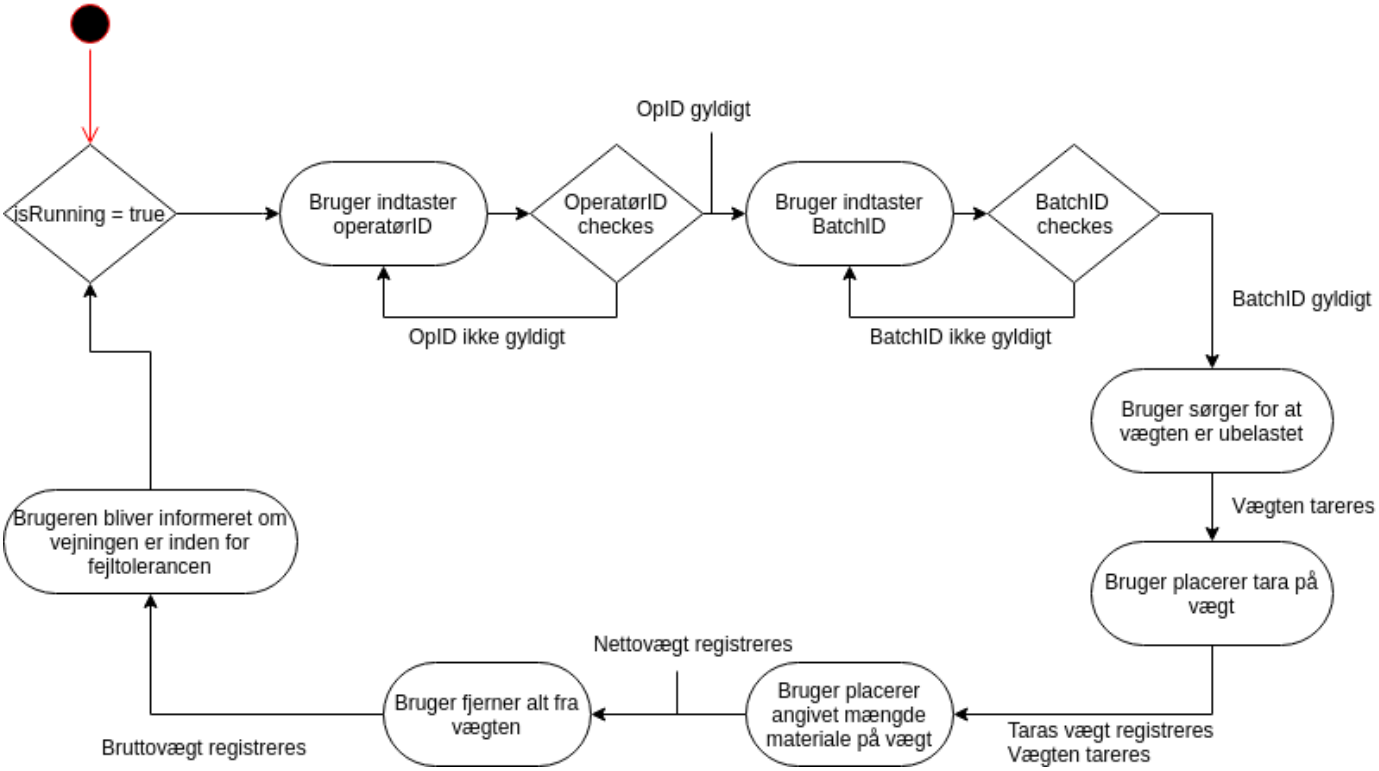
Programmet er designet efter en tre-lagsmodel, hvor programmet er delt op i userinterface, funktionalitet og data. Funktionalitetslaget styrer både interface og datalaget og binder dem sammen uden at de kender til hinanden.

UI-laget skal sørge for at der kan blive vist relevant data i både konsollen og på vægten.

Data-laget skal sørge for at vores data bliver holdt på en forsvarlig og fornuftig måde, hvor vi kan tilgå den nødvendige data så let som muligt.

For at kunne kommunikere fornuftigt med brugeren af vægten skal der implementeres funktionalitet så det er muligt at vise tekst, vente på brugerens input, informere ham om mulige fejl i indtastningen. Desuden skal programmet naturligvis kunne læse belastningen på vægten, og styre ting eksternt som eksempelvis tarering.

Vægten, og i forlængelse vægt-emulatoren har en del funktionalitet som er let implementeret vha. simple kommandoer sendt til et socket. Dette gør implementation af ovenstående funktioner simpel.
Programmet er bygget op omkring vægt-emulatoren der var givet ved opgaven da den fysiske vægt ikke var lettilgængelig. Dette giver naturligvis nogle problemer da emulatoren ikke er 1:1 med den fysiske vægt, men i store træk kan de det samme, så i værste fald ville der skulle skiftes en enkelt klasse ud i UI-laget.
Et flowchart over en typisk kørsel af programmet ville se således ud:



5 Implementering

For at kunne løse opgaven har vi udviklet en helt bestemt klasse nemlig 'WeightHandler'. Denne klasse bliver benyttet til at kunne implementere de forskellige kommandoer og hvordan de forskellige kommandoer skal kunne snakke med vægten via programmet, altså kommunikationen mellem programmet og vægten.

Der sættes fokus på implementationen af metoden getInput. I denne metode holdes der styr på kommandoen "RM20 8". I denne metode tages en streng som bliver formateret korrekt så den bliver sendt afsted i det format som vægten forventer den i. Dette resulterer så i at vægten viser den givne streng på displayet, hvorefter vægten forventer at brugeren

indtaster noget, som så bliver sendt tilbage til programmet, hvor funktionen så piller den modtagne streng fra hinanden og returnerer udelukkende det som brugeren har indtastet.

Da den egentlige funktionalitet af programmet er så relativt simpelt, ligger hele proceduren inde i main-funktionen, hvilket vil sige at Main klassen påtager sig hele funktionalitetslagets rolle.

6 Test

I dette testafsnit vil der blive udført en brugertest for at finde ud af, om vores metoder i WeightHandler-klasse fungerer optimalt.
Metoden showText blev sat til at printe "Velkommen", som tekst på stimulatior. Da programmet blev kørt blev teksten vist på stimulatioren og dermed kan vi konkludere at metoden er vellykket.
Efter at have testet denne metode var det oplagt at teste metoden clearText. Og eftersom at teksten "Velkommen" denne gang ikke blev vist på stimulatoren kan vi konkludere, at denne metode også er vellykket, eftersom den fjerner teksten.

7 Konklusion

Det kan nu konkluderes at der er blevet implementeret og udviklet et program, som kan kommunikere med vægten. Der implementeres klassen IweigtHandler som håndtere kommunikationen fra bruger til vægt. Hvor sisc kommandoerne S, T, D, DW, P111 og RM20 8 er blevet implementeres som metoder i denne klasse. Der er blevet udarbejdet Design-Klassediagram og et flowchart-kort, som har været med til at give et større overblik over implementeringen. Der er gjort brug af Exceptions for at man nemmere har haft mulighed for at behandle fejl i koden.