

CDIO del 1

Group 15

Deadline: 2th of March 2019

This report contains 7 pages excluding front page and appendix



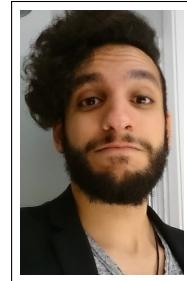
Rasmus Sander Larsen
s185097



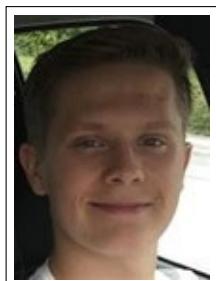
Søren Poulsen
s180905



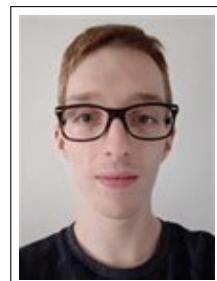
Alfred Röttger Rydahl
s160107



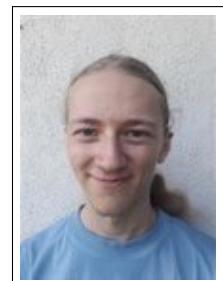
Noah F.M. Hamza
s185084



Nikolaj Tscharn Wassmann
s185082



Anders la Cour Lønborg
s185100



Johannes Verner Kornø Piil
s185114

1 Timeregnskab

Opgave/Navn	Alfred	Søren	Rasmus	Noah	Johannes	Anders	Nikolaj
Design	2	0	1	4	2	0	2
- Navneord	0	0	0	0	0	0	0
- Designklasse	0	0	0	0	0	6	0
- System sekvens	0	0	0	0	2	1	0
- Sekvens	0	0	0	0	0	0	0
Implementering							
General opsætning	0	5	5	1	1	1	5
NoDB opsætning	0	7	3	0	0	0	0
FileDB opsætning	1	0	7	0	0	0	0
SQLDB opsætning	0	0	5	0	3	6	0
TUI	1	5	2	1	0	0	1
Bugfixing	2	0	10	0	1	1	0
Rapport	0	0	0	6	5	8	8
- Kravspecifikation	0	0	0	0	1	0	1
- Analyse	0	0	0	0	4	1	2
- Implementering	0	0	0	0	4	2	1
Total	6	17	33	12	23	26	20
Nettoforbrug	137	timer					

Indhold

1	Timeregnskab	
2	Indledning	1
3	Kravspecifikation	1
3.1	"Must have"krav	1
3.2	"Should have"krav	1
4	Analyse	1
4.1	Use-Case diagram	1
4.2	Brief use case beskrivelser	2
4.3	System sekvensdiagram	2
5	Design	4
5.1	Designklasse diagram	4
6	Implementation	5
6.1	Data package	5
6.1.1	Conn package	5
6.1.2	Dao package	5
6.1.3	Database package	6
6.1.4	File package	6
6.2	Logic package	6
6.3	TUI package	6
7	Dokumentation	6
7.1	Interfaces	6
7.2	SQL	6
8	Konklusion	7
9	Bilag	7
9.1	Github repo	7

2 Indledning

Rapporten er udarbejdet af gruppe 15 til kurset "Videregående programmering - 02324" og "Indledende databaser og database programmering - 02327". Opgaven tager udgangspunkt i CRUD modellen - Create, Read, Update, Delete

Programmet er en TUI - Tekstbaseret UI . Programmet benyttes til administration af brugere og operatører, dog skal programmet ikke tage hensyn til nogle sikkerhedsmæssige foranstaltninger. Programmet indeholder 3 forskellige databaser en hhv. persistent, ikke-persistent og en SQL database. Den persistente databaser sørge for at gemme de brugere, der er blevet oprettet, i en .csv fil. Dermed kan de tilgås hvis man lukker og gemmer data i programmet. Det ikke-persistente datalag gemmer derimod ikke brugerne når programmet lukkes. SQL-databasen gør i bund og grund det samme som den persistente database, bortset fra at databasen tilgås uden for Java, ved hjælp af JDBC.

3 Kravspecifikation

Kravene for systemet følger CRUD (Create, Read, Update, Delete), og er opdelt efter hvor vigtig de er for funktionaliteten af systemet. De fremgår nedenfor, hvor rækkefølgen er sorteret efter funktionaliteten af kravene.

3.1 "Must have"krav

- (M₁) Opret bruger
- (M₂) Vis bruger
- (M₃) Opdater bruger
- (M₄) Slet bruger

3.2 "Should have"krav

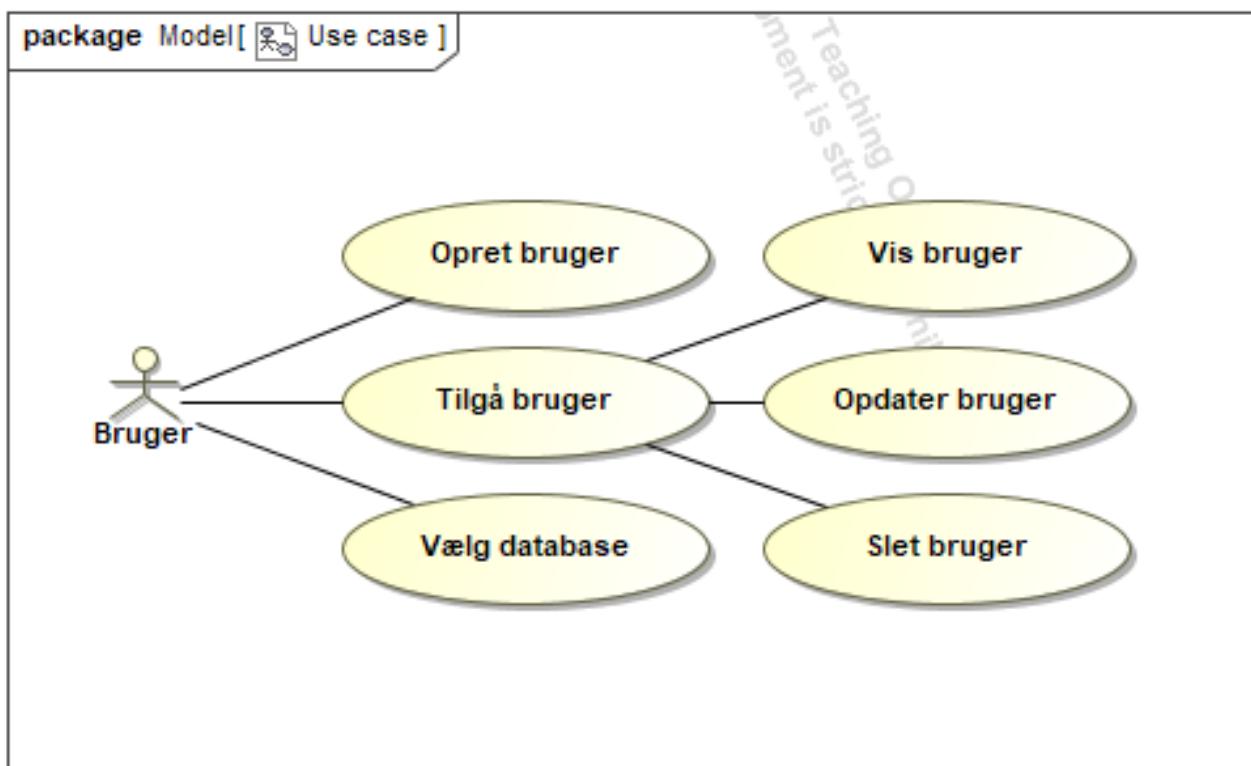
- (S₁) 3 Databaser
- (S₂) Vælg database

4 Analyse

Til analyse af produktet laves der et Use-Case diagram og et System sekvensdiagram, som er med til at illustrere idéen bag softwaren, og hvordan vi har tænkt os at få det til at køre.

4.1 Use-Case diagram

Nedenstående Use-Case diagram viser hvilke operationer, som en bruger kan udføre igennem systemet, og de bliver ydeligere defineret og beskrevet i brief form under diagrammet.



Figur 1: Use Case Diagram med SubUseCases

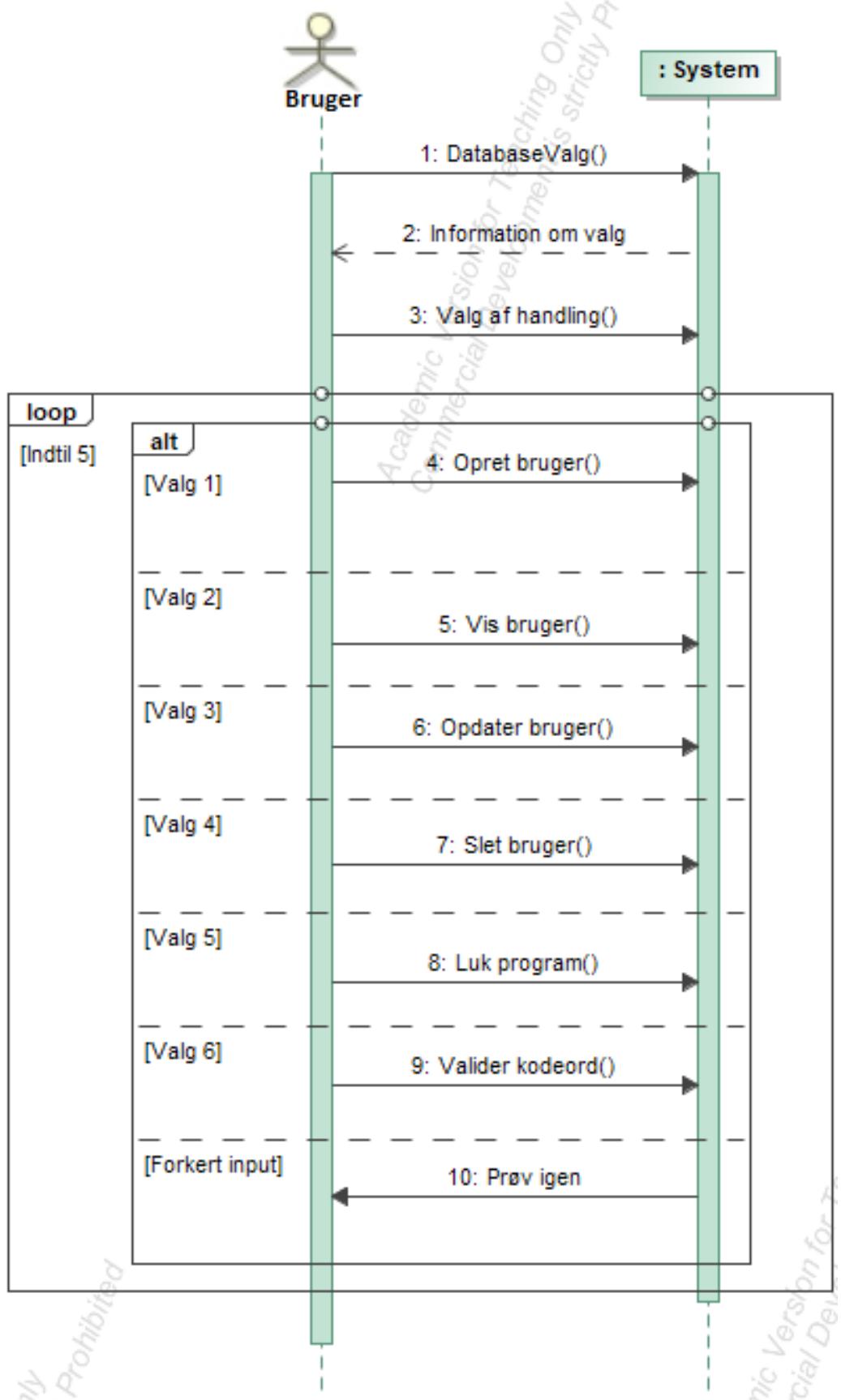
4.2 Brief use case beskrivelser

Use case	Beskrivelse
1: <u>Opret bruger</u>	Man skal kunne oprette en bruger
2: <u>Tilgå bruger</u> - 2.1: Vis bruger - 2.2: Opdater bruger - 2.3: Slet bruger	Man skal kunne tilgå eksisterende brugere Man skal kunne se sin egen og eller alle andre brugere Man skal kunne opdatere attributterne på en bruger Man skal kunne slette en bruger
3: <u>Vælge database</u>	Man skal selv bestemme, hvilken database man bruger

4.3 System sekvensdiagram

Der er blevet udarbejdet et system sekvensdiagram for at give kunden et overordnet overblik over systemets funktionære opbygning. Diagrammet viser hvilke metoder der bliver kaldt, hvornår de bliver kaldt samt hvilke dele af systemet der tages i brug.

Nedenunder system sekvensdiagrammet opstillet, og det viser hvordan brugeren først vælger den ønskede database, hvorefter han kan oprette, se, opdatere eller slette brugere i den givne database. Dette kan brugeren frit blive ved med at gøre, indtil han har fået nok, og vælger at lukke forbindelsen til databasen, hvorefter programmet lukker ned.



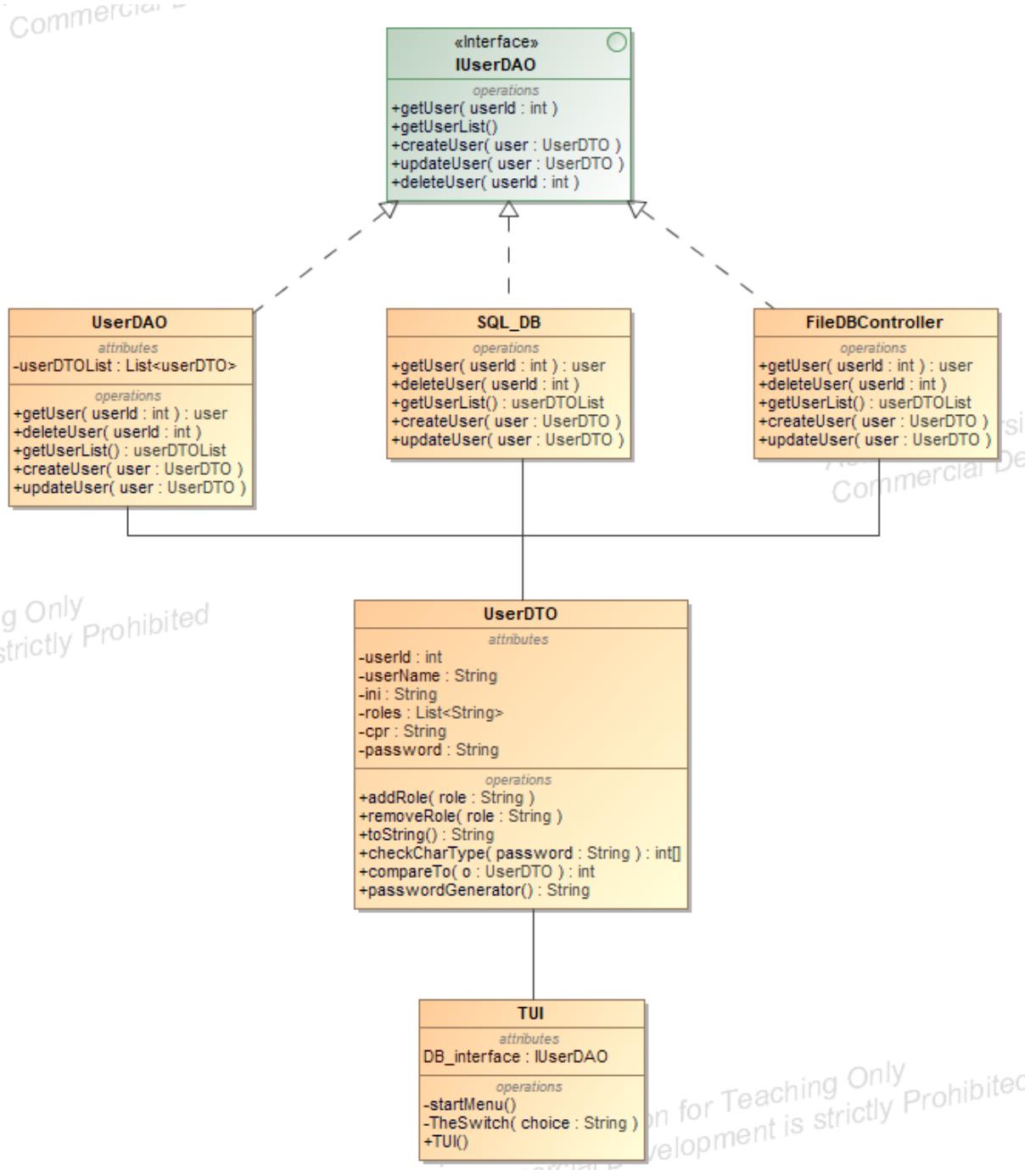
Figur 2: System sekvensdiagram

5 Design

For at opnå et design, der faciliterer en løsning af opgaven på en tidseffektiv og simpel, dog tilfredsstillende, vis, er der blevet udarbejdet et design klasse diagram for vores system, da dette giver os et overblik over hvordan de forskellige objekter skal interagere med hinanden.

5.1 Designklasse diagram

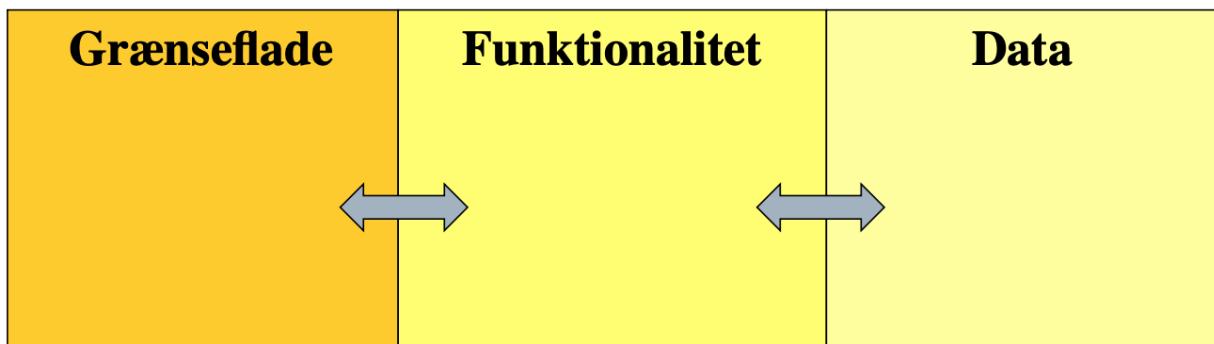
Herunder ses systemets design klassediagram, som viser attributter og operationer for de forskellige klasser samt deres indbyrdes forhold.



Figur 3: Design klassediagram

6 Implementation

Koden er inddelt i 3 hovedpakker; Data, Logik og TUI, som opfylder 3 lags modellen: Grænsefalte, Funktionalitet og Data. Nedenfor ses hvad de forskellige package indeholder samt hvad klasserne specifikt står for at gøre



Figur 4: 3 Lags modellen

6.1 Data package

Data pakken er hvor vi har alt kode, der omhandler vores data, information. Kode der omhandler vores lagring af bruger-objekter findes under pakken dto.

De 3 implementationer af interfacet IUserDAO findes henholdsvis i 3 pakker: "dao" for løsnigen som ikke gemmes, "File" for den som gemmes på en fil og pakken "conn" for SQL løsningen. Derudover er der også en DBController, som bruges til at vælge, hvilken løsning man ønsker at bruge, dvs. hvilken database man vil tilgå.

6.1.1 Conn package

Conn indeholder kun en klasse, der står for oprettelse af "connection" til databasen, hvor de forskellige bruger lagres.

SQL_DB står for alle operationer det har med SQL databasen at gøre. Her bliver funktioner som "getUser", "createUser", "updateUser" osv. brugt til at administrere databasen.

6.1.2 Dao package

Dao package indeholder to centrale klasser for systemet i form af IUserDAO og UserDAO.

IUserDAO er et interface, der danner grundlaget for alle brugere, dvs. IUserDAO indeholder metoder som sørger for at man kan udføre CRUD operationer: "Create", "Read", "Update", "Delete".

UserDAO er en klasse der danner grundlaget for den ikke persistente database, dvs. at alt data gemmes ikke, hvis man tilgår denne database.

6.1.3 Database package

Database package indeholder klassen DBController, som indeholder de metoder, der hjælper TUI'en med at printe information, om hvordan de forskellige databaser kan tilgås samt skaber forbindelsen til databaserne.

6.1.4 File package

File package indeholder 3 klasser der står for at skrive og læse alle informationer, der har med den persistente database at gøre (Ikke SQL databasen). Disse klasser står for at skrive bruger informationer om til en .csv fil og tilgå dem.

6.2 Logic package

I Logic pakken ligger klassen SwitchLogic, som beskriver de metoder, der bliver kaldt af switchen i TUI. Heri ligger bl.a "createUser()", "showUser()", "update()" og "delete()", som alle bliver defineret så de kan bruges i de tre database løsninger. Der bliver også defineret en metode for at tjekke om ens kodeord overholder visse standarder mht. tal, store og små bogstaver, etc., samt en metode for at afslutte programmet.

6.3 TUI package

I TUI pakken defineres den tekstbaseret UI, som projektet bruger, når brugerene skal vælge hvad de vil gøre med databasen, f. eks om de vil oprette, se, opdatere eller slette informationer.

7 Dokumentation

7.1 Interfaces

For at lave et interface erklæres en række metoder som en klasse skal implementere for at blive af typen I. Interfacet er vigtigt at gøre brug af for at opnå lav kobling. Tanken bag er at ens kode kun referer til et interface I uden at kende den konkrete implementering bag. Dette er med til at gøre ens kode mere robust, da det implementerings ændringer i en anden kode ikke påvirker ens egen kode. I et interface er alle erklærede metoder pr. definition abstrakt og public, og alle erklærede felter er public, static og final.

En klasse der implementerer et interface må enten implementere alle metoder i interfacet, eller erklæres som abstract. En klasse kan implementere flere interfaces, men kun arve fra én klasse.

7.2 SQL

SQL - Structured Query Language er et sprog til relationelle databaser. Det er en database som også giver en mulighed for at manipulere og hente data. Det er et deklarativt sprog, hvilket betyder, at

man kan beskrive hvad der skal ske, men ikke hvordan det sker. For at lagre data skal man oprette en forbindelse med en database og dette kan gøres i Java ved hjælp af JDBC driveren. De mange forskellige programmeringssprog har forskellige måder at implementere en SQL database.

8 Konklusion

Der er blevet lavet et program, der kan oprette, opdatere, vise samt slette en bruger, og der er mulighed for at bruge tre forskellige databaser. En der gemmer dataet midlertidigt, en der gemmer dataet i en fil og en der gemmer dataet i en SQL database. Man kan vælge mellem de tre database ved hjælp af en tekst baseret brugergrænseflade (TUI), hvor man også kan benytte de forskellige muligheder for administration af brugere i systemet. Programmet er opbygget efter 3-lagsmodellen, hvor vi har et grænseflade lag, der er der hvor brugeren kan kommunikere med programmet, et funktionalitet lag, som er der hvor der bliver udvekslet informationer mellem grænsefladen og dataet, og til sidst har vi et data lag, hvor alt dataet bliver opbevaret.

9 Bilag

9.1 Github repo

<https://github.com/Team15DTU/S2CDIO1.git>