Svømmeklubben Delfinen

KØBENHAVNS ERHVERVSAKADEMI

DAT-16-A

12/12-2016

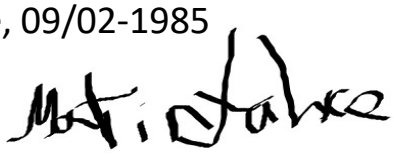
Bertram Baltzer Andersen, 24/08-1993

Underskrift: 

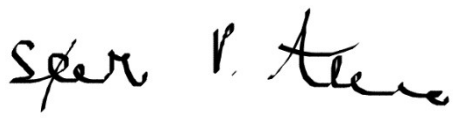
Leivur Hammer, 28/08-1994

Underskrift: 

Martin Fahnøe, 09/02-1985

Underskrift: 

Søren Peter Andersen, 16/08-1996

Underskrift: 

Indholdsfortegnelse

Indledning

Svømmeklubben ”Delfinen” ønsker et IT-system til at håndtere medlemmer i forbindelse med en stigning i medlemskabet. Klubformanden skal have følgende muligheder: Oprettelse af medlemmer, redigering af medlemmer, sletning af medlemmer og oversigt over medlemmer. Oversigt konkurrencemedlemmer, oprettelse af konkurrence / træningstider, oversigt over top-5 konkurrencesvømmere i de forskellige discipliner (butterfly, crawl, rygcrawl, brystsvømning og hundesvømning) og aldersgrupper.

Derudover skal klubbens kasserer kunne se eventuel restance på medlemmer og være i stand til, at håndtere kontigent.

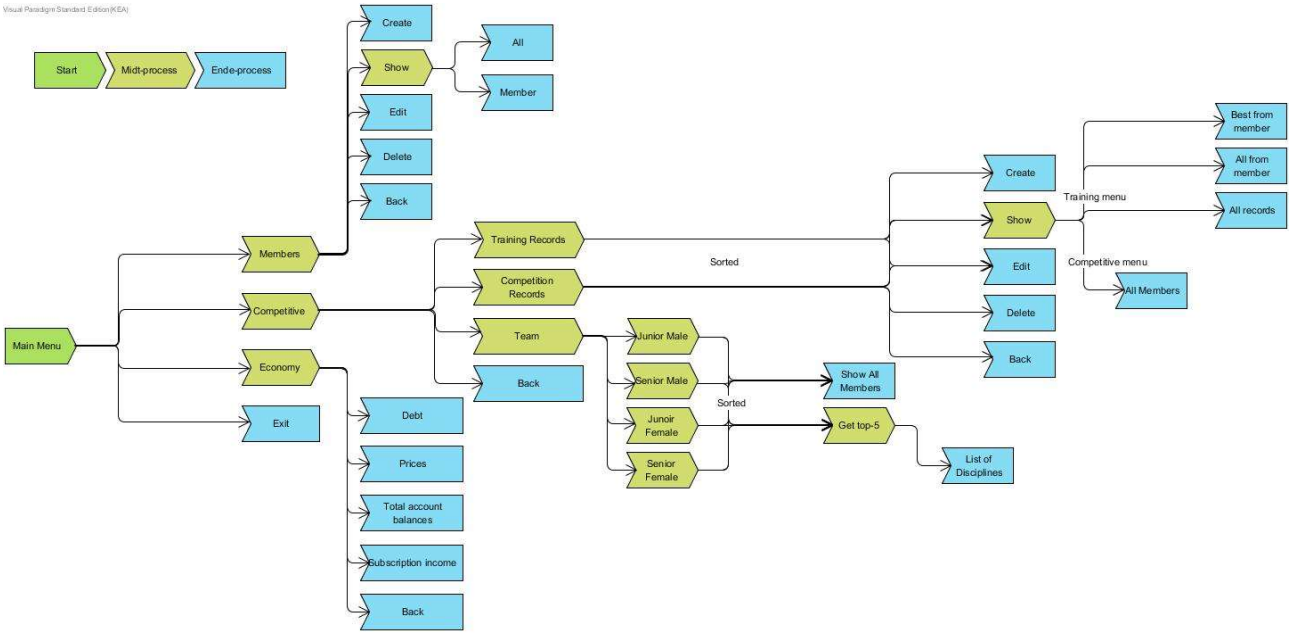
Medlemmerne er opdelt i fire priskategorier: Junior-, senior-, passiv og veteransvømmer.

Afgrænsning & funktionalitet

Martin & Søren

Menuens funktionalitet

Til menuens struktur og funktionalitet er konstrueret et diagram, til belysning af dette:



Dette diagram giver en oversigt over menuens funktionalitet. Trykkes der eksempelvis på ”Members”, bringes brugeren videre til en typisk CRUD (Create, Read, Update, Delete) menu. Herfra kan brugeren endvidere gå ind i ”Show” menuen, som viser to muligheder for, hvordan medlemmer kan printes; de to muligheder er enten alle, eller et specifikt medlem, som kan søges efter via medlemmets CPR nummer.

Afgrænsning

- Klubbens formand.
- Person oplysninger registreres:
  - Klub ID til hver medlem.
  - Navn

    - Personnummer
    - Telefonnummer
    - Adresse
    - CPR.
    - Alder
    - Aktivitetsform
        - Aktivt.
        - Passivt.
        - Junior.
        - Senior.
        - Motionist.
        - Konkurrencesvømmer.

- Klubbens Kasserer.
- Overblik over medlemmer i restance.
- Kontigentbetaling.
    - Kontigent priser:
        - 1000 kr. (under 18 år)
        - 1600 kr. (over 18-60 år)
        - 25% rabat af 1600 kr. for medlemmer over 60 år.
        - For passivt medlemskab er taksten 500 kr. årligt.

- Konkurrencesvømmere.
    - Ungdomshold (Under 18).
    - Seniorhold (Over 18).

- Menu for trænere.
    - Registrering
        - Træningsresultater og dato

- ✦ Stævne resultater
- ✦ Placering og tid
- ✦ Overblik over top-5 bedste resultater i hver disciplin.

- Alle konkurrencesvømmere skal have en træner og svømmedisciplin registreret.
- Svømmernes bedste træningsresultater og dato for løbende registreres under deres svømmedisciplin.
- Svømmere der har deltaget i konkurrencer skal have registreret stævne, placering og tid.
- Træneren bruger træningsresultater til at udtage medlemmer til konkurencer, og ønsker derfor en oversigt der kan vise klubbens top-5 svømmere inden for hver disciplin.
  - Butterfly
  - Crawl
  - Rygcrawl
  - Brystsvømning
  - Hundesvømning.

Vision

Martin

Funktionelle

- Opret en kunde.
- Vælg kundetype.
- Overblik:
  - Konkurrencesvømmere.
  - Almene svømmere.
  - Kunder i restance.
  - Hold.
  - Top-5 svømmere i de forskellige discipliner.

Non-funktionelle

- Brugervenlig konsol-UI.
- Hurtig adgang til information.

Faseplan

Leivur

Faseplanen blev opretteret den 21/11-2016, men er opdateret løbende.

- Aftalekontrakt – Se bilag 1.
- Virksomhedsanalyse
- S.W.O.T. analyse
- Vision
- Første Use-Case
  - Fully dressed
  - Extensions & inclusion
- F.U.R.P.S. analyse
- Glossary
  - Forretningsmæssige begreber
- Afgrænsning
- Domænemodel
- Sekvensdiagram
- Designklassediagram
- Anden Use-Case
  - Fully dressed
- Trejde Use-Case
  - Fully dressed
- //Insert some code here
- //Insert brief Use-Cases here

Virksomhedsbeskrivelse

Søren

Forretningsgrundlag

Delfinen vil gerne have flere kunder / medlemmer. De ønsker derved et attraktivt og nemt system, så det er nemt at blive oprettet og få knyttet personlige trænere (mod betaling) knyttet til det omtalte medlem m.m.

Mission

Grundet stigende medlemmer i svømmehallen "Delfinen", ønsker institutionen et IT system, hvori medlemmer kan oprettes, så der derved holdes styr på hvem der har betalt og ikke har betalt, samt hvilken aldersgruppe og svømmetype medlemmer tilhører. Konkurrencetrænerne ønsker, at kunne bruge systemet til, at få dannet overblik over de bedste konkurrencesvømmere, så de kan få arrangeret stævner m.m.

Vision

Delfinen regner med, at der kommer fortsat flere medlemmer og de skal derfor have et system, som kan holde styr på alle medlemmerne. På baggrund af det nye system regner institutionen med, at implementeringen vil medføre bedre arbejdsvilkår og bedre økonomi.
Konkurrencetrænerne vil gerne kunne benytte dette system som baggrund til stævnearrangementer.

S.W.O.T. Analyse

Martin

Strengths / Stryker	Weaknesses / Svagheder
• Stigende antal af kunder. • Personlige trænere. • Sund økonomi. • Klare mål strategi og planer.	• Mangel på IT system at håndtere stigende antal af kunder. • Mangel på organisering. • Mangel på overblik over medlemmer. • Bedre IT system til give overblik over konkurrencesvømmere og deres tider.
Opportunities / Muligheder	**Threats / trusler**
• Stigende antal af kunder giver større købekraft til nye faciliteter og systemer. • Med nyt IT system kan vi nemt håndtere det stigende antal af kunder.	• Popularitet af svømning falder. • Nye konkurrenter. • Nye IT systemer giver uventede problemer. • Hygiejne.

F.U.R.P.S. Analyse

Søren

- Functionality (Hvad kunden vil have)
  - Tilføj, slet, rediger and filtrér medlemmer.
  - Oversigt og håndtering af betaling (PIBS).
  - Konkurrence overblik og liste af konkurrenter.
  - Programmet skal være nemt og hurtigt at bruge.
  - Tjek for duplikationer ved oprettelse / redigering
- Usability (Hvor effektivt er programmet? Er det acceptabelt? Er dokumentationen i orden?)
  - Programmet er hurtigt, nemt og effektivt.
  - Der er skrevet dokumentation til koden og der er kommentarspor i koden.
- Reliability (Maksimale downtime? Fejl er forudsigelige? Hvordan kan det recovers?)
  - Medlemmer gemmes i en database, så eventuel downtime resulterer ikke i mistet data.
  - Det kan reddes ved, at programmet genstartes og sidste handling inden fejlen rapporteres til fiks.
- Performance (Hvor hurtigt er programmet? Højeste responstid? Proccessor forbrug?)
  - Det er et lille logistik program, så der forventes hurtig responstid og lavt forbrug af processor og RAM, dvs. det er et hurtigt program.
- Supportability (Kan det testes, udbygges, vedligeholdes, installeres, konfigureres eller overvåges?)
  - Grundet programmets opbygning, kan både menuen og moduler nemt og effektivt udskiftes, hvis dette ønskes.
  - Det er endvidere nemt at både tilføje og fjerne moduler eller dele af disse.

Glossary

Martin

- Kontigent,
  - Årlig betaling til klubben.
- Svømme discipliner.
  - Hundesvømning.
  - Rygcrawl.
  - Crawl.
  - Brystsvømning.
  - Bedste tider.
- Junior medlem.
  - Under 18 år.
- Senior medlem.
  - Over 18 år.
- Passivt medlem.
  - Lavere kontigent.
- Klubformand.
  - Administrativt arbejde.
- Træner.
  - Håndterer tider.
    - ✦ Konkurrencetider.
    - ✦ Træningstider.
- Pensionist medlem
  - Rabat på kontigent
- Medlemsoplysninger.
  - Navn.
  - Addresse.
  - CPR-nummer.
  - Telefon.
  - Konkurrence oplysninger.
  - Trænings svømmetider.

- Passivt medlem
- Aktivt medlem.

Use-cases

Leivur & Martin

Lav nyt medlem – brief

En kunde ankommer til svømmeklub delfinen og vil gerne melde sig til som motionist svømmer. Klub formanden spørger om personoplysninger og om hvilken type svømmer kunden ønsker, at melde sig til og bagefter skriver klubformanden informationen ind i Dolphin systemet der opdateres i systemet.

Lav nyt medlem – Fully dressed

Scope:

DMS (Database management system)

Level:

User goal

Primary Actor:

Klubformand.

Stakeholder and interests:

- Klub formand: Vil have at systemet er brugervenligt, hurtigt og stabilt.
- Kunde: Vil have hurtig service med minimalt arbejde.

Preconditions:

Ingen.

Success Guarantee:

Det nye medlems information bliver registreret i systemet, kontigent prisen bliver udregnet efter det CPR

nummer der bliver indtastet.

Main Success Scenario (Basic flow):

1. Kunde ankommer til klub delfinen.
2. Klub formand informerer kunden om kontigent priser.
3. Kunde giver personoplysninger(Fulde navn, CPR, Tlf. nummer, Adresse)
4. Kunde vælger en type svømmer (Konkurrence, Motionist)
5. Klub formand vælger Lav Nyt Medlem funktionen og indtaster oplysninger.
6. System gemmer oplysningerne i databasen.

Extensions: Alternative scenarios.

5.a Kunde giver forkert format af CPR, systemet giver eksempel på korrekt CPR og spørger brugeren om at indtaste CPR på ny.
5.b Klub formand laver fejl ved indtastning, formanden lukker systemet, kunden bliver ikke gemt og formanden begynder på ny.

Special requirements:

Technology and Data Variations List:

Computer til håndtering af data i systemet.

Frequency of Occurence:

Muligvis kontinuert.

Open issues:

- Kan klub formanden håndtere overførsel af penge?

Bedste svømmetider – Brief

Træneren ønsker at se overblik over svømmeklubbens bedste svømmetider inden for en bestemt disciplin og hvem rekorden tilhører. Han tænder for Dolphin systemet og går ind i Svømmetider funktionen, hvorefter han vælger en køn og disciplinen, hvortil Dolphin systemet printer de bedste tider ud i konsolen.

Bedste svømmetider – Fully dressed

Scope:

DMS (Database management system).

Level:

User goal

Primary Actor:

Klubtræner.

Stakeholder and interests:

-Klubtræner: Vil have at systemet er brugervenligt, hurtigt og stabilt. Derudover skal det give et godt overblik over de forskellige konkurrence svømmere.

Preconditions:

Adgang til IT-systemet for at kunne indtaste og aflæse svømmetider.

Success Guarantee:

Træneren kan ud fra medlemslisten trække tider på de 5 bedste mænd/drenge kvinder/piger i de forskellige svømme discipliner

Main Success Scenario (Basic flow):

1. Træneren taster konkurrence svømmere.
2. Derefter trykker han/hun på senior/junior.
3. Vælger køn.
4. Og til sidst svømmedisciplin.
5. Der printes en liste over de 5 bedste med overstående kriterier.
6. Træneren kan derfor nemt udtage de bedste til næste svømmestævne.

Extensions: Alternative scenarios.

1a) Systemet er ikke blevet opdateret, så listen over konkurrence svømmere stemmer ikke overens med systemet.

2a) et medlem er ikke blevet rykket fra junior til senior selvom han/hun er blevet over 18 år.
4a) medlem er rykket til anden svømme disciplin og træneren har ikke fået noteret det i systemet.
5a) svømmetider er ikke blevet noteret korrekt.
6a) systemet skelner ikke mellem træningstider og stævnetider.

Special requirements:

Technology and Data Variations List:

Muligt log-in system for medarbejdere i fremtiden. Derudover skelne mellem trænings tider og stævne tider.

Frequency of Occurence:
Muligvis kontinuert.

Open issues:
Mulighed for at konkurrence svømmere selv kan noterer deres tider.

Overblik over medlemmer i restance – Brief

Kasseren hos Svømmeklub Delfinen vil gerne have et overblik over kunder i restance, han åbner Dolphin systemet, går ind i Medlemmer funktionen, og vælger Vis overblik over kunder i restance. Systemet printer så et overblik over kunder der er i restance og deres personoplysninger ud i konsolen.

Overblik over medlemmer i restance – Fully dressed

Scope:
DMS (Database management system).

Level:
User goal

Primary Actor:
Klub kasserer.

Stakeholder and interests:
-Klub Kasserer: Vil have systemet til give et overblik over kunder i restance, desuden skal systemet være hurtigt og brugervenligt.

Preconditions:
Muligt log-in system til at identificere kasseren.

Success Guarantee:
Det nye medlems information bliver registreret i systemet, kontigent prisen bliver udregnet efter det CPR nummer der bliver indtastet.

Main Success Scenario (Basic flow):
1. Kasseren tænder for systemet.
2. Kasseren vælger "Økonomi."
3. Kasseren vælger "Vis overblik over kunder i restance."

4. Systemet printer overblik over kunder der er i restance, og deres information ud i konsollen.
5. Kasseren lukker systemet.

Extensions: Alternate scenarios.

1. Kasseren kan ikke tænde for systemet.
 1a. Kasseren kontakter system admin.
2. Der er ingen medlemmer i restance.
 2a. System informerer kasseren igennem konsolen at der ikke er kunder i restance.

Special requirements:

Andet system til at håndtere overførsel af penge.

Technology and Data Variations List:

Muligt log-in system for medarbejdere i fremtiden

Frequency of Occurence:

Muligvis kontinuert.

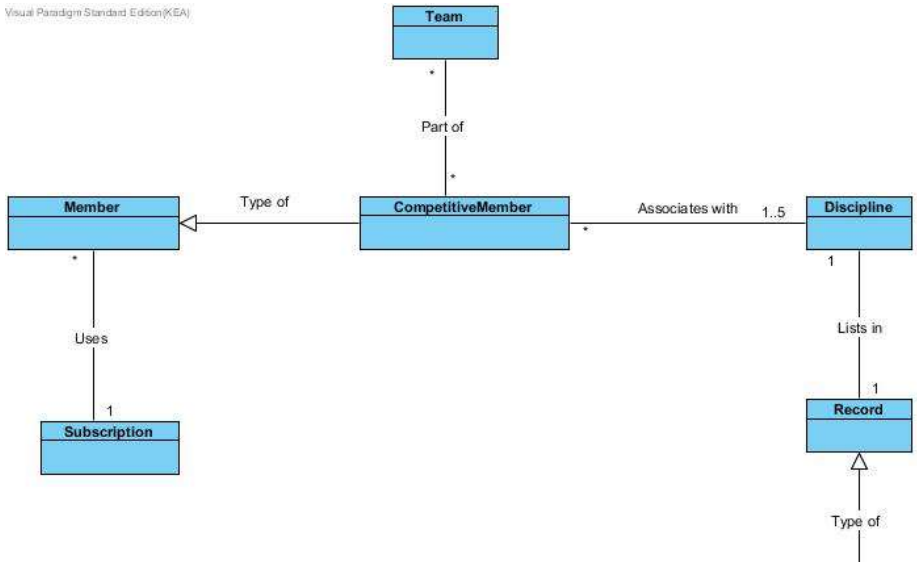
Open issues:

Ingen.

Domain Model

Søren

"Member" er fokuspunktet i denne model. Member er superklassen til "CompetitiveMember". Dette betyder, at både Member og CompetitiveMember begge har en subscription som field / attribut (foruden fornavn, efternavn, adresse m.m.). Specielt for CompetitiveMember, har denne klasse en ArrayList over discipliner, "Discipline", som denne er aktiv i. Derudover har CompetitiveMember også et team som field / attribut. Discipline indgår også i "Records", som er en nedarvning af klassen "CompetitiveRecord". Dette gør det muligt for systemet, at have både træningstider (trænings records) og konkurrencetider (Competitive records).



Class Diagram

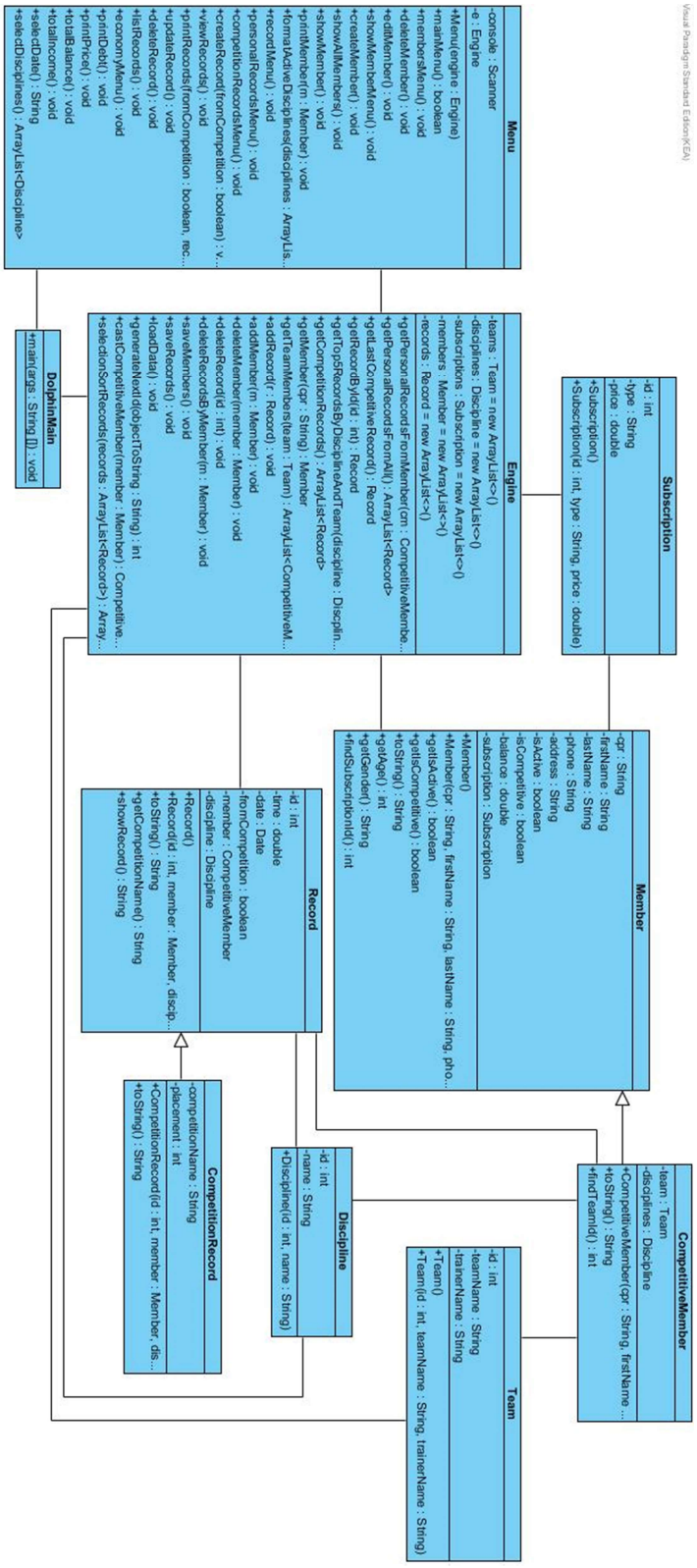
Bertram, Martin & Søren

På billedet ses et klassediagram over systemet til svømmehallen Delfinen.

I diagrammet ses det, at ”Main” ikke har særlig meget kode i sig, men derimod sørger for, at styre de større klasser, som får programmet til at køre som det skal. Det er lavet på denne måde, så det er nemt og smertefrit, hvis man ønsker at skifte nogle af klasserne ud.
Dette betyder i store træk, at Main sørger for, at initialisere vores ”Engine”, som loader alt vores data ind, således det kan benyttes. Derudover initialiserer den også ”Menu”, som er programmets interaktive modul. Denne klasse står for al kommunikation mellem bruger og system. ”Member” klassen er således den tredje store klasse, som får fields / attributter såsom: navn, adresse, telefonnummer og lignende trivielle informationer. Derudover får Member klassen også ”Subscription” og ”Record” fieldet / attributten. Subscription er en klasse, som sørger for, at medlemmerne betaler for den kontigent, som de skal. Dvs. juniorer, seniorer og passive medlemmer skal betale mindre end normalprisen (jævnfør ”Afgrænsning & Funktionalitet”, side 5).

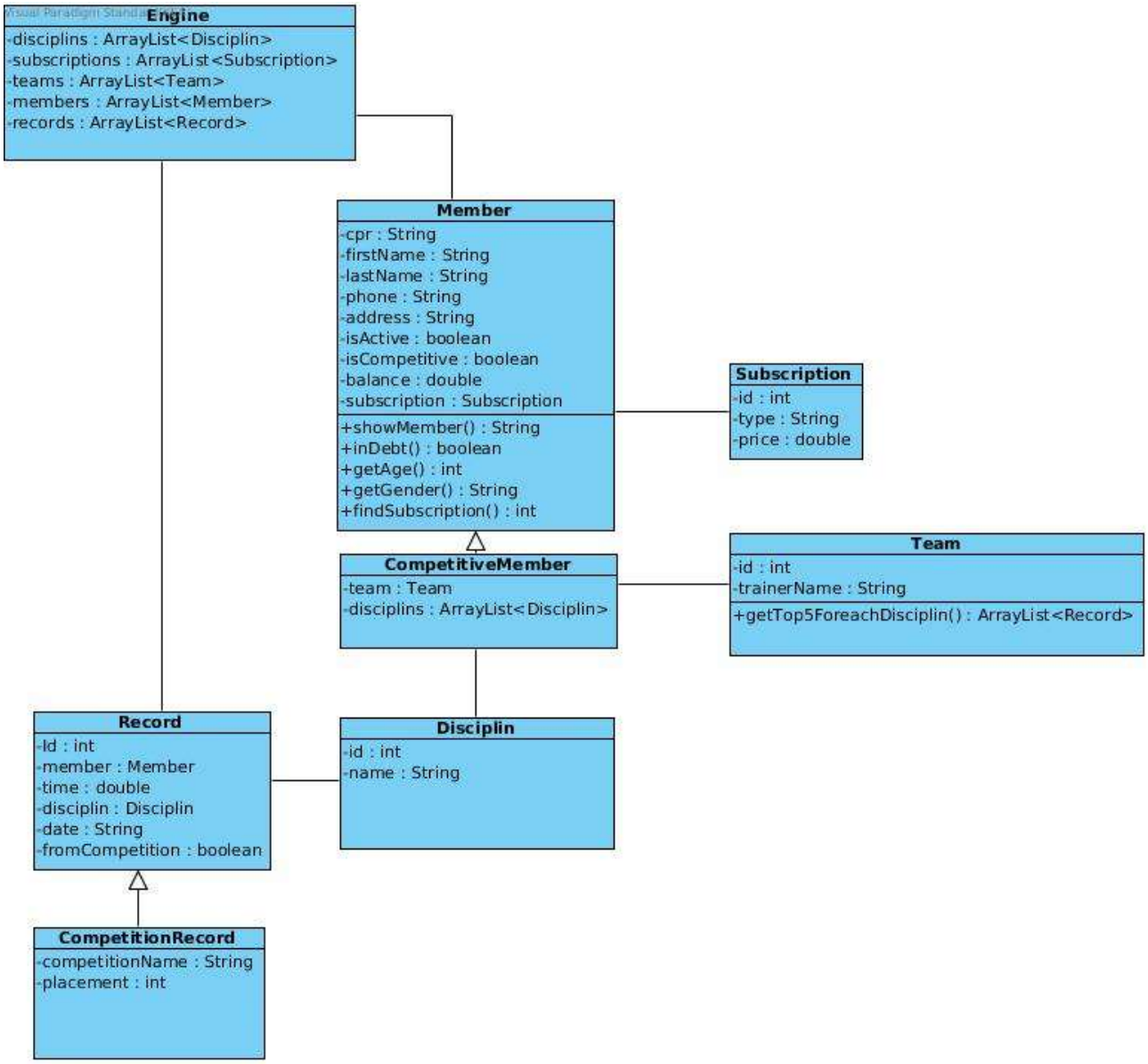
Record klassen sørger for, at man nemt og bekvemt kan lave en Member om til en type CompetitiveMember (eksempelvis hvis et medlem beslutter sig for, at stille op i en disciplin).

”CompetitiveMember” klassen nedarver fra Member klassen og er derfor en subclass. Udover de normale fields / attributter og den normale behaviour, som CompetitiveMember får fra Member, får CompetitiveMember endvidere også følgende klasser som fields: ”Team”, ”Discipline” og ”CompetitiveRecord”. Team sørger for, at inddele de enkelte konkurrencesvømmere i korresponderende hold. Discipline klassen holder styr på hver enkelte medlems aktive discipliner (discipliner det enkelte medlem stiller op til konkurrencer i). Til sidst sørger CompetitiveRecord for, at det enkelte medlems svømmertider er registeret og gemt til senere brug og sammenligning med



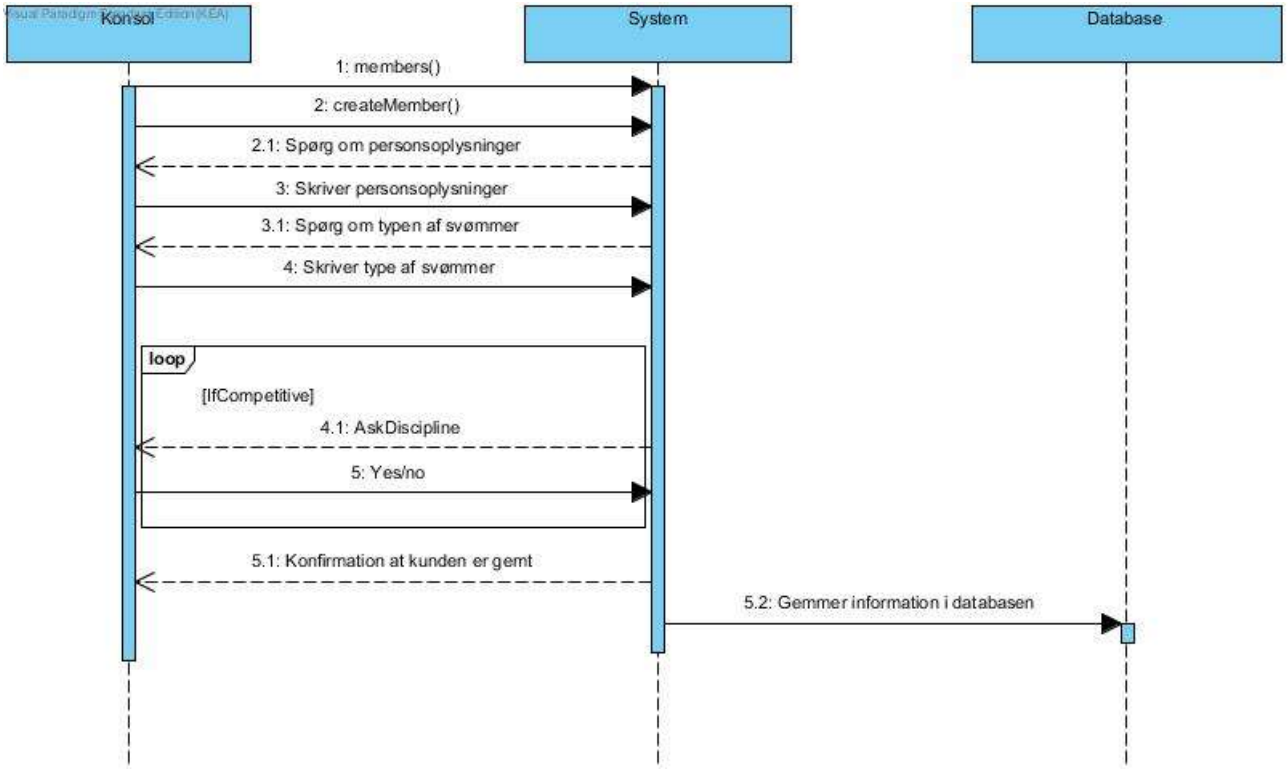
andre medlemmers. Eksempelvis bruges dette til funktionen ”Show top-5” (jævnfør ”Afgrænsning & Funktionalitet, side 4).

Neden for ses første udkast af klassediagrammet. Som det kan ses, er den største ændring tilføjelsen af en ”Menu” klasse for at systemet er mere brugervenligt, fleksibelt og der mulighed for at tilføje moduler eller eventuelt GUI. Derudover er der mange metoder tilføjet, som på forhånd var svært at forudse.



Sequence Diagram

Martin & Søren

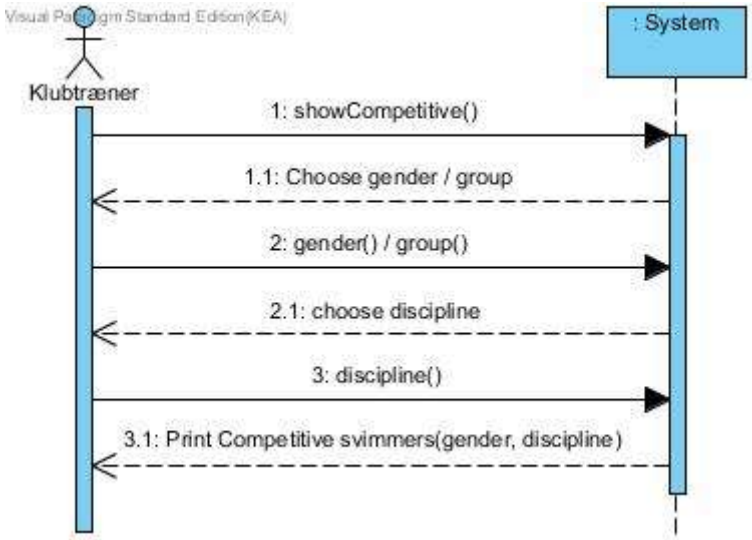


Ovenfor ses et Sequence Diagram, som belyser tilgangen for henholdsvis systemet og brugeren, når der skal oprettes et medlem i klubben. Som det kan ses, vælger brugeren (her vil det som oftest være klubbens formand) menuen ”members”, for derefter at vælge handlingen ”createMember”. Dette får systemet til, at spørge brugeren om personoplysninger, såsom navn, CPR-nummer, telefonnummer og lignende. Dernæst spørger systemet brugeren, hvilken type svømmer den omtalte kunde er – Aktiv, passiv og konkurrence-svømmer. Programmet opretter dernæst kunden med de indtastede oplysninger, giver en konfirmationsbe-sked og gemmer dernæst i databasen.

System Sequence Diagram

Bertram, Leivur, Martin & Søren

Vis Konkurrencesvømmere

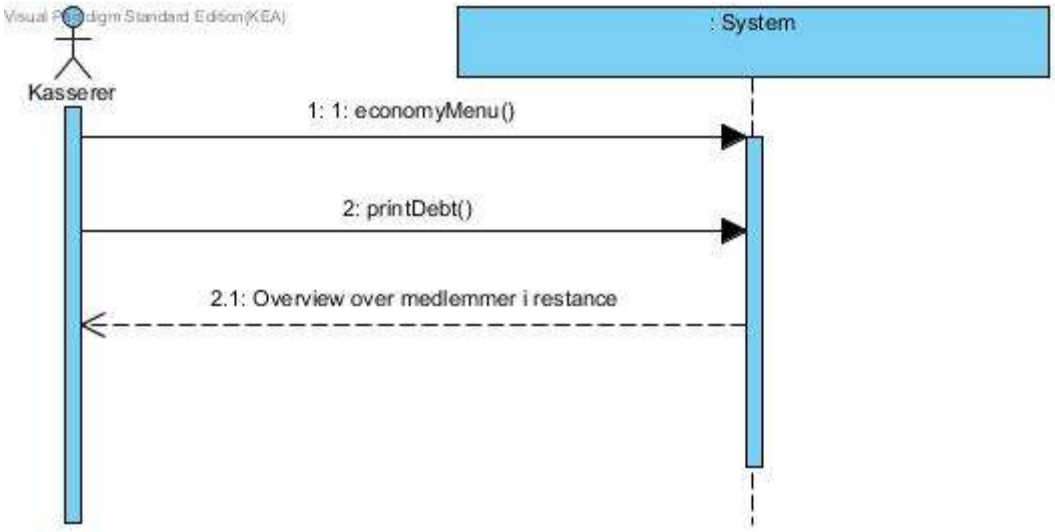


Ligesom med sekvensdiagrammet fra forrige side, viser denne type diagram en bestemt handling. Dog er forskellen for denne type diagram kontra sekvensdiagrammet det, at systemsekvensdiagrammet kun viser det interaktive og operationelle del af programmet og ikke en eventuel lagringsprocess, som det ses i sekvensdiagrammet.

På dette billede ses handlingen, hvor klubtræneren printer en liste ud med de 5 bedste svømmere inden for et given køn eller alderstype, samt en given disciplin. Det ses, at klubtræneren vælger funktionen ”vis konkurrencesvømmere” inde i konkurrence menuen. Dernæst vælger klubtræneren køns- eller aldersgruppen, hvortil systemet viser hvilke discipliner træneren kan printe information ud fra. Sidst printer systemet den specificerede liste ud til træneren.

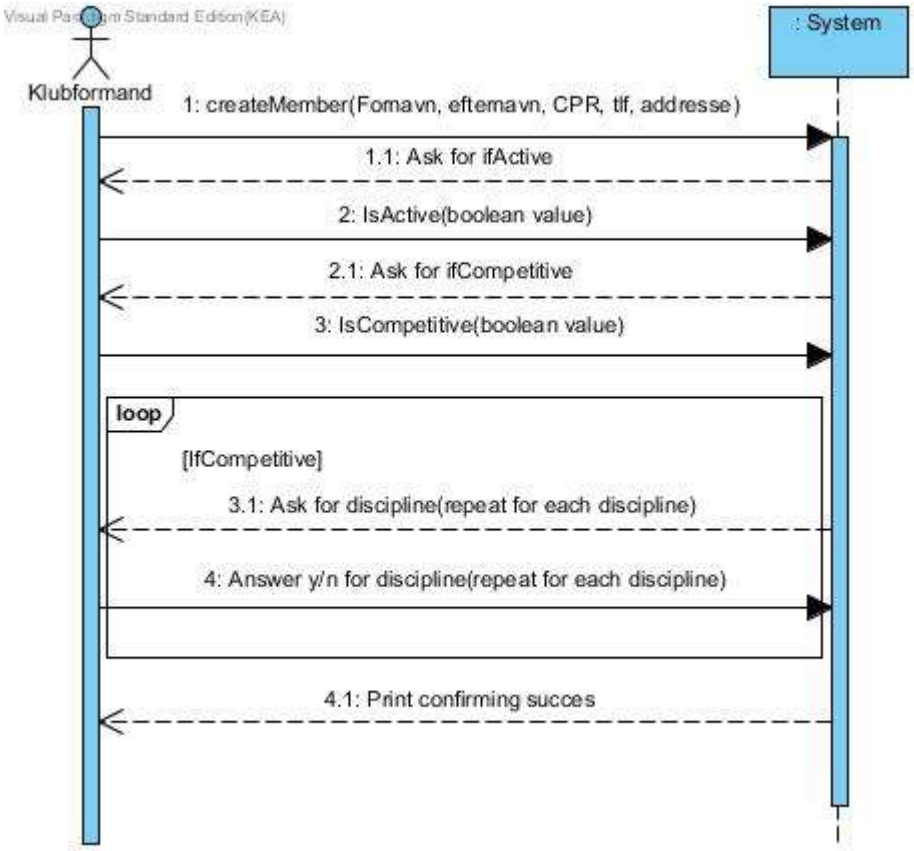
Vis medlemmer i restance

På nedenstående billede kan det ses, hvordan klubbens kasserer kan printe alle medlemmer, som skylder klubben penge ud. Kassereren vælger økonomimenuen og dernæst funktionen ”udprint medlemmer med negativ kontigent”. Systemet printer dertil en liste ud med de medlemmer, hvis kontigent er i negativ balance.



Opret nyt medlem

På nedenstående billede ses handlingen ”opret nyt medlem”. Først og fremmest vælger klubformanden ”opret medlem” i ”medlemmer” menuen. Dernæst indtaster klubformanden vigtige oplysninger, såsom: navn, alder, CPR-nummer og lignende. Dernæst spørger systemet, om det er et aktivt eller passivt medlem og klubformanden svarer tilsvarende. Dernæst spørger systemet brugeren om det er et almindeligt medlem, eller om det er konkurrencesvømmer, som brugeren svarer på. Hvis brugeren indtaster ”ja” til, om det er et konkurrencemedlem der er tale om, går systemet i loop og spørger brugeren om personen stiller op i en given kategori, hvortil brugeren svarer enten ”ja”, eller ”nej”. Dette gør systemet til det har kørt en liste over mulige discipliner igennem. Sidst printer systemet en bekræftelsesbesked ud til brugeren, så ingen forvirring opstår.



Produktets konstruktion

Bertram

Inden vi startede med at lave diagrammerne og koden, var tanken at projektet skulle være så objektorienteret som muligt. Klasser skulle kunne holde referencer til objekter fra andre klasser.

Målet var også at have en klar opdeling af strukturen. Main skulle udelukkende bruges til at starte programmet. Al data, logik og lister skulle initialiseres af Engine. Menu skulle være grænsefladen og stå for al kommunikation med brugeren. Engine og de andre klasser må ikke indeholde spor af at det er en konsolapplikation. Det vil sige System.out.print i Engine er ”fy fy”, og klasserne må heller ikke have metoder til at formatere pænt til konsolen. Ved at holde det helt adskilt ville man kunne skifte menuklassen ud med f.eks. en JavaFX GUI, uden at skulle rydde op i hvad der kunne have blevet spaghetti.

Efter at have designet klassediagrammet, lavede vi alle klasserne med constructors, getters & setters og toString metoderne.

Næste skridt var at lave en loadData() metode i Engine, som kunne lave super og sub klasse objekter ud fra tekstfiler.

For at kunne have mellemrum i vores String fields, og for at kunne skelne mellem super og sub klasser bruger vi delimiters i vores tekst, der splitter hver linje op i et String array.

’:’ indikerer at det er et nyt token og ’#’ at det er tokens, der er unikke til subklasser.

For at gøre det lettere at arbejde med super og subklasser, har vi et boolean field, der fortæller om objektet er super eller sub klasse.

Derfor kan vi f.eks. lave et ”if statement”, der checker om et medlem er af subklassen CompetetitiveMember, mens vi læser en linje i filen ved at parse en bestemt plads i arrayet til boolean og dermed lave en instans af den rigtige klasse.

Nogle af disse objekter skulle også have fields med referencer til andre objekter. Et eksempel på dette er Record, som har et Member field, der indikerer hvem der har sat rekorden.

Når det kommer til at objekter skal indkapsle andre objekter, er det essentielt at de fremmede objekter er blevet initialiseret først.

Det betyder at vi har høj binding, når vi loader dataene fra tekstfilerne. Vi bliver nødt til at loade alt og i den rigtige rækkefølge.

En af de første idéer vi fik var, at i stedet for at havde et ID for hvert medlem, kunne vi bruge et CPR-nummer. Vi kan så bruge CPR-nummeret til at regne ud, hvor gammel en person er – og finde det passende kontigent ved programstart.

Vi har også en liste med konkurrencehold, et for junior og senior, for hvert køn. Ved at tage det sidste ciffer i CPR-nummeret, kan vi se hvilket køn konkurrencemedlemmet er, og sammen med alderen placere dem på det rigtige hold.

Der blev diskuteret indbyrdes, om vi skulle lave et login modul, og om hvordan man skulle fortolke opgavebeskrivelsen med hensyn til, at forskelligt personale skal bruge forskellige funktioner i programmet.

Vi kom til den konklusion, at vi ville springe dette trin over, blandt andet fordi vi hele tiden at skulle tjekke brugerens privilegier, ville blive snørklet, og at fordi vi på første semester kun har lært at gemme oplysninger i plaintext lokalt. Derfor skulle passwords enten hardcodes i kildekoden eller også ville alle med adgang til systemet kunne bruge operativsystemets stifinder til at åbne filerne og aflæse koderne.

Efter at fundamentet var på plads, uddelegerede vi forskellige undermenuer, så alle havde ansvar for at lave en del af programmet. Vi blev enige om at bruge git og github som versionsstyring, så man kunne se de ændringer og opdateringer de andre havde lavet, og kunne splejse vores kode sammen.

Vi aftalte på forhånd, at objekter med persistens, dvs. rekorder og medlemmer skulle have CRUD funktioner.

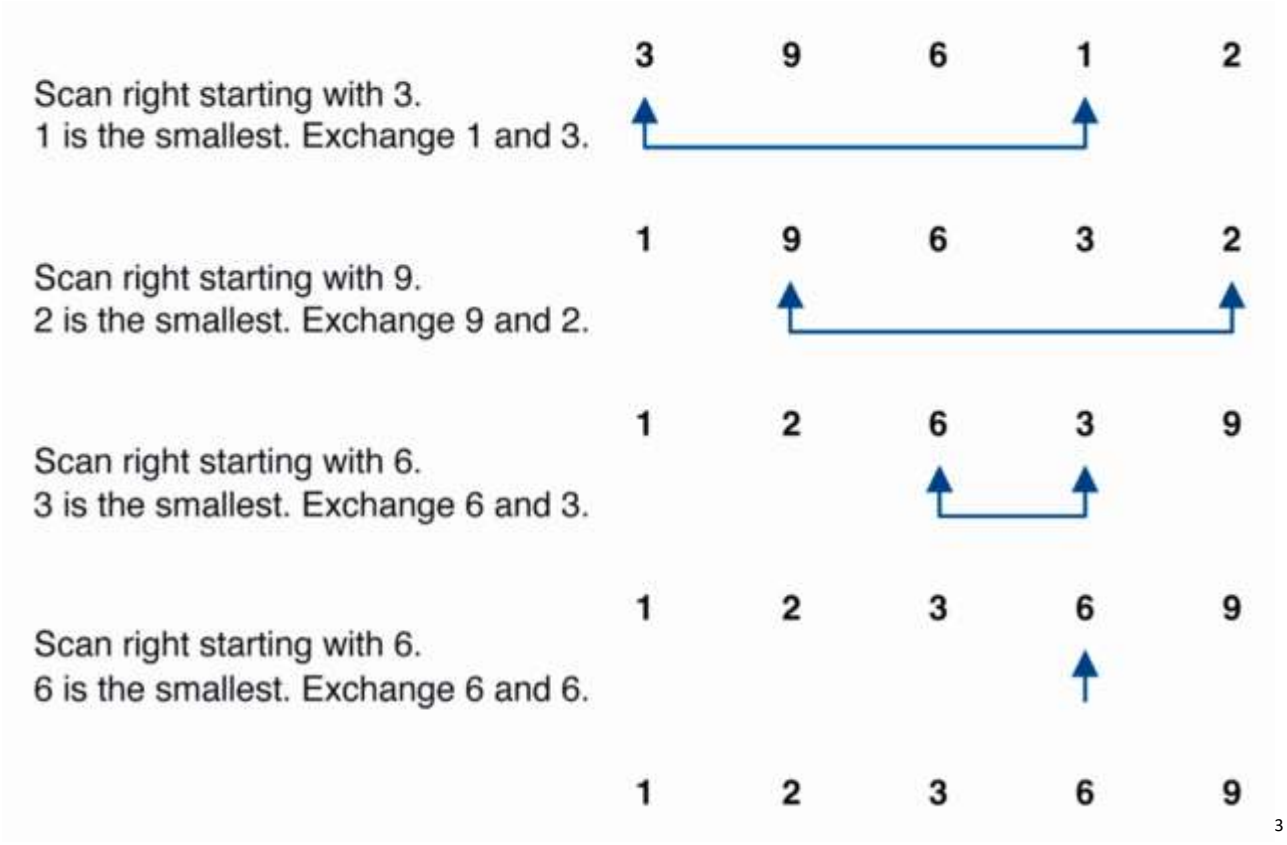
Discipliner, kontigenttyper og hold står beskrevet statisk i opgavebeskrivelsen, så i vores program har vi ikke lavet menu funktioner til at ændre dem.

Rekord funktioner er presset ind i undermenuen ”Competitive Menu” og hvor der er endnu 3 undermenuer, Personal records, Competition records og Teams.

Som beskrevet i opgavebeskrivelsen skulle en træner kunne trække sine bedste 5 svømmere inden for hver disciplin. Det blev en udfordring, fordi vores liste med rekorder ikke er sorteret efter tid, men efter tilføjelse. En normal Collections.sort() kunne ikke bruges, fordi vores liste ikke består af simple datatyper, men af Record objekter. Vi skulle bruge en algoritme til at sortere vores rekorder efter tid. En hurtigt tur på Wikipedia [1, 2] afslørede, at ”Selection sort” vil være en simpel algoritme at kode og implementere. Som vist på billedet, går man listen igennem, og bytter den nuværende værdi rundt med den mindste værdi i listen, der kommer efter sig selv.

[1] https://en.wikipedia.org/wiki/Sorting_algorithm

[2] https://en.wikipedia.org/wiki/Selection_sort



[3]

Efter at listen er sorteret, looper vi den igennem og returnerer de 5 første rekorder, hvor disciplinen er lig med den disciplin brugeren har spurgt efter, og hvor rekorden tilhører et medlem på holdet.

Da vi ikke overskriver eller sletter et medlems rekorder, når medlemmet laver en ny, tjekker vi også om medlemmet allerede er blevet trukket som en af de 5 bedste.

Rekorderne og de ting man skal kunne med dem ifølge opgavebeskrivelsen, virker ret beskedne. Vi diskuterede om vi skulle udbygge systemet, så der også er distancer og bedre integration med turneringer m.m. Men det endte med, at vi besluttede os for at holde os ret tæt på de krav der blev stillet.

[3] http://dopey.cs.vt.edu/courses/cs1706/slides/sorting.html

Konklusion

Ud fra kundens ønsker kan vi konkludere, at de fik, hvad de bad om. Ydermere fik det ekstra funktioner, som de ikke havde specificeret, men som vi mente ville gavne dem og systemet som helhed. Eksempelvis gav vi klubtræneren muligheden for, at dele konkurrencesvømmerne ind i hold efter svømmedisciplin, køn og alder. Derudover gav vi kasseren mulighed for, at tjekke balancen på hvert enkelte medlem, da alle kunder er registreret med en konto, hvori der kan betales. Endvidere er systemet blevet gjort nemmere for medarbejderne, da systemet selv finder ud af, hvor meget de enkelte kunder skal betale. Dette gøres ud fra deres CPR-nummer, hvor systemet trækker aldersgruppen ud og inddeler betalingen således.
Systemet er delt op i moduler, som gør det ekstra fleksibelt og nemt, hvis der skal udskiftes moduler. Eksempelvis, hvis man ville skifte konsol-UI 'en ud med et GUI, kan dette gøres forholdsvis nemt.

Vi lagde hurtigt en arbejdskontrakt, således vi alle vidste, hvornår alle havde mulighed for, at arbejde. Der blev aftalt en generel arbejdsperiode, som lød på 10:00 – 18:00. Dette betød ikke, at man skulle være fysisk til stede i den periode, men derimod at det var den periode man skulle være til rådighed i. Denne periode blev ikke ændret på, med undtagen af et par dage, hvor folk skulle på arbejde. Vi valgte at møde lidt senere, da nogle havde længere til skole end andre, nogle kunne bedre lide at arbejde lidt senere og nogle havde søvnproblemer. Foruden disse små problematikker med tiden, har det generelt været et godt gruppearbejde. Der har på intet tidspunkt været negativ stemning og der har været plads til, at have det sjovt samtidig med, at man skulle arbejde. Inden vi gik fra skolen af aftalte vi, hvad der skulle laves derhjemme og der blev aftalt et medie, hvorpå vi alle kunne mødes i tilfælde af problemer og/eller hjælp (Facebook, Discord, Teamviewer og lignende).
Vi benyttede os af Git og GitHub til versionsstyring, så vi hurtigt og effektivt kunne skubbe opdateringer ud til hinanden.