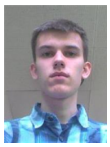
02324

Vidregående programmering

CDIO - Del 1

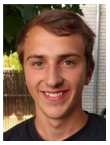
Gruppe 17

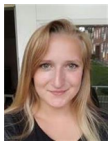
s185118
Aleksander Lægsgaard Jørgensen



s175561
Andreas Østergaard Schliemann

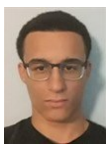
s185096
Jonatan Amtoft Dahl





s164177
Josephine Weirøse

s185103
Søren Hother Rasmussen



s185121
Theodor Peter Guttesen

25. februar 2019

DTU Compute
Institut for Matematik og Computer Science

DTU Diplom
Center for Diplomingeniøruddannelse

Kode og Git: https://github.com/theogutt/17_CDIO1

Indhold

1 Indledning

I dette CDIO projekt har vi skulle fokusere på at du skulle kunne bruge CRUD med en database. Vi har lavet nogle metoder, som hver især står for sin egen del af CRUD, for at kunne oprette data, læse data, opdatere data og slette data.

2 Krav

2.1 Kravliste

ID	Funktionelle krav
Must have	
M01	Systemet skal kunne oprette en bruger
M02	Systemet skal kunne vise en bruger
M03	Systemet skal kunne opdaterer en bruger
M04	Systemet skal kunne slette en bruger
M05	Systemet skal benytte en TUI
Should have	
S01	Systemet skal automatisk generere en adgangskode, der overholder DTUs standarder

3 Analyse

3.1 Use case

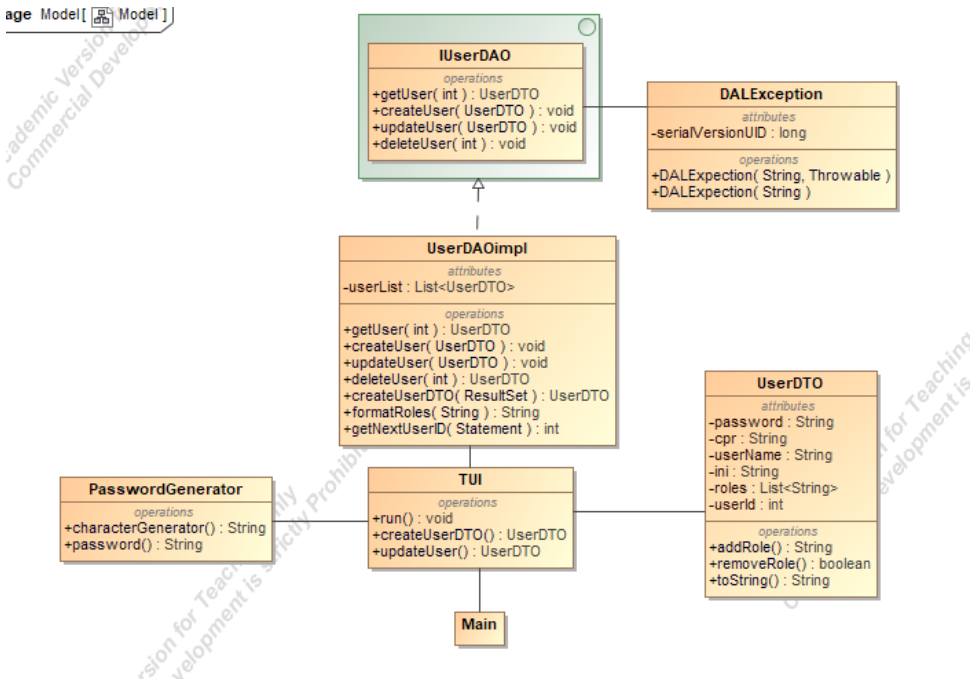
Vi har fået vores fire use-cases fra opgave beskrivelsen, så dette er listen over vores use-cases. Disse fire use-cases er henholdsvis også de fire principper i CRUD, Create, Read, Update og Delete.

1. Opret bruger
2. Vis bruger
3. Opdater bruger
4. Slet bruger

4 Design

4.1 Klassediagram

Vi har lavet et simpelt klassediagram til at give en overskuelig måde at kunne se hvordan, de forskellige dele af programmet taler med hinanden. Vi har indskrevet vores interface samt vores database, dog kunne vi ikke finde nogle officiel syntaks for databaser i et UML klassediagram. Derfor har vi lavet databasen som en klasse men bare kaldt den Database.



Figur 1: Klasse Diagram

5 Implementering

I koden har vi fokuseret på at følge tre lags-modellen, og lave metoder som opfylder CRUD principperne. Måden vi har valgt at implementere tre lags-modellen er ved at have, vores forskellige lag opdelt i forskellige packages. Hele data laget ligger i databasen, her gemmer vi alt information om de forskellige objekter/brugere. Funktionalitet niveauet ligger i vores dal package, og deri er et interface og en klasse som implementere det interface. Denne klasse har alle de metoder vi har som taler med databasen, derfor er dette vores funktionslag, da denne klasse sørger for at sende information rundt, både fra grænsefladen og fra data laget. I vores grænseflade er vores TUI som tager information fra brugeren, og sender det videre ned til funktionalitetslaget.

6 Test

6.1 Positiv test

Vi har testet alle vores *must have* krav, som i dette tilfælde er alle kravene. Programmet bestod alle testene, så vi kan konkludere at vores projekt kan gøre de ting som vi forventede.

Funktionelle krav	**Positiv test**	**Status**
M01	Opret bruger	Bestået
M02	Vis bruger	Bestået
M03	Opdater bruger	Bestået
M04	Slet bruger	Bestået
M05	Tui	Bestået

7 Konklusion

Vi kan konkludere at ved brug af intelliJ kan man nemt få en connection til databaser, og dermed redigere i informationen som står i databasen. Databaser kan derfor nemt bruges til at have sin data, her kan man foreksempel have alle objekter samt informationer om disse objekter. I dette projekt har vi kun læst og redigeret i en tabel fra en database, dog ud fra denne kode ville det være nemt at have flere tabeller samt eventuelt flere databaser.