

Datakommunikation - CDIO 2

Journal af gruppe 16:

- Mathias Larsen
- Emil Eriksen
- Khan Noori
- Jens Nielsen
- Kim Rylund
- Thomas Mortensen
- Christin Holt
- Magnus Vestergaard

Indledning: (Magnus)

Meningen med projektet er at udvikle et program til at styre vægten ved hjælp af en TCP socket. Vi skal konstruere en fastlagt dialog mellem vægten og operatøren.

Regex

Vi har 3 regex mønstre i vores program, den første er denne:

`^RM20 A ([^"]*)$`

^ gør så der ikke står nogle tegn før det første tegn i regexet.

Regex-mønstret skal starte med RM20 A .

Efter det vil regexen gemme alle tegn indtil der dugger et par gåseøjne op eller strengen slutter. Regexen skal ende med disse tegn, hvilket vil sige at den eneste måde redexet ikke matcher ens streng hvis den starter med RM20 A er hvis der er et par gåseøjne senere i strengen.

`^T S +([0-9]*\.[0-9]+) kg$`

Skal starte med sekvensen "T S".

Det efterfølgende mellemrum og + gør at der derefter skal stå minimum et mellemrum.

Efter dette vil regexen gemme hvad der står inde i parantesen. Inde i parantesen skal der stå mellem 0 og uendelige tal, dette kan være efterfulgt af et punktum, og til sidst vil der stå mellem 1 og uendelige tal. Som eksempel så vil 1 og 1.1 blive godtaget, det vil .1 også blive, men 1. vil blive afvist da det ikke følger mønstret med at der skal være mellem 1 og uendelige tal efter det valgfrie punktum.

\$ gør at strengen skal slutte med " kg" (uden gåseøjnene) for at den følger det redex-mønster som vi har lavet.

`^S S +([0-9]*\.[0-9]+) kg$`

Denne regex gør det samme som den forrige, den eneste forskel er at strengen skal starte med S i stedet for T.

Implementation

Programmet er delt op i række steps, hvert step refererer til det efterfølgende samt det foregående, hvis muligt.

Det første step er at bede om operatørnummeret, dette gør vi ved sende en RM20 4 kommando til vægten.

```
writer.writeBytes("RM20 4 \"Operatør nummer:\" \" \" \" \" \"\r\n");
```

Vi forventer at vægten returnerer RM20 B, hvis ikke vises køres step1error() funktionen som informere brugeren omkring fejlen og forsøger at køre steppet igen.

```
if (!reader.readLine().equals("RM20 B")) {  
    step1error();  
    return;  
}
```

Når brugeren har indtastet en værdi, forventer vi at modtage RM20 A #, hvor # er den indtastede værdi. Funktionen RM20() udplukker den indtastede værdi fra svaret, hvis ikke svaret fra vægten er på den korrekte form, returnerer funktionen null og funktionen step1error() køres. Der tjekkes også om svaret er et gyldigt tal.

```
String response = RM20(reader.readLine());  
if (response == null || !response.matches("[0-9]+$")) {  
    step1error();  
    return;  
}
```

step1error() funktionen viser en meddelelse på vægten, venter to sekunder, og prøver derefter step 1 igen.

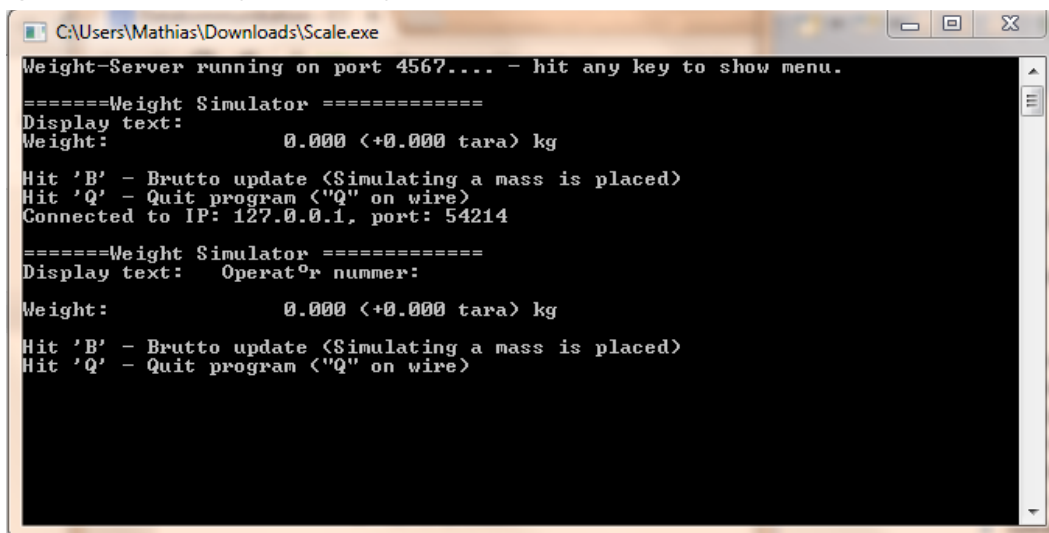
```
writer.writeBytes("D Ukendt operatør.\r\n");  
if (!reader.readLine().equals("D A")) {  
    step1error();  
    return;  
}  
Thread.sleep(2000);
```

```
step1();
```

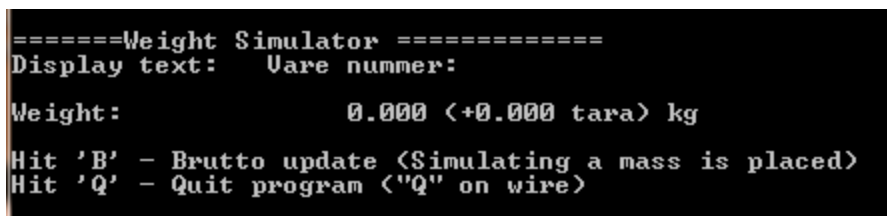
Kørsel af program:

Herunder vil vi give en udførlig beskrivelse af programmets flow. Undervejs i programmet vil det hele tiden være muligt at gå en menu tilbage ved at trykke 0.

Scale.exe åbnes og derefter startes javaprogrammet. Forbindelsen etableres og sessionen er kørende. Brugeren promptes til at starte med for et operatørnummer. Indtast her et gyldigt operatørnummer (1, 2 eller 3)



```
C:\Users\Mathias\Downloads\Scale.exe
Weight-Server running on port 4567.... - hit any key to show menu.
====Weight Simulator====
Display text:
Weight:          0.000 (<+0.000 tara> kg)
Hit 'B' - Brutto update <Simulating a mass is placed>
Hit 'Q' - Quit program <'Q' on wire>
Connected to IP: 127.0.0.1, port: 54214
====Weight Simulator====
Display text:  Operatør nummer:
Weight:          0.000 (<+0.000 tara> kg)
Hit 'B' - Brutto update <Simulating a mass is placed>
Hit 'Q' - Quit program <'Q' on wire>
```



```
====Weight Simulator====
Display text:  Vare nummer:
Weight:          0.000 (<+0.000 tara> kg)
Hit 'B' - Brutto update <Simulating a mass is placed>
Hit 'Q' - Quit program <'Q' on wire>
```

Indtast herefter et af følgende varenumre:

1. Badesalt
2. Hund
3. Kat
4. Halm
5. Ko
6. Gris

7. Amfetamin
8. Rækkehus

Vi vælger at tage vare nr. 6, som er en gris. Derefter vil vægten spørge efter en bekræftelse om det er den korrekte vare. Tast 1 for ja og 0 for nej.

```
=====Weight Simulator =====
Display text:  Korrekt vare?
gris 1/0
Weight:        0.000 <+0.000 tara> kg
Hit 'B' - Brutto update <Simulating a mass is placed>
Hit 'Q' - Quit program <"Q" on wire>
```

Efter en varebekræftelse vil programmet bede om at få sat skålen på vægten.

```
=====Weight Simulator =====
Display text:  Placer skålen på vægten.
1/0
Weight:        0.000 <+0.000 tara> kg
Hit 'B' - Brutto update <Simulating a mass is placed>
Hit 'Q' - Quit program <"Q" on wire>
```

Herefter skal varen placeres i skålen.

```
=====Weight Simulator =====
Display text:  Placer vare i skålen.
1/0
Weight:        0.000 <+0.000 tara> kg
Hit 'B' - Brutto update <Simulating a mass is placed>
Hit 'Q' - Quit program <"Q" on wire>
```

Til sidst er der mulighed for at rydde vægten.

```
=====Weight Simulator =====
Display text:  Ryd vægten.
1/0
Weight:        0.000 <+0.000 tara> kg
Hit 'B' - Brutto update <Simulating a mass is placed>
Hit 'Q' - Quit program <"Q" on wire>
```

Programflowet er hermed slut og proceduren kan herfra gentages hvis det ønskes.

Databasefiler:

Vi har 2 såkaldte flade filer, som henholdsvis er en liste over operatørnumre og deres navne og en liste over produktnumre og navne. Disse filer er i vores tilfælde udelukkende til at slå op i. Der bliver altså ikke ændret i disse filer, som begge er txt. Filer. Man kunne til senere brug forestille sig, at filerne kunne ændres ved administrator menu, så der f.eks kunne tilføjes en ny operatør. I øjeblikket er alle operatørerne hard-coded af os på forhånd. I vores program bruger vi filerne til at slå op, om et operatørnummer findes i tabellen. Hvis det gør, har vi samtidig navnet på operatøren.

Ud over de to filer, har vi også en 3. fil, som agere log fil. I denne bliver der gemt informationer

om hver session. Filen bliver altså ændret hver gang vores program bliver kørt. Dette svarer til, at hver gang en operatør vejer noget på vægten, så bliver alle informationerne om den afvejning gemt.

konklusion:

Vores udviklede program (vægt-simulatoren), der skal styre vægten ved hjælp af TCP socket. programmet kan blandt andet bede operatøren om at identificere sig, og om varenummer og derefter slå varenummer op i databasen og udskrive varenavnet. Vi har simuleret en sekvens mellem operatør og vægt, som er bestemt af os. Vægt-simulatoren udskriver fejlmeddelelserne og den ikke starter forfra ved fejlsituation og dermed kan afvejningsproceduren afbrydes når som helst.