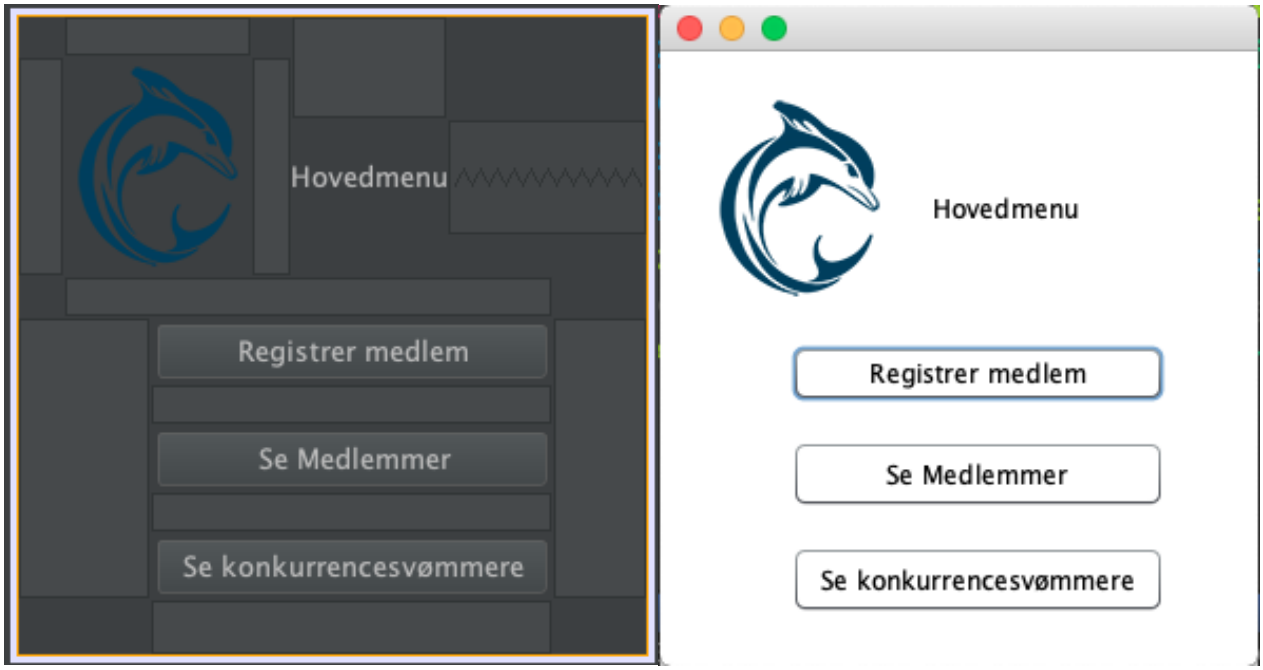
Delfinen

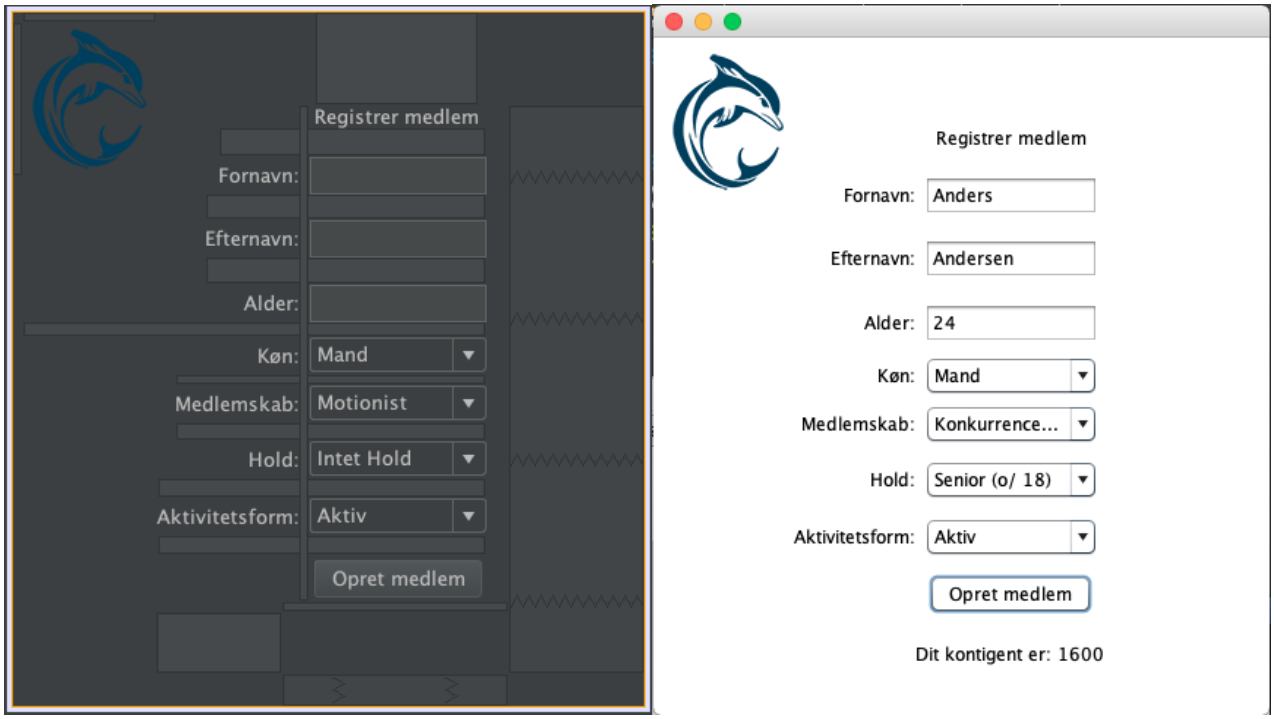
Funktionalitet

Der blevet oprettet en database med tre tabeller, disse navngivet henholdsvis competitive, member og team.
Competitive tabellen viser alle konkurrencesvømmere. Her har hver konkurrencesvømmer også fået tildelt et ID, dette har almindelige medlemmer ikke.
Member tabellen viser alle medlemmer, inklusiv konkurrencesvømmerne.
For team er der ikke nået at blive tilføjet kode, men herunder skulle alle medlemmer, alt efter alder, sættes på enten Junior eller Senior hold.

Programmet starter en Main menu GUI som giver adgang til de forskellige funktioner.

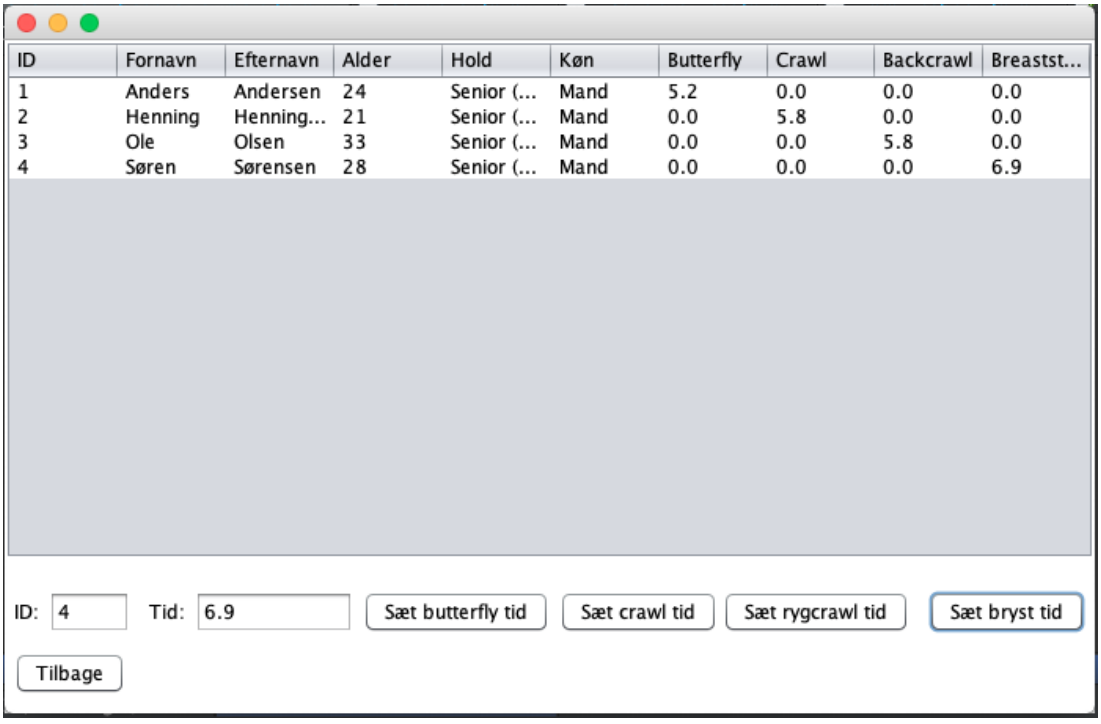


I programmet er der tilføjet nogle forskellige funktioner. Herunder er det muligt at se en liste over alle medlemmer i svømmeklubben, hvilket var en use case. Ydermere use casen “administrer medlemmer” også implimenteret. Herunder kan man tilføje et medlem efter at have udfyldt alle felter.



Vælger man at tilføje et nyt medlem som konkurrencesvømmer bliver medlemmet tilføjet både til member tabellen og competitive tabellen.

Det er også gjort muligt at se alle konkurrencesvømmerne i et jTable. Herunder er der tilføjet fire kolonner - butterfly, crawl, rygcrawl og brystsvømning. Denne GUI giver også adgang til at kunne redigere tider for den enkelte konkurrencesvømmer.



Ikke-funktionelle krav

Genbrug

Programmet er opdelt i hovedsageligt tre mapper, herunder en .data mappe som administrerer forbindelsen mellem programmet og databasen i MySQL. Samtidig er det under denne mappe hvor alle String Queries er skrevet ind under forskellige metoder, disse findes under DataAccessorDB klassen.

Dernest er der en .presentation mappe som indeholder hele programmets Graphic User Interface (GUI). Vi har i alt 4 GUIer som adminstrerer vores Main Menu, Create Member, Show Member og Show Competetive Member.

Robusthed

Der er blevet forsøgt at lave JUNIT tests til vores klasser, dog virker de ikke helt. Vi skal fremtidigt have øvet os mere på at udarbejde bedre JUNIT-tests til vores program hvor at vores metoder er void, og ikke retunere noget.

Der er lavet try catches på alle metoder i DataAccessorDB klassen, og alt der har haft med Databasen eller DataAccessorDB klassen at gøre har fået en try/catch rundt om sig.

I vores GUI er der ikke blevet lavet exceptions for hvad man kan og ikke kan skrive. Skriver man tal ind i navnefeltet (e.g. '123') så kommer der ingen fejl, og 'memberen' bliver oprettet i databasen. Dog kan man ikke skrive bogstaver ind i feltet 'age', da den bruger metoden parseInt(). Dette giver dog kun stackstace, intet bliver printet ud til brugeren.

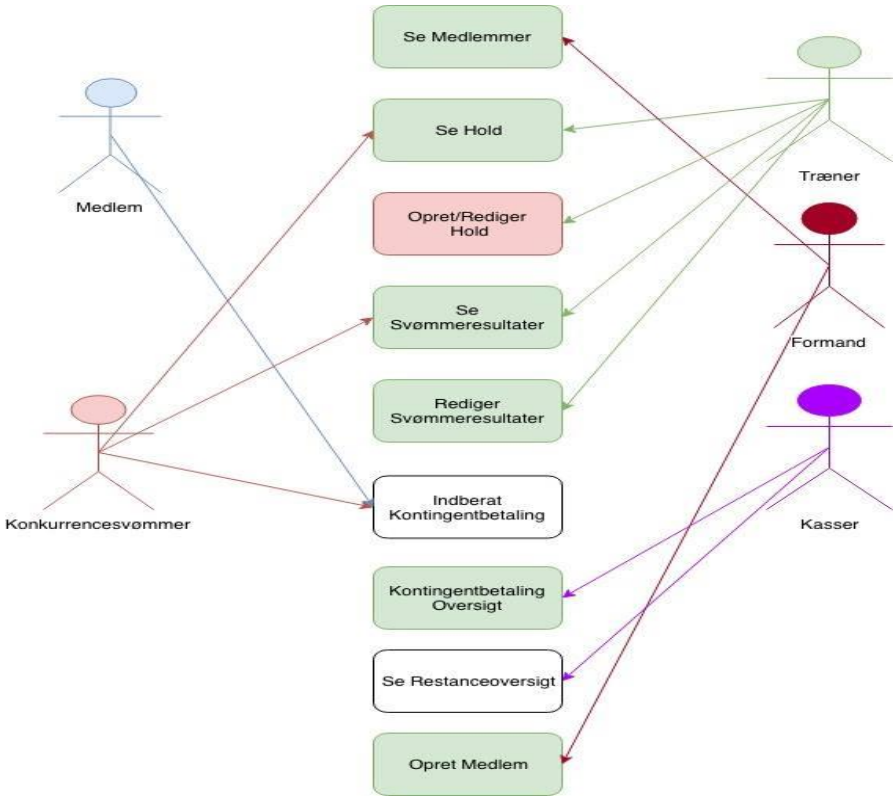
Vedligeholdelsesvenlighed

I forhold til vedligeholdelse og opdeling af vores JAVA program, er det delt op i 3 lag: Data, Logic og Presentation. Dette er gjort så source koden er nemmere at navigere og finde rundt i. I Data

package indgår bl.a. Vores klasser DBConnector (Connection til vores database) og DataAccessorDB (Henter/bringer data fra/til databasen).
I DataAccessorDB indgår metoder som henter fx lister fra databasen samt metoder som opretter en nyt medlem i databasen eller ændrer på en eksisterende medlems værdier. Alle metodernes navne er simple og forklare kort hvad det er metoden gør. Alle instanser er også navngivet simpelt og gør det nemt at gennemskue.

I Logic laget indgår bl.a. Vores Member og CompetitiveMember klasser, de bruges til at oprette objects og til at assigne navne, alder, hold, tider osv.
I fx Member indgår alle de informationer som svømmeklubben kunne være interesseret i at have om deres medlemmer. Her indgår firstname, lastname, age, team (hvilket hold de er på), sex(køn), membership(motionist eller konkurrence svømmer) og activity(passiv/aktiv).
Her indgår Controller klassen som et mellemled, så vi kalder ikke forbindelsen direkte fra databasen, men får en form for "politimand" indover som et ekstra sikkerhedstjek.

I vores Presentations package har vi vores GUI'er (Graphic User Interface) som er hvad brugeren ser dvs. Hvad træneren, formanden og kasseren kan se. GUI'en viser først og fremmest en main menu og derefter kan man klikke sig videre til de andre GUI's hvor man bl.a. kan tilføje medlemmer (den kalder metoder fra controlleren) og vise lister af både members og competitive members samt ændrer i værdierne i databasen (Member værdier, dvs. Navn, alder, aktivitet).



I vores uml diagram til venstre kan man se de klasser vi startede med at have som mål. Vi har arbejdet efter SCRUM metoden, hvor vi har taget én enkelt case af gangen, gjort den helt færdig og så gået videre til den næste. På den måde vil vi altid have noget at vise kunden, selvom vi ikke noget at blive helt færdige til den endelige deadline.

Genbrug

En god portion af det kode vi har lavet kan godt genbruges; hele vores DBConnector og meget fra vores DataAccessorDB kan genbruges ved små navne ændringer.
(VED ÆRLIGT IKKE HELT HVAD DER SKAL SKRIVES HER, I MÅ LIGE KIGGE PÅ DET, ER IKKE SIKKER)