COPENHAGEN SCHOOL OF
DESIGN AND TECHNOLOGY

Svømmeklubben Delfinen

Datamatikeruddannelsen
DAT17C

1. semester rapport

08-12-2017

Gruppemedlemmer:

Casper Frost Andersen	14-02-90
Martin Løseth Jensen	12-04-97
Christian Strunge	18-11-92
Rasmus Sadurskis	23-12-87

INDHOLD

INDLEDNING

Af Casper Frost Andersen

Valhalla IT-solutions er blevet kontaktet af svømmeklubben Delfinen, der er beliggende i en mindre by i Midtjylland.

Grundet en rokering i det nylige kommunalvalg, er der blevet lovet et større engangsbeløb til de lokale idrætsforeninger, deriblandt Delfinen. Der er desuden også blevet gjort en masse gratis reklame for bl.a. Delfinen, for at få gang i forretningen, hvilket naturligvis har medført en stigning i antallet af henvendelser.

Derfor har bestyrelsen valgt at kassere deres manuelle registreringssystem, og i stedet investere i et administrativt IT-system, der skal kunne gøre de ansattes arbejde lettere og mere overskueligt.

Da Delfinen stadig er en lille klub, som ikke har haft nogen større økonomisk tilslutning fra lokale sponsorer eller fra kommunens side, har den på nuværende tidspunkt kun været muligt at anskaffe sig en enkelt computer, uden opkobling til internettet.

Derfor ønsker bestyrelsen nu et simpelt system, der kan anvendes på computeren, og som desuden lagrer filerne lokalt, grundet den manglende opkobling.

PROBLEMFORMULERING

Af Casper Frost Andersen

Denne rapport har til formål at besvare følgende problemformulering:

> *Hvordan laver vi mest effektivt et administrativt system, som opfylder de krav, som klubben Delfinen søger?*

Dette vil ske ud fra en redegørelse af følgende delpunkter:

- En analyse af svømmeklubben Delfinen som virksomhed (ITO).
- Et design-afsnit, hvori vi både skriftligt og visuelt formulerer hele og dele af vores bud på en løsning (SWD).
- Et afsnit hvor der redegøres for det system vi har lavet, samt argumentation for de løsninger vi har valgt (SWC).

Derudover forholder vi os til følgende spørgsmål:

- Kan der introduceres nogle features som tilgodeser kundens behov, og som stadig er i overensstemmelse med det oprindelige udgangspunkt?

AFGRÆNSNING & SUGGESTIONS

Hele gruppen

AFGRÆNSNING

- Vi afholder os fra at man kan redigere medlemmers stamoplysninger, da det ikke er et krav til funktion af programmet.
- I stævneregistrering: Vi har valgt kun at registrere det nyeste stævne under konkurrencesvømmernes informationer, i stedet for at gemme hvert stævne i hver deres fil.
- Error handling af specielle tegn: user indtaster noget og den ikke prompter for ugyldigt tegn.
- Vi har ikke inkluderet muligheden for at oprette nye ansatte
  (Hvis dette skulle blive en mulighed senere hen, ville det være en funktion som formanden ville have adgang til.)
- Medlemmer har kun ét fornavn og ét efternavn
- Formanden kunne evt. tilføje medlemmer til restance/betalt) når han opretter et medlem. Af tidsmæssige årsager har vi udeladt dette.
- I vores designe class diagram (DCD) er vi opmærksomme på en fejl i nedarvingen fra medlem → Pensionist, senior mv. Vi kunne have oprettet en superclass ved navn kontingent, som pensionist, senior, junior og passiv var nedarvet fra.

SUGGESTIONS

1. Redigere i medlemsoplysninger.
2. Tilføje flere discipliner og tider til medlem.
3. Oprette ansatte.
4. Automatisk restance opdatering på medlem (ud fra dato).
5. Aktivitets diagram med swimlanes.
6. Automatisk opdatering af kontingentpris hvis man flyder år, eller skrifter mellem passiv og aktiv.

HOVEDAFSNIT: IT OG ORGANISATION

Af Rasmus Sadurskis

BRAINSTORM

Fra svømmeklubben- og systemudviklernes side:

Er der nogen som kommer i berøring med klubben?

Hvem er ejerne af Svømmeklubben Delfinen? Hvem har en indflydelse i svømmeklubben?

- Ledelsen
- Formanden
- Kassereren
- Træneren
- Medlemmer

Hvordan har systemudviklerne en indflydelse i svømmeklubben?

- Systemudviklerne skal lave et styresystem, der opfylder de specifikke krav ledelsen har til systemet.
- Udviklerne skal være færdige inden for en bestemt deadline.

Hvilken rolle har formanden i klubben?

- Formanden tager imod nye medlemmer, og registrerer dem med deres stamoplysninger, fx navn, alder osv.

Hvem betaler til klubben?

- Det gør medlemmerne. Kontingenterne lyder som følger:
  - Under 18 år (1000 kr.) årligt.
  - Over 18 år (1600 kr.) årligt.
  - Aktive medlemmer over 60 år får 25% rabat
  - Passivt medlemskab (500 kr.) årligt.

Hvad skal systemudviklerne være opmærksomme på mht. priserne?

- Udviklerne skal være opmærksomme på, om medlemmerne er over eller under 18 år.
- Udviklerne er opmærksomme på om medlemmerne er over 60 år.

Hvem kan blive generet af klubbens tilstedeværelse?

- Konkurrenter til klubben, altså andre svømmeklubber i nærområdet.
- Konkurrencesvømmer der konkurrerer imod Delfinerne.

Er der mulighed for, at skabe en karriere i svømmeklubben?

- Man kan gå til konkurrencesvømning, med en tilknyttet træner dertil.
- Konkurrencesvømmerne er inddelt i to hold: ungdomshold (under 18 år) og seniorhold (18 år og over).

SORTERING

Er der nogen som kan havde en indflydelse og/eller havde en påvirkning?

Man kan sige, at jo mere indflydelse og påvirkning der er i klubben desto mere interesse er der i klubben.

SVØMMEKLUBBENS MATRIX

	Ikke indflydelse i klubben	*Indflydelse i klubben*
Interesse i klubben	Andre svømmeklubber Andre konkurrencesvømmere	Aktive medlemmer Konkurrencesvømmere Ledelsen Kassereren Træneren Sponsorer
Ikke interesse i klubben	Er der en kantine? Pedellen i svømmeklubben	Passive medlemskaber Mindre klub som er i vækst Rengøring af klubben

SYSTEMUDVIKLERNES MATRIX

	Ikke indflydelse i klubben	*Indflydelse i klubben*
Interesse i klubben	Andre systemudviklere	Os som systemudviklere Ledelsen (et hurtigt system)
Ikke interesse i klubben	Det gamle styresystem (manuelt)	Skift af gamle styresystem

INTERESSENTERNE

Hvis man skulle udvælge de vigtigste interessenter der måtte være.

Hvad ønsker ledelsen at se fremadrettet?

- En øget vækst af medlemmer.
- Gode resultater fra konkurrencesvømmerne.
- En øget vækst i økonomien.
- At kunne udkonkurrere de andre svømmeklubber i nærområdet.

Hvad ønsker ledelsen af det nye styresystem?

- Det bliver nemmere at registrere nye medlemmer.
- Systemudviklerne overholder deadlinen.
- Systemet skal som udgangspunkt ikke fornys inden for de første fem år.

Vil der kunne opstå ulemper som ledelsen ikke ønsker at se fremadrettet?

- En faldene vækst af medlemmer.
- Dårlige resultater fra konkurrencesvømmerne.
- Nye svømmeklubber åbner.
- Faldende omsætning.

Hvad ønsker ledelsen ikke at se mht. systemet?

- At systemet ikke lever op til ledelsens krav.
- Udviklerne overholder ikke deadlinen.
- Systemet bliver for hurtigt forældet.

Hvad ønsker udviklerne af ledelsen?

- De har en klar idé om hvad systemet skal kunne.
- En fast deadline.

Hvis man skulle vurdere vigtigheden af medlemmerne i svømmeklubben (1-5)?

- Vigtigheden af medlemmer er en 5'er, for uden medlemmer, ingen svømmeklub.

INTEGRER

Analyse af svømmeklubben / systemudviklerne.

Svømmeklubben skal identificere interessekonflikter, af problemer der kan opstå for svømmeklubben.
En god idé for svømmeklubben er, hvis de laver en *risikoanalyse*, mulige *målsætninger i svømmeklubben*, og så en *kommunikationsplan*.

En RISIKOANALYSE er en god idé, fordi man hurtig kan lave en brainstorm om, hvilke risici der kan opstå for svømmeklubben.

- Vil der opstå strejke?
- Skader på bygningen og deres faciliteter.
- Samfundsøkonomi.

Hvad der kunne opstå for udviklerne.

- Sygdom.
- Overholder ikke deadlinen.

Hvad vil svømmeklubbens målsætninger være? Hvilken vision har svømmeklubben?

- Vil svømmeklubben prøve og etablere sig blandt, Danmarks bedste svømmeklubber?
- Har svømmeklubben en drøm om at skabe Danmarks næste store svømme ikon?

Udviklernes målsætning.

- Holde morgenmøde hver dag.
- Havde et løbende overblik over hvilke mål der er nået.
- Være klar over, hvad der skal laves.

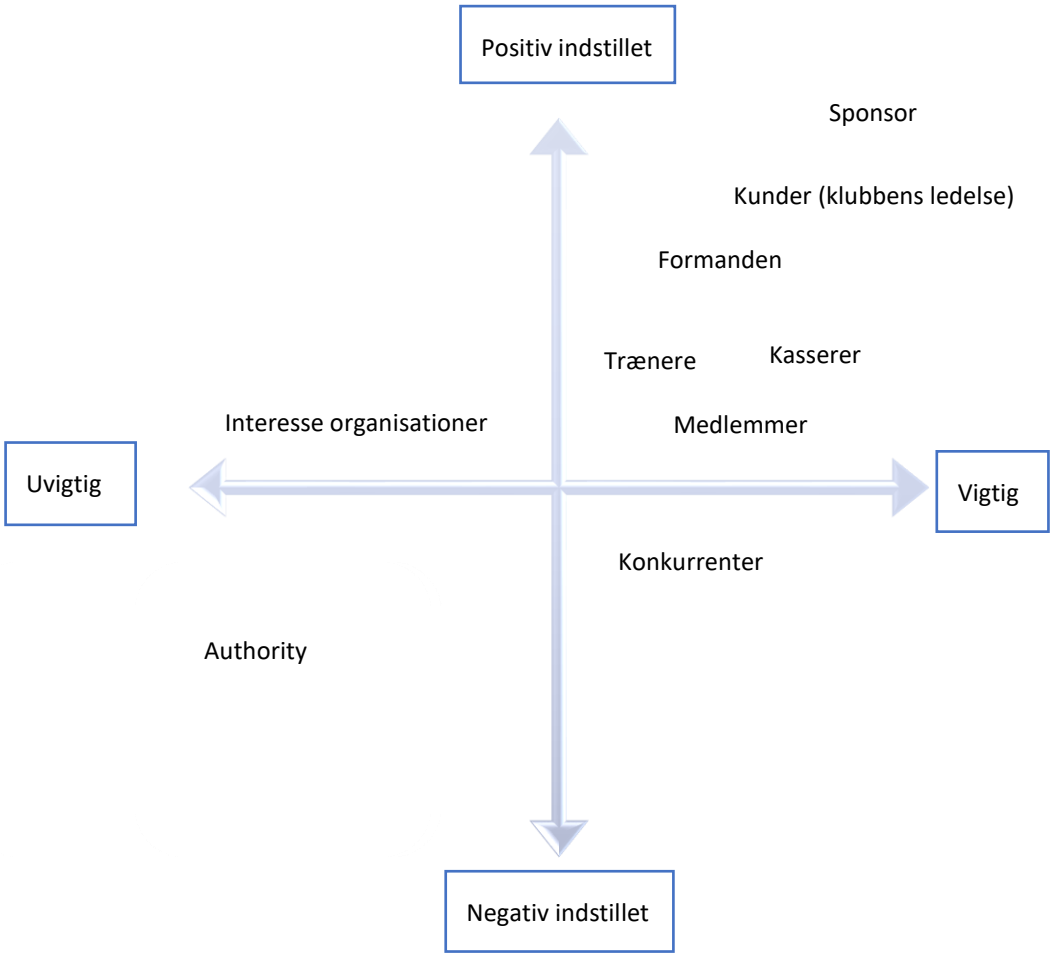
En KOMMUNIKATIONSPLAN tydeliggør interessenternes opmærksomhed. Folk (medlemmer og nysgerrige udefrakommende) vil gerne havde information, og ønsker nogen krav om svømmeklubben, vil de overhovedet havde nogen indflydelse i svømmeklubbens historie? "Medlemmerne" i denne situation er interessenterne, det er dem svømmeklubben skal satse på. Hvordan skal svømmeklubben sende deres budskab ud?

Udviklernes kommunikationsplan er

- Møder op til morgenmøde hver dag.
- Færdig gøre dagens protjek.
- Være sikker på de overholder datelinen.
- Teste færdig kode.
- Brainstorm med resten.

INTERESSENT MATRIX

af Martin Løseth Jensen & Rasmus Sadurski



Figur 1.1: Interessent matrix for svømmeklubben Delfinen
Matrixen viser hvilken indflydelse de forskellige interessenter har på klubben.
- En sponsor har en positiv indflydelse samt en vigtig rolle, for klubben økonomi derfor er den placeret højst oppe mod højre.

SWOT-ANALYSE

Af Rasmus Sadurski & Martin Løseth Jensen

SWOT-analysen skal hjælpe os med, at skabe struktur og overblik over svømmeklubbens styrker og konkurrencesituation. SWOT-analysen giver et virtuelt billede af, hvad svømmeklubbens aktuelle situation er.

SET UD FRA SVØMMEKLUBBEN

Interne forhold	
Stærke sider (+)	Svage sider (-)
- Ny svømmeklub (nyhedsværdien) - Flere aktiviteter og medlemskaber[1] - Central placering - Deltagelse ved sociale aktiviteter - Betaling (at pris og faciliteter hænger sammen) - Konkurrencesvømmerne har deres egen træner - Reklame	- Ny svømmeklub (ukendt, mangel på reklame) - Ikke så mange medlemmer - Dårlig placering - For lav deltagelse ved sociale aktiviteter - Betaling (at pris og faciliteter ikke hænger sammen) - Mangel på konkurrencesvømmere - Mangel på reklame/dårlig reklame[2]
Eksterne forhold	
Muligheder (+)	Trusler (-)
- Bedre faciliteter - Udvidelse af anlæg - Konkurrencestævner - Samarbejdspartnere - Sponsorer	- Konkurrence fra andre svømmehaller - Manglende mulighed for udvidelse af anlæg - Manglende deltagelse ved konkurrencestævner/andre begivenheder - Svømmeklubbers konkurrenceevne - Mangel på samarbejdspartnere - Mangel på sponsorer

Figur 1.2: SWOT-analyse fra svømmeklubbens synspunkt.

1) Da det er en mindre svømmeklub under vækst, ønsker den selvfølgelig at få så mange nye medlemmer som muligt. Det kan svømmeklubben blandet andet gøre via reklame.
2) Det vides ikke om svømmeklubben har reklameret for sig selv, derfor vil det vær en god idé at sende en form for nyhed ud til den lokale befolkning (for at tiltrække nye medlemmer). Svømmeklubben kan evt. reklamere med konkurrencesvømning.

SET UD FRA SYSTEMUDVIKLERNE

Interne forhold	
Stærke sider (+)	Svage sider (-)
- For at få erfaring[1] - Gå på mod - Gode til at samarbejde - Brainstorm	- Mangle på erfaring[2] - Håndtering af projektet - Finde den rigtige arbejdsmetode - Mangle på information - Manglende dynamik i team
Eksterne forhold	
Muligheder (+)	Trusler (-)
- Flere større projekter - God omtale - Flere/større kunder	- Andre systemudviklere tager jobbet - Dårlige omtale - Mangle på kunder - Tid kontra betaling

Figur 1.3: SWOT-analyse fra udviklernes synspunkt.

1) Mere erfaring vi for i dette projekt, jo bedre bliver vi også fremadrettet. Det vil blive nemmere at få et overblik, over nye projekter. Ved flere projekter giver det også en bedre forståelse på en god arbejdsmetode.
2) Da vi ikke har så meget erfaring, hverken i kode heller designe, giver det os, nogen få problemer hvordan man skal se projektet. Hvordan skal vi håndtere projekt, hvilken arbejdsmetode bruger vi.

HOVEDAFSNIT: SOFTWARE DESIGN

HISTORIK

Dato	Version	Forfatter	Beskrivelse
20-11-2017	00.01.01	Hele gruppen	Første sammensætning af OOAD.
21-11-2017	00.02.01	Hele gruppen	Implementering af OOAD i rapport.
24-11-2017	00.02.02	Hele gruppen	Tilføjelse af UC007 + SSD, tilføjelse til SWOT.
24-11-2017	00.02.03	Hele gruppen	Tilføjelse af arbejdsprocesdel.
27-11-2017	00.02.04	Hele gruppen	Rettelser i nuværende use cases, SSD'er Tilføjet: UC009 SD Use case diagram
28-11-2017	00.02.05	Hele gruppen	Print af restance list Print betalingsstatus list Register konkurrencesvømmer til stævner Gemmer til stævner fil Printer konkurrencesvømmer i udvalgte disciplin US001 US002 US008
29-11-2017	00.02.06	Hele gruppen	Testning af SWC Update af SWD Rettelser/update af Faseplan
30-11-2017	00.02.07	Hele gruppen	Redigér kontingent Format på console og når det gemmes i fil Opfylder krav for aktiv og passiv UC004 top 5
1-12-2017	00.02.08	Hele gruppen	Opdatering af UCD, tilføjelse af 3 brief UC's Print stævneliste Kontintentpris Print junior og senior hold

4-12-2017	00.03.01	Hele gruppen	Ret rapport SD Formatering Træner Snippets af kode til rapport
4-12-2017	00.03.02	Hele gruppen	Tilføjet udkast til indledning og problemformulering.
5-12-2017	00.03.03	Hele gruppen	Tilføjet hovedafsnit: konstruktion
6-12-2017	01.00.00	Hele gruppen	Færdiggjort hovedafsnit: konstruktion og resten af rapport. Klar til aflevering.
07-12-2017	01.00.01	Casper Frost Andersen	Redigering af problemformulering

VISION

Af Casper Frost Andersen

Vi ønsker at lave et administrativt program til svømmeklubben Delfinen. Programmet skal håndtere indmelding af nye medlemmer samt deres specifikationer, herunder deres personlige data, som alder, medlemstype, og aktivitetsniveau (motionist/konkurrence).

Herudover skal medlemmernes kontingenter registreres ved indmeldelse. Kontingentet er reguleret af flere variabler, herunder alder, aktivitetsform og type af medlemskab. En kasserer skal kunne se en oversigt over medlemmer i restance.

En træner, der er tilknyttet svømmeklubben, skal kunne se en oversigt over deltagerne på sine hold, samt deres discipliner og bedste tider, indenfor disse.
Ydermere skal træneren kunne få printet en top 5 oversigt over de bedste konkurrencesvømmere inden for de forskellige discipliner, i forbindelse med udtagelse til stævner.

KRAV

Af Casper Frost Andersen

- Systemet skal kunne lagre data af forskellige typer fra forskellige brugere.

  Overordnet er der tre typer af data, der skal gemmes:
  - Medlemmernes informationer
  - Kontingenter og restancer
  - Medlemmernes svømmetider
  - Data relateret til stævner, hvor stævnernes navne, og svømmernes tider indgår.

- Formanden vil være den eneste, der kan oprette nye medlemmer i svømmeklubben. De nye medlemmer skal opgive deres navn, type medlemskab (aktiv/passiv), alder og aktivitetsform (motionist/konkurrencesvømmer).

- Kassereren håndterer alle kontingenter og restancer. Vedkomne skal være i stand til at udskrive en oversigt over medlemmer i restance. De forskellige typer abonnementer er:
  - Aktiv:
    - Under 18: 1000 kr.
    - Over 18: 1600 kr.
    - Senior over 60 får 25% rabat.
  - Passivt: 500 kr.

  Alle gebyrer indbetales årligt.

- Træneren har to hold: en for medlemmer under 18 år (ungdomshold) og en for over 18 (seniorhold).
  Hvert medlem har deres disciplin(er) tildelt i datafilen.
  Hver svømmer får deres bedste tid registreret, og den dato den blev opnået.
  Hvis nogen af medlemmerne konkurrerer til et stævne, gemmes navnet på stævnet sammen med svømmernes placering og tid.
  Træneren skal være i stand til at udskrive en top 5 af svømmerne i alle discipliner, med henblik på udtagelse til stævner.
  Disse er:
  - Butterfly
  - Crawl
  - Rygcrawl
  - Brystsvømning
  - Hundesvømning

USE CASE DIAGRAM

Af Casper Frost Andersen & Christian Strunge

Figur 2.1: Use case diagram: visuelt diagram over alle funktioner programmet indeholder.

USE CASES

Hele gruppen

BRIEF

USE CASE 004

Use Case Section	Kommentar
Use Case ID	Use Case 004
Use Case Navn	Print Udtagelseskandidater
Main Succes Scenario	Træner skal kunne se en liste over den nuværende top 5 (de 5 bedste tider) inden for hver enkel disciplin.

USE CASE 005

Use Case Section	Kommentar
Use Case ID	Use Case 005
Use Case Navn	Print konkurrencesvømmere
Main Succes Scenario	Træner skal kunne printe en liste over alle konkurrencesvømmere.

USE CASE 006

Use Case Section	Kommentar
Use Case ID	Use Case 006
Use Case Navn	Rediger betalingsstatus
Main Succes Scenario	Kassér skal kunne ændre medlemmers kontingent.

USE CASE 7

Use Case Section	Kommentar
Use Case ID	Use Case 007
Use Case Navn	Registrer stævne
Main Succes Scenario	Træneren skal kunne registrere konkurrencesvømmere som har deltaget i stævner.

USE CASE 9

Use Case Section	Kommentar
Use Case ID	Use Case 009
Use Case Navn	Login
Main Succes Scenario	Brugeren af systemet skal have mulighed for at logge ind som en bruger

USE CASE 10

Use Case Section	Kommentar
Use Case ID	Use Case 010
Use Case Navn	Print betalingsstatus
Main Succes Scenario	Kasserer skal kunne printe en liste over alle medlemmer samt deres betalingsstatus

USE CASE 11

Use Case Section	Kommentar
Use Case ID	Use Case 011
Use Case Navn	Print stævneliste
Main Succes Scenario	Træner skal kunne se en liste over samtlige konkurrencesvømmere, der har deltaget i stævne med tid og placering

USE CASE 12

Use Case Section	Kommentar
Use Case ID	Use Case 012
Use Case Navn	Print junior/senior hold
Main Succes Scenario	Træner skal kunne printe 2 hold ud, hhv. junior- og seniorhold, fra konkurrencesvømmerlisten.

USE CASE 13

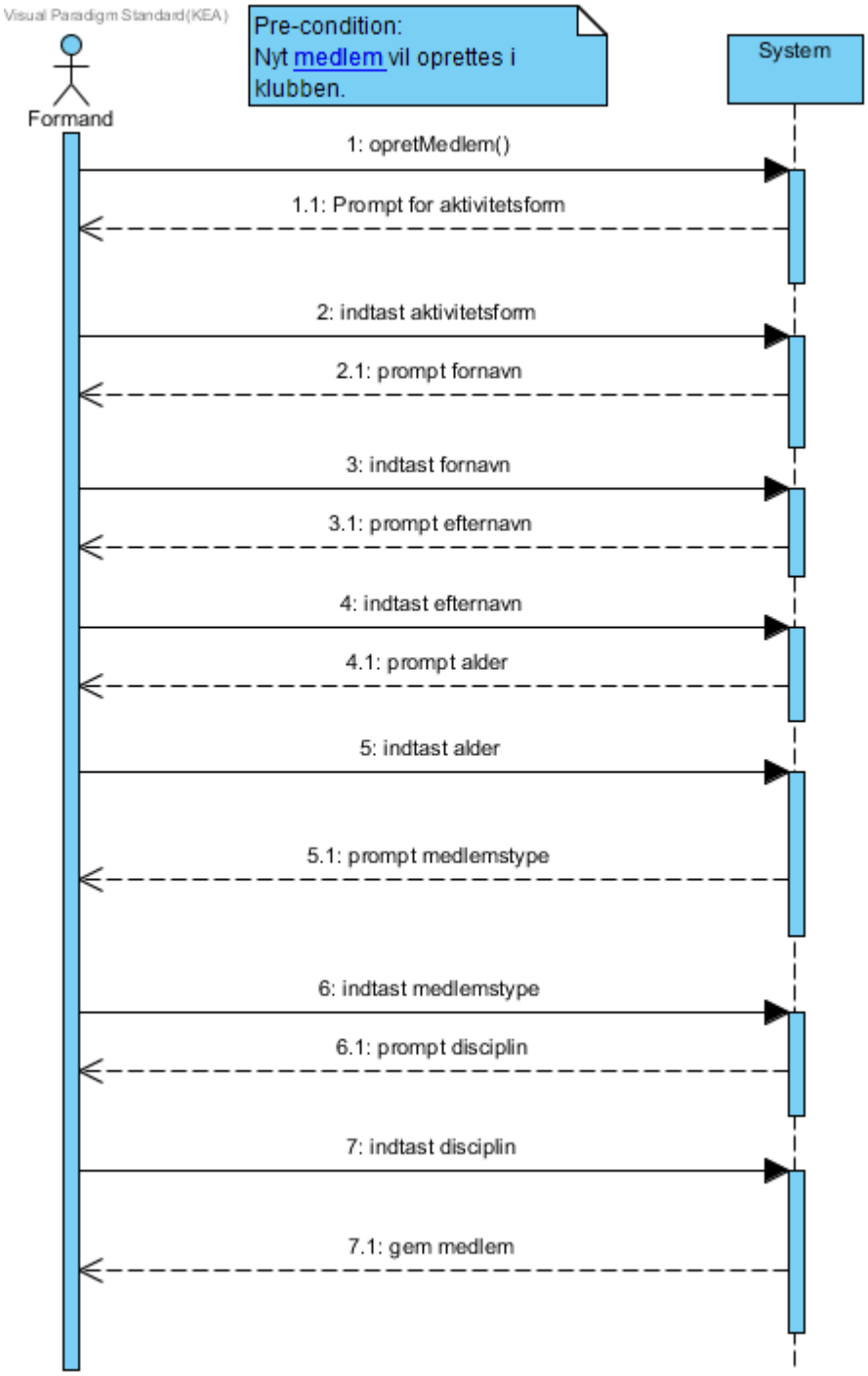
Use Case Section	Kommentar
Use Case ID	Use Case 013
Use Case Navn	Print et medlems kontingentpris
Main Succes Scenario	Kassereren skal kunne se en liste over medlemmer, vælge et medlem, og se dennes kontingentbetaling.

FULLY DRESSED

USE CASE 001

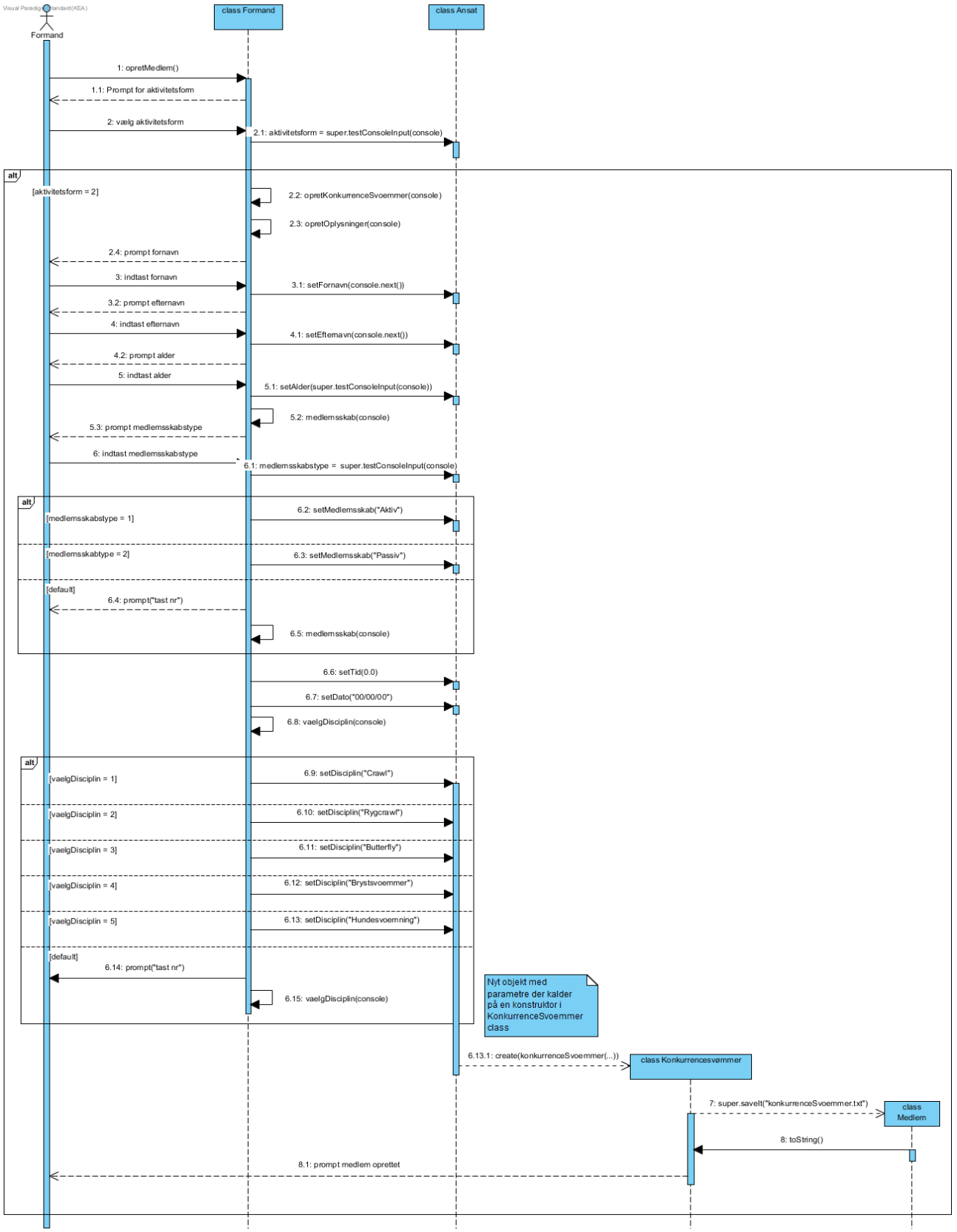
Use Case Section	Kommentar
Use Case ID	UC001
Use Case Navn	Opret medlem
Scope	Svømmeklub Delfinen administrativt system
Beskrivelse	Formand skal kunne oprette nye medlemmer, og tilknytte staminformation og aktivitetsform.
Aktør(er)	Formand
Stakeholders	Formand for svømmeklubben Delfinen
Preconditions	Nyt medlem vil oprettes i klubben
Succes garanti	Medlemmets informationer bliver gemt i systemet
Main flow	1. Formanden vælger *opret nyt medlem* fra en menu. 2. Systemet prompter for aktivitetsform 3. Formand indtaster aktivitetsform 4. Systemet prompter for fornavn 5. Formand indtaster fornavn 6. System prompter for efternavn 7. Formand indtaster efternavn 8. System prompter for alder 9. Formand indtaster alder 10. System prompter for medlemstype 11. Formand indtaster medlemstype 12. Systemet prompter for disciplin 13. Formand indtaster disciplin 14. Systemet gemmer medlemmet i fil.
Extensions	Gældende for trin 3, 5, 7, 9, 11, 13: 1. Formand indtaster ugyldigt input 2. Hvis input er gyldigt, fortsæt da til næste trin. Ellers returner til samme trin. 3a. Formand vælger motionist 1. Følger trin 4 til 11. 2. Skipper trin 12 & 13 3. Systemet gemmer medlem i fil.

UC001 SSD



Figur 2.2: UC001 System Sequence Diagram (fremover SSD)

SEQUENCE DIAGRAM UC001

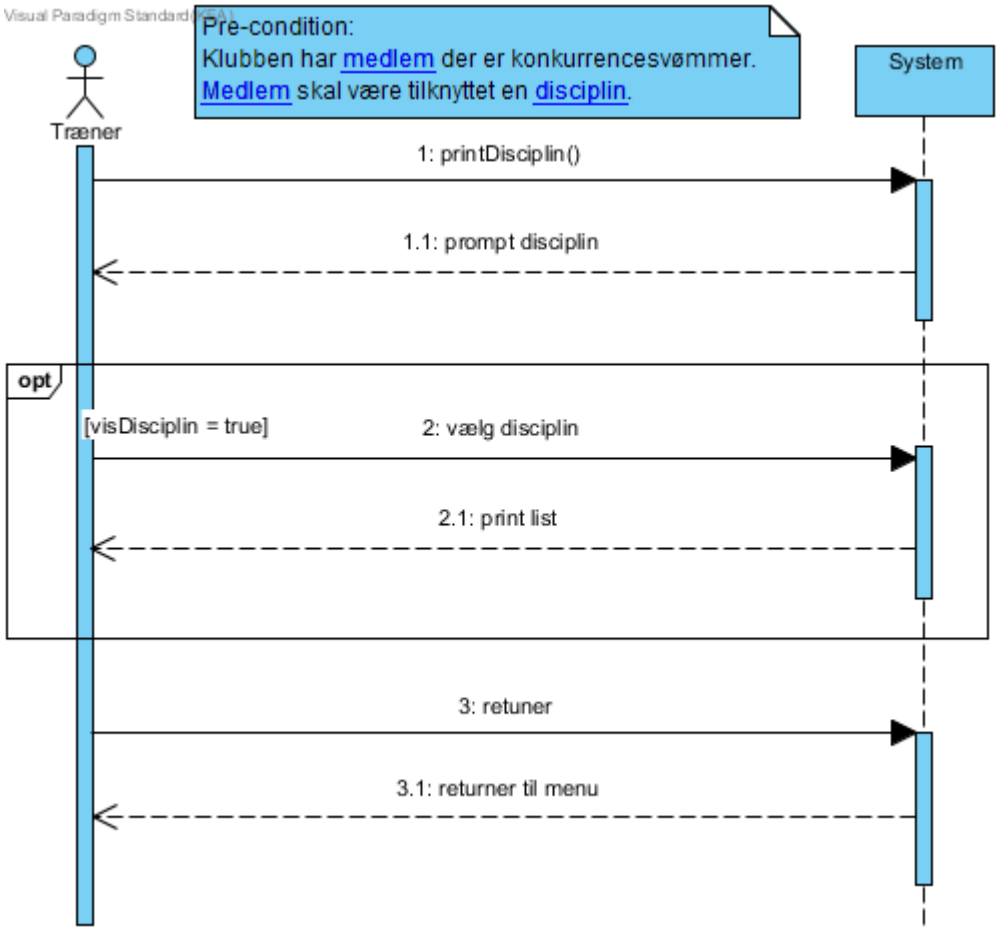


Figur 2.3: Sequence Diagram for Use case 1.

USE CASE 002

Use Case Section	Kommentar
Use Case Navn	Print svømmediscipliner
Use Case ID	UC002
Scope	Svømmeklub Delfinen administrativt system
Beskrivelse	Træner skal kunne se en eller flere lister over svømmere, der er tilknyttet de forskellige svømmediscipliner.
Aktør(er)	Træner
Stakeholders	Træner for svømmeklubben Delfinen
Preconditions	Klubben har medlemmer der er konkurrencesvømmere. Medlem skal være tilknyttet en disciplin.
Succes garanti	En liste over svømmere og de tilknyttede discipliner bliver printet.
Main flow	1. Træner vælger *print discipliner* fra træner menu. 2. System prompter for disciplin. 3. Træner vælger disciplin. 4. System printer liste 5. System prompter om returnering 6. Træner vælger at returnere 7. System returnerer til ”træner menu”
Extensions	3a. Træner indtaster ugyldigt input. 1. System prompter for gyldigt navn for disciplin. 2. Hvis navn er gyldigt, fortsæt til punkt 4. Ellers returner til 3a 5a. Træner vælger at printe en ny liste ud. 1. Returner til punkt 2. Ellers fortsæt til punkt 6.

UC002 SSD

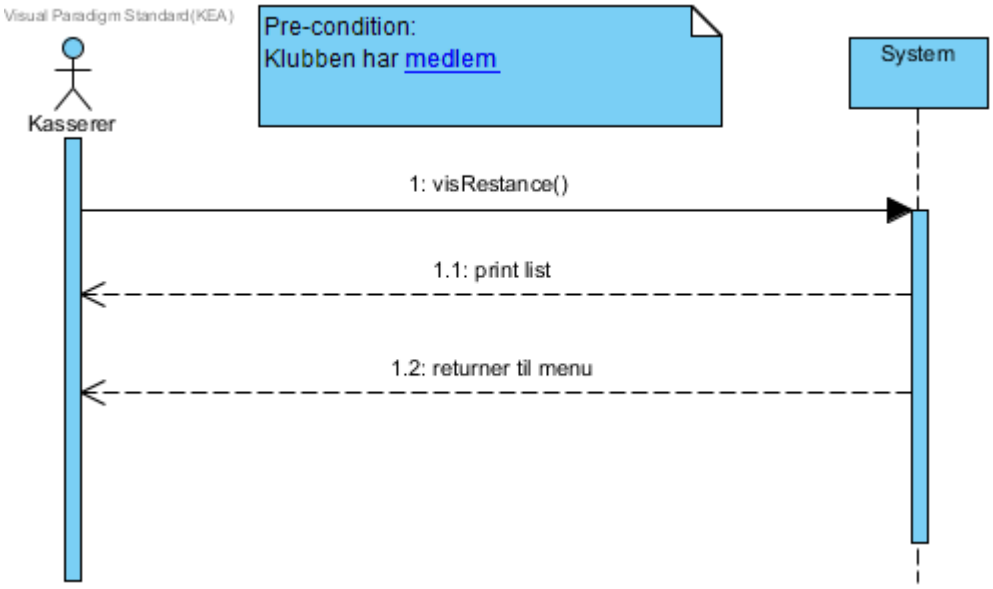


Figur 2.4: SSD for Use case 002

USE CASE 003

Use Case Section	Kommentar
Use Case Navn	Print restancer
Use Case ID	UC003
Scope	Svømmeklub Delfinen administrativt system
Beskrivelse	Kassereren skal kunne se en liste over de medlemmer der er i restance.
Aktør(er)	Kasserer
Stakeholders	Kasserer for svømmeklubben Delfin
Preconditions	Medlem har betalt ved oprettelse
Succes garanti	En liste over medlemmer der er i restance bliver printet ud på konsollen.
Main flow	1. Kasserer vælger *print restanceliste* fra menu. 2. System printer liste. 3. System returnerer til menu.
Extensions	1a. Ingen medlemmer er i restance. 1. System printer besked og fortsætter til punkt 3.

UC003 SSD

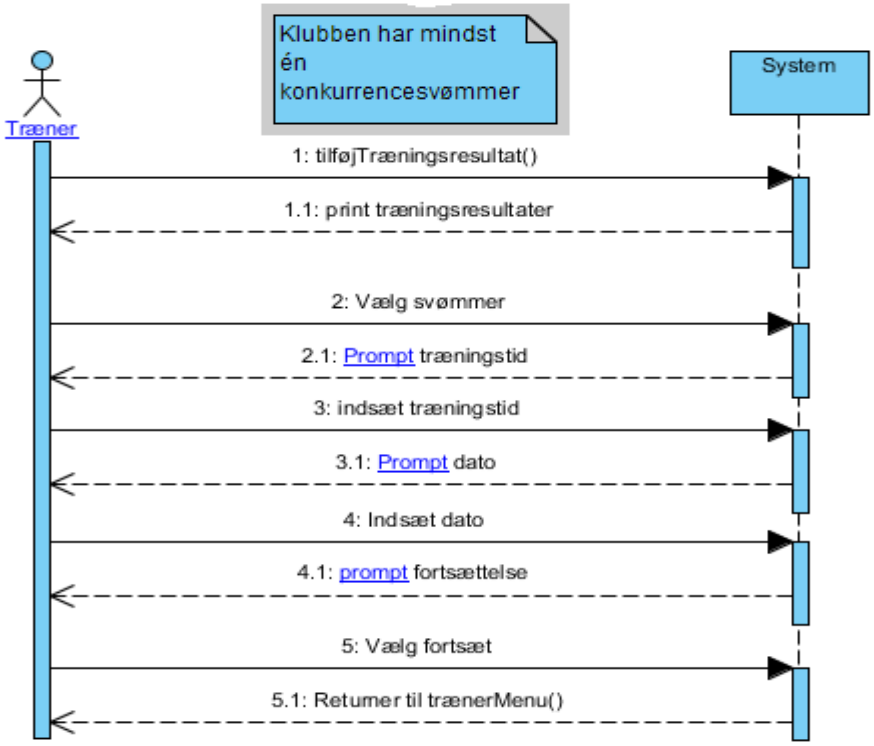


Figur 2.5: SSD for Use case 003

USE CASE 8

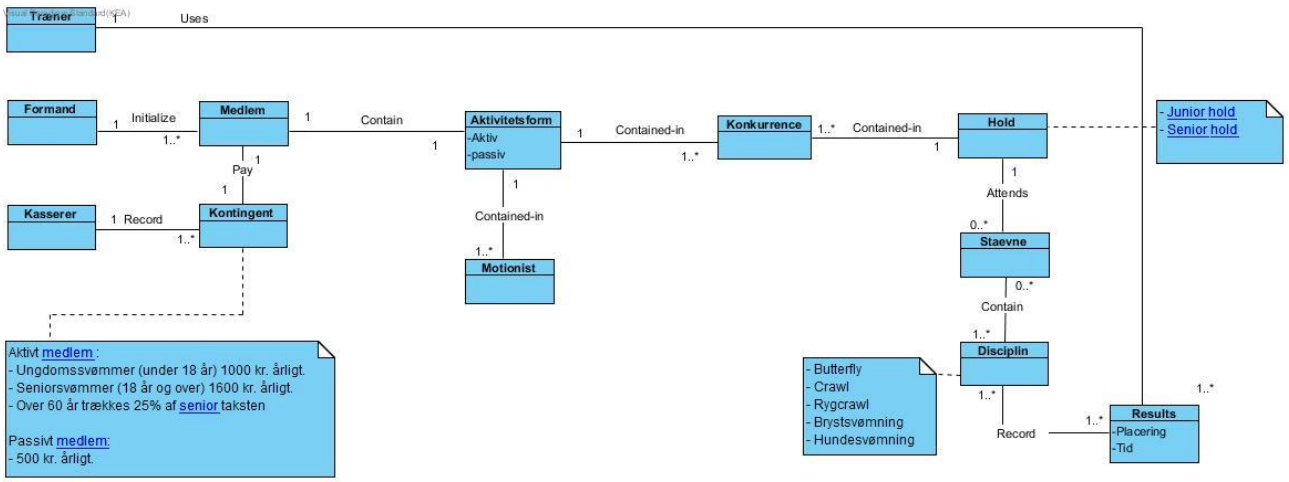
Use Case Section	Kommentar
Use Case Navn	Opret ny bedste tid
Use Case ID	UC008
Scope	Svømmeklub Delfinen administrativt system
Beskrivelse	Træner skal kunne tilføje træningsresultat og dato til konkurrencesvømmere, og med mulighed for gentagelse
Aktør(er)	Træner
Stakeholders	Træneren i svømmeklubben Delfinen
Preconditions	Klubben har mindst én konkurrencesvømmer
Succes garanti	Et træningsresultat og dato for denne er blevet tilknyttet en eksisterende konkurrencesvømmer
Main flow	1. Træner vælger *tilføj træningsresultat* fra *træner menu* 2. System printer liste over eksisterende konkurrencesvømmer(e) 3. Træner vælger svømmer fra liste 4. Systemet prompter for træningstid 5. Træner indtaster træningstid 6. Systemet prompter for dato 7. Træner indtaster dato 8. System prompter om fortsættelse 9. Træner vælger at fortsætte 10. System returnerer til *træner menu*
Extensions	3a. Træner giver ugyldigt input 1. prompt for gyldigt input 2. Hvis input er gyldigt, fortsæt til punkt 4 5a. Træner giver ugyldigt input 1. prompt for gyldigt input 2. hvis input er gyldigt, fortsæt til punkt 6 7a. Træner giver ugyldigt input 1. prompt for gyldigt input 2. hvis input er gyldigt, fortsæt til punkt 8 9a. Træner vælger at indtaste nyt træningsresultat 1. returner til punkt 2

UC008 SSD



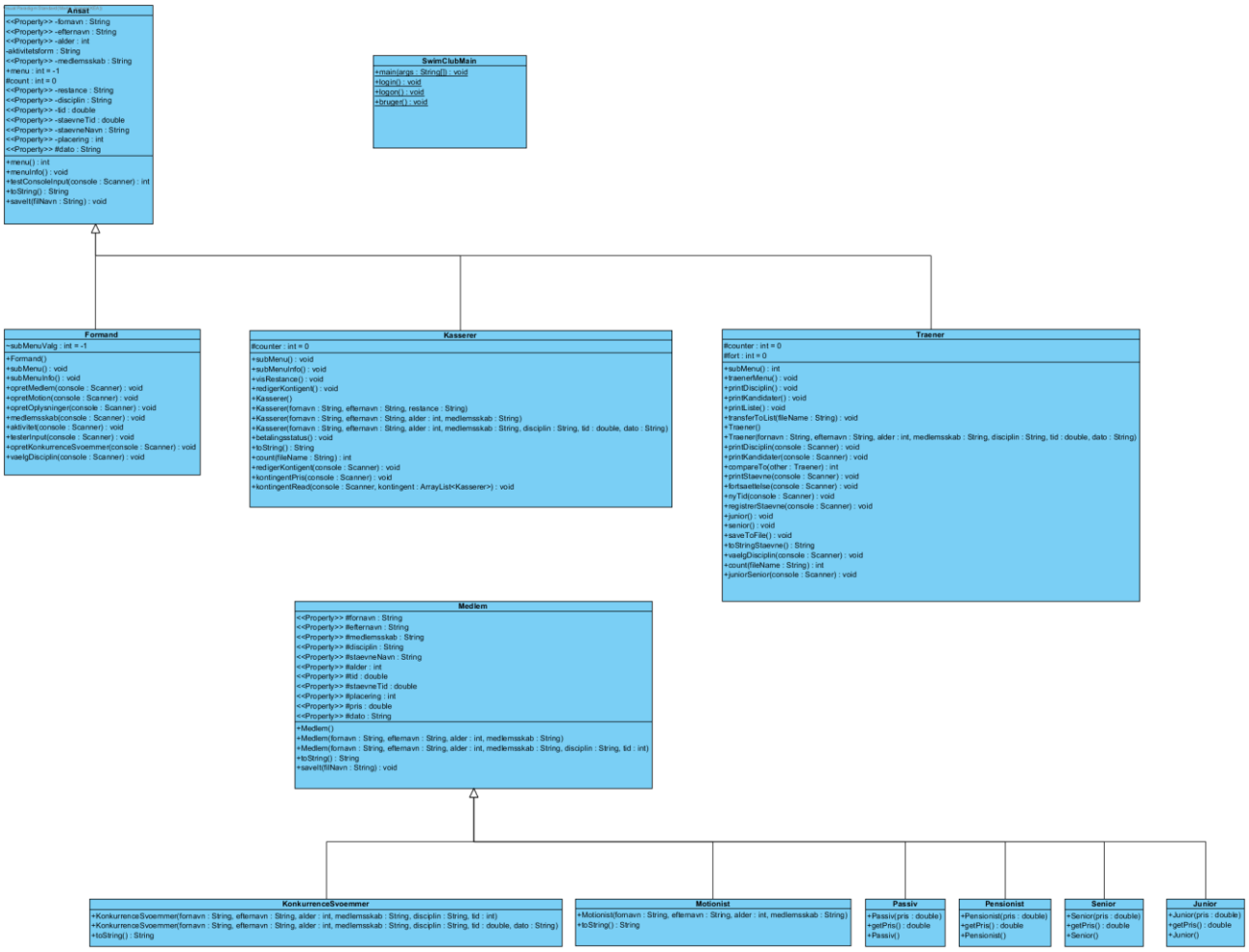
Figur 2.6: SSD for Use case 008

DOMAIN MODEL



Figur 2.7: Domain model for programmet til svømmeklubben Delfinen.

DESIGN CLASS DIAGRAM



Figur 2.7: Design Class Diagram over det færdige produkt.

FURPS

Af Casper Frost Andersen

FUNKTIONELT

- Systemet skal kunne gemme data om:
    - Medlemmer og deres medlemskabstyper og aktivitetsform.
    - Kontingenter og restancer for medlemmer.
    - Medlemmernes bedste tider i de forskellige discipliner.
    - Oversigt over konkurrencesvømmeres deltagelse i stævner (Placering, bedste tid og stævnets navn).

- Formanden i klubben skal være i stand til at oprette nye medlemmer i svømmeklubben.
- Kassereren skal kunne se et udprint over medlemmernes restancer i programmet.
- Træneren skal være i stand til at se et overblik over de fem bedste svømmere i alle discipliner, med henblik på udtagelse til stævner.
- Systemet skal kunne gemme data på lokale filer.

BRUGERVENLIGHED

- Forskellige brugere kommer til at have forskellige funktioner i systemet (formanden kan for eksempel oprette nye medlemmer, men vil ikke have adgang til de økonomiske funktioner).
- Systemet skal være brugervenligt.
- Bruger skal kunne hente tilgængeligt indhold fra lagrede data.

PÅLIDELIGHED

- Systemet vil løbende indsamle nye data, som brugerne indtaster. Disse bør bakkes op på eksterne lagerenheder en gang pr. uge, i tilfælde af et system-breakdown eller andre hændelser, der kan resultere i tab af data.
- Medlemmerne data skal opbevares på en password beskyttet computer.

YDEEVNE

- Systemet skal skrives uden redundans og unødvendige beregninger, for at sikre optimal program-hastighed.
- Programmet kræver ikke meget hukommelse, så programmet bør fungere optimalt på alle nyere computere.

SUPPORT

- Programmet skal være letlæseligt og skal kunne tilpasses, hvis kunden ønsker nye funktioner tilføjet.
- Programmet skal være veldokumenteret, så det er let at vedligeholde.

HOVEDAFSNIT: SOFTWARE KONSTRUKTION

INDLEDNING

Af Casper Frost Andersen & Christian Strunge

Dette afsnit har til formål at redegøre for selve programmeringsprocessen. I afsnittet vil vi komme ind på vores fremgangsmåde, en overordnet gennemgang af koden, eksempler fra source koden og argumentation for, hvorfor vi har valgt netop de løsninger, og til sidst en redegørelse af de ekstra features vi har implementeret.

FREMGANGSMÅDE

Af Casper Frost Andersen & Christian Strunge

Vi har fra starten aftalt at holde møder dagligt, hvor vi forventningsafstemmer i forhold til hvor langt vi er nået i opgaven – har vi nået det vi skulle, har vi fået nogle gode idéer, og hvad vi skal lave fremadrettet. Derudover har vi også holdt mandags- og fredagsmøder, hvor vi hhv. redegør for ugens delmål, og efterfølgende opsummerer og reflekterer over ugens forløb.

Vi har taget udgangspunkt i de Use cases vi har lavet i design-delen, og iterativt tilføjet dem til programmet. Efter hver implementering har vi testet og diskuteret, om koden passer til den allerede eksisterende kode. Efterfølgende har vi reflekteret over suggestions, afgrænsning, og om vi kan undgå redundans.

I de løbende møder vi har haft, har vi diskuteret opgavekravene, og flere gange opdaget krav vi manglede at tilføje til programmet (se bilag). Det har medført at vi har haft brug for at omstille og tilpasse koden, så den passer til de krav der løbende blev stillet.

USE CASES TIL KONSTRUKTION: EKSEMPLER

Af Casper Frost Andersen & Christian Strunge

I det følgende afsnit vil vi redegøre for nogle Use cases, og hvordan vi er kommet fra tanke til produkt, samt hvilke overvejelser vi har gjort os løbende. Eftersom vi har opbygget vores produkt rundt om de Use cases vi har med i denne rapport, er det vigtigt at belyse vores fremgangsmåde og resultat.
Vi har her taget udgangspunkt i 3 Use cases, som hver gør noget forskelligt.

UC001 – OPRET MEDLEM

Et af kravene til opgaven var, at en formand skulle kunne registrere og gemme nye medlemmer med deres stamdata (navn, alder, aktivitetsform og medlemskabstype). Vi har lavet en superclass kaldet ”Ansat”, og en af subclasserne har vi døbt ”Formand”. Formand classen bruger fields fra Ansat til at oprette et nyt objekt, som er det nye medlem.

I første omgang registrerede vi løbende alle medlemmer i samme tekstfil, men vi fandt hurtigt ud af, at det kunne være en fordel at inddele dem i to filer, kaldet ”motionister.txt” og ”konkurrenceSvoemmere.txt”, da sidstnævnte har en del flere funktioner i programmet end førstnævnte (blandt andet disciplin, bedste tid og dato for denne).

Vi startede med blot at gemme objekterne i filerne, men konkurrencesvømmere skal kunne inddeles og sorteres efter disciplin og bedste tid. Derfor valgte vi at oprette medlemmerne med flere tokens, som vi senere hen i programmet har kunne bruge til netop disse funktioner.

Derudover har vi også tilføjet medlemskabstypen som token, i og med at vi har tilføjet en funktion, hvor Kassereren skal kunne printe et medlems kontingent ud. Her vælger Kassereren om det skal være en motionist eller en konkurrencesvømmer, der skal vises, og kassereren kan derefter vælge medlemmet ud fra en liste der bliver printet til konsollen.

Når et medlem bliver oprettet, har vi sat en standard træningstid og dato. Dette har vi gjort for at kunne læse dem som tokens, og kunne redigere i dem senere, da et af kravene til opgaven var, at Træneren skal kunne opdatere træningstider løbende i udtagelsesøjemed (se UC008).

Resultat: Vi har nu en funktionsdygtig class der opretter et medlem med diverse stamdata, og gemmer dem i to separate filer, en til motionister og en til konkurrencesvømmere.

UC003 – PRINT RESTANCER

Kassereren I svømmeklubben har bedt om at kunne få printet en liste ud over medlemmer i restance.

Vi fortolkede i første omgang opgaven som, at der ikke var et betalingssystem tilknyttet programmet. Derfor valgte vi at lave en statisk liste i en tekstfil, hvorfra Kassereren kunne printe oversigten. I den liste tilføjede vi x antal medlemmer med fornavn, efternavn og betalingsstatus (betalt/restance).

Måden vi valgte at gøre det på var, at gemme fornavn, efternavn og betalingsstatus som tokens i en ArrayList. Herved kunne vi lave en metode, hvor vi går ind og scanner hvert enkelt token, og hvis et givent token stod som ”restance”, ville linjen blive printet ud.

Efter nogle overvejelser vi gjorde os efterfølgende, kom vi frem til, at det ville være hensigtsmæssigt at tilføje en funktion, som kunne ændre medlemmernes betalingsstatus.
Til denne funktion har vi valgt at bruge et 2-dimensionelt array, for at kunne gå ind og ændre i det token hvor betalingsstatussen står, uden at ændre i fornavn og efternavn.

Efter tilføjelsen af denne funktion reflekterede vi også over, at man kun kunne se en liste over medlemmer i restance, og det derfor ikke var muligt at sætte medlemmer der havde betalt, i restance. Vi tilføjede derfor en simpel metode der læste alle linjerne i filen og printede dem ud til konsollen.

Resultat: Vi har nu lavet en class der kan printe en liste over alle medlemmers på listen i restance, der kan printe en liste over alle medlemmer på listen og deres betalingsstatus, og som kan redigere i medlemmernes betalingsstatus. Vi har valgt at tilføje de sidste to funktioner for at gøre programmet mere brugervenligt og for at det bedre kan simulere et program der kan bruges i den virkelige verden.

UC008 – OPRET NY BEDSTE TID

Det var et krav til opgaven, at træneren løbende skulle kunne opdatere konkurrencesvømmernes træningstider, i forbindelse med udtagelse til stævner.

Vi skulle derfor oprette en metode til at kigge på filen ”konkurrenceSvoemmere.txt”, som løbende får tilføjet nye medlemmer, finde en svømmer, og derefter opdatere svømmetiden og datoen den blev sat på.

Først ville træneren få printet en liste over alle konkurrencesvømmere der er registreret i klubben. Hvert medlem har et nummer tilknyttet til sit navn, og ved indtastning af dette, vil man komme ind på dette medlem. Dernæst kan man indtaste den nye tid og dato.

Vi har valgt at bruge et 2d-array for, at kunne indtaste nummeret på de indekserede medlemmer, og derefter erstatte den gamle tid og dato med den nye, uden at skulle overskrive hele linjen.
Nye medlemmer får tildelt en standard tid og dato (hhv. 0,0 og 00/00/00). Dette er for at vi igen kan inddele stringen i tokens, og vælge hvilke vi vil søge efter og erstatte

Resultat: Vi har nu fået implementeret en løbende funktion, hvor træneren kan opdatere tider og datoer løbende, også for nyligt tilføjede medlemmer.

OPSUMMERING:

Vi har fået konstrueret flere metoder, hvis funktionaliteter lever op til kundens krav. Nogle af metoderne kan endvidere bruges i de andre klasser, hvilket simplificerer koden og gør den mere kompakt.

Disse Use cases repræsenterer kun en lille del af programmet, men de dækker de fleste af de funktioner vi bruger i løbet af koden.

REDEGØRELSE AF UDVALGTE METODER

Af Rasmus Sadurskis & Martin Løseth Jensen

OPRETMEDLEM()

Af Martin Løseth Jensen

Når man vælger at oprette et medlem, bliver man spurgt om hvilket medlem man vil oprette: en motionist eller konkurrencesvømmer. Grunden til dette er, at der promptes for flere informationer for en konkurrencesvømmer, end der gør for en motionist.
Vi har derfor valgt to metoder, der får gemt informationerne som medlem på hver sin måde.
Vi har en klasse til "Motionist", og en klasse til "KonkurrenceSvoemmer". Dette har vi gjort så vi kan overskrive metoder der er nedarvet fra forældre klassen "Medlem".

```
  KonkurrenceSvoemmer nytMedlem = new KonkurrenceSvoemmer(getFornavn(),getEfternavn(),
                        getAlder(),getMedlemsskab(),getDisciplin(),getTid(),getDato());
public class KonkurrenceSvoemmer extends Medlem {

   public KonkurrenceSvoemmer(String fornavn, String efternavn, int alder, String medlemsskab,
                                    String disciplin, double tid, String dato)throws Exception{
      setFornavn(fornavn);
      setEfternavn(efternavn);
      setAlder(alder);
      setMedlemsskab(medlemsskab);
      setDisciplin(disciplin);
      setTid(tid);
      setDato(dato);

      //gemmer til fil
      super.saveIt("konkurrenceSvoemmere.txt");

      System.out.println("...MEDLEM GEM SOM KONKURRENCESVOEMMER...\n\n");
   }

   //overskriver toString metoden i forældreklassen Medlem
   @Override
   public String toString(){
      return getFornavn()+" "+getEfternavn()+" "+getAlder()+" "
               +getMedlemsskab()+" "+getDisciplin()+" "+getTid()+" "+getDato();
   }
}
```

Her ses KonkurrenceSvoemmer klassen. Constructoren får overført sin værdier og kalder derefter en metode fra forældre klassen "Medlem", "super.saveIt(…)".
Der opnås også polymorphisme ved at man overskriver toString().

Her ses forældre klassen "Medlem", som har de to metoder der bruges defineret. Det er disse der overskrives.

Så ved at vi får oprettet nogle klasser der har hver deres værdier der skal gemmes, opnås polymorphisme hvilket er samme kode, men med andet output.

```
public String toString(){ return " "; }

//Metode til at gemme til fil
//FileOutputStream er stream der sætter sig på det allerede eksiterende stream
public void saveIt(String filNavn)throws Exception{
   PrintStream nytMemberInfo = new PrintStream(new FileOutputStream(filNavn,true));
   nytMemberInfo.println(toString());
   System.out.print("\n");
}
```

Kode snippet 2 saveIt(...) gemmer den nye stream oven på den nuværende stream.

SUBMENU()

Af Rasmus Sadurskis

subMenu metoden arbejder sammen med vores *testConsoleInput* som stammer fra Ansat classen. Den returnerer this.menu til vores subMenu() med en værdi, hvorefter den tester switchens cases', om den kan finde en værdi der passer til en case, ellers bliver kører default.

Vi opretter en ny Scanner objekt, hvor den læser det der bliver indtastet, når den kommer ind i en case, så sker der et metodekald, med parmeteroverførsel. Det den får overført er console, hvilket er det Scanner objekt vi har fået initialiseret i subMenu() metoden.

```
//Undermenu metode
public void subMenu()throws Exception {
    Scanner console = new Scanner(System.in);
    int menu = -1;

    while(menu != 0){
        subMenuInfo(); //linje 74

        super.testConsoleInput(console); //linje 86 i Ansat klassen
        //Indhold af menu til kassér
        //"menu = 0" så submenuen kører indtil man selv
        //ber den om at returnere
        switch(this.menu){
            case 1:
                visRestance(); //linje 112
                menu = 0;
                break;
            case 2:
                betalingsstatus(); //linje 84
                menu = 0;
                break;
            case 3:
                redigerKontigent(console); //linje 169
                menu = 0;
                break;
            case 4:
                kontingentPris(console); //linje 244
                menu = 0;
                break;
            case 0:
                this.menu = -1; //for at den ikke også hopper ud af ansats menu
                super.menu(); //menu hos ansat
                break;
            default:
                System.out.println("TAST VENLIGST ET NUMMER DER ER FREMVIST\n");
                subMenu();
        }
        break; //for kun at komme ud af dette loop
    }
}
```

Figur snippet 1 subMenu() metode

TESTCONSOLEINPUT()

Af Martin Løseth Jensen

subMenu metoden arbejder sammen med vores *testConsoleInput* som stammer fra Ansat classen. Den returnerer this.menu til vores subMenu() med en værdi, hvorefter den tester switchens cases', om den kan finde en værdi der passer til en case, ellers bliver kører default.

Vi opretter en ny Scanner objekt, hvor den læser det der bliver indtastet, når den kommer ind i en case, så sker der et metodekald, med parmeteroverførsel. Det den får overført er console, hvilket er det Scanner objekt vi har fået initialiseret i subMenu() metoden.

```
//Undermenu metode
public void subMenu()throws Exception {
    Scanner console = new Scanner(System.in);
    int menu = -1;

    while(menu != 0){
       subMenuInfo(); //linje 74

       super.testConsoleInput(console); //linje 86 i Ansat klassen
       //Indhold af menu til kassér
       //"menu = 0" så submenuen kører indtil man selv
       //ber den om at returnere
       switch(this.menu){
          case 1:
             visRestance(); //linje 112
             menu = 0;
             break;
          case 2:
             betalingsstatus(); //linje 84
             menu = 0;
             break;
          case 3:
             redigerKontigent(console); //linje 169
             menu = 0;
             break;
          case 4:
             kontingentPris(console); //linje 244
             menu = 0;
             break;
          case 0:
             this.menu = -1; //for at den ikke også hopper ud af ansats menu
             super.menu(); //menu hos ansat
             break;
          default:
             System.out.println("TAST VENLIGST ET NUMMER DER ER FREMVIST\n");
             subMenu();
       }
       break; //for kun at komme ud af dette loop
    }
}
```

Figur snippet 1 subMenu() metode

Denne metode er lavet til at kontrollere om inputtet er et tal som brugeren indtaster.

```
public int testConsoleInput(Scanner console){
   while(true){
      String input = console.next();

      try{
         return this.menu = Integer.parseInt(input);
      }catch(Exception e){
         System.out.println("UGYLDIGT INPUT");
         this.menu = -1; //sørger for at den ikke kan komme
                         //ind i andre cases hvis den fejler
         break;//hopper ud
      }
   }
   return 0;//dummy eftersom vi har en break som afslutter i catch blocken
            //derfor skal der være en return.
}
```

Kode: snippet 3 viser en metode der indeholder try/catch som er en form for error handling. Man kommer altid ind i while loopet eftersom denne er true by default. Try indeholder den "normale" kode – den kode som vi forventer går godt.
Hvis alt går godt, så kører den videre, men hvis ikke kører catch blokken
Catch hvis der kastes en Exception.
"Integer" er en klasse nedarvet fra "Numbers" klassen, der igen er nedarvet fra "Object" klassen.
Integer har en masse fields og metoder i sig, som vi gør brug af her.
Integer.parseInt(input) tager imod en String variabel og omdanne denne til int.

Metoden bliver genbrugt i adskillige sammenhænge, hvor det giver mening at bruge det. Et eksempel kan være alder der skal indtastes. Dette er for at undgå redundans.

```
setAlder(super.testConsoleInput(console));
```

Kode snippet 4 Metoden er implementeret i setter for tid. Det er i stedet for at have redundans og lave endnu en metode som vist i kode snippet 1 for at kontrollere input fra bruger.

JUNIOR ()

Af Rasmus Sadurskis

junior() metoden bruger en scanner der læser fra "konkurrenceSvoemmere.txt", for at få alle der er under 18 år printet ud. Måden det bliver gjort på, er via en while (alder.hasNextLine()), while loppet kører hele tiden indtil der ikke er flere linjer tekst filen.
For at læse en linje af gangen, bruges en String variable der gå ind og scanner hver token på den pågældende linje, og læser om personen er over eller under 18 år. Hvis ikke gå den videre til næste linje.

```
public void junior() throws Exception {
   Scanner alder = new Scanner(new File("konkurrenceSvoemmere.txt"));
      System.out.printf("%-12s%-15s%-10s\n", "FORNAVN", "EFTERNAVN", "ALDER");
      System.out.println("---------------------------------");

         while(alder.hasNextLine()){
            String line = alder.nextLine();
            Scanner token = new Scanner(line);

            String jFornavn = token.next();
            setFornavn(jFornavn);

            String jEfternavn = token.next();
            setEfternavn(jEfternavn);

            int jAlder = token.nextInt();
            setAlder(jAlder);

            if(jAlder < 18) {

               System.out.printf("%-12s%-15s%-10s", getFornavn(), getEfternavn(), getAlder());
               System.out.println();
            }
         }
      System.out.println();
}
```

Figur snippet 1 junior() metode

REDIGERKONTINGENT()

Af Martin Løseth Jensen

I redigerKontingent() kan man redigere et enkelt medlems betalingsstatus . Dette har vi gjort ved at gemme hvert enkelt token i et 2 dimensional array.
I count(…) metoden printer den alle medlemmer fra tekstfil med et tal, for at indikere hvilket medlem man vil vælge.

```
public void redigerKontigent(Scanner console)throws Exception{
   Scanner scanRest = new Scanner(new File("visRestance.txt"));
   String fileName = "visRestance.txt";
   count(fileName); //linje 157
   String redigerRestance[][] = new String[this.counter][3];

   while(scanRest.hasNext()){
      for(int i = 0; i < this.counter; i++){
         for(int j = 0; j < 3; j++){
            String item = scanRest.next();
            redigerRestance[i][j] = item;
         }
      }
   }
   int dummy = 0;
   while(dummy == 0){
      System.out.println("\nTAST NUMMER PAA MEDLEM DU OENSKER AT OPDATERE BETALING PAA");
      int num = console.nextInt(); //Check vaerdi?
      while (dummy == 0) {
         if (num <= 0 || num > this.counter) {
            System.out.println("\nTAST NUMMER PAA MEDLEM DU OENSKER AT OPDATERE BETALING PAA");
            num = console.nextInt(); //Check vaerdi?
         }
         else if (num > 0 && num <= this.counter){
               dummy = 1;
         }
      }
      System.out.println("HAR MEDLEMMET BETALT TAST 1\nMANGLER MEDLEMMET AT BETALE TAST 2");
      String input = console.next();
      if(input.equals("1")){
         setRestance("Betalt");
         redigerRestance[num-1][2] = getRestance();
         dummy = 1;
      }else if(input.equals("2")){
         setRestance("Restance");
         redigerRestance[num-1][2] = getRestance();
         dummy = 1;
      }else {
         System.out.println("UGYLDIGT INPUT\nTAST VENLIGST ET TAL DER ER FREMVIST");
         this.counter = 0;
         redigerKontigent(console); //linje 166
      }
   }
```

**Kode snippet 5 det i den blå kasse kontrollerer om den indtastet værdi er et nummer der findes blandt udprintet af medlemmerne.
I if/else if kan man redigere betalingsoplysningen.**

Derefter bliver det opdateret og gemt til filen igen. Dette gøres med en PrintStream, som overskriver ændringen på den pågældende plads i 2 dimensionalt array.

KONKLUSION

Af Casper Frost Andersen

DEL 1: ITO

Svømmeklubben Delfinen er en lille klub i vækst, og for at imødekomme tilslutningen af medlemmer, har den været nødt til at gøre visse overvejelser.
Blandt disse er, hvordan klubben skal forholde sig til andre lignende klubber, og hvordan klubben kan lokke nye medlemmer til.

Vi har konkluderet, at Delfinen skal forholde sig til nogle problemstillinger, qua. den eksponentielle interesse i klubben.
En manglende tilslutning fra nye medlemmer, manglende sponsorering fra lokale interessenter, samt manglende interesse fra tilskuere til stævner, kan alle være faldgruber for en lille virksomhed i vækst.
Dette kan løses ved at synliggøre sig i lokalsamfundet, udover den reklame der allerede er gjort fra kommunens side.

Delfinen har mulighed for at etablere sig på konkurrenceplan, hvis den nødvendige tilslutning fra konkurrencesvømmere findes, og der derved opnås en konkurrencedygtighed med de andre klubber.

DEL 2: SWD

Vi har, ud fra kundens krav, udarbejdet Use cases, der afspejler de funktioner som programmet skal indeholde.
alle Use cases har vi visualiseret i både skemaer og diagrammer, for at give kunden en fornemmelse af, hvordan systemet overfladisk kommer til at fungere:

For hver aktør (formand, kasserer og træner) har vi introduceret de nødvendige funktioner, for hvem det hhv. er muligt at oprette et nyt medlem, printe en liste over skyldnere, og for træneren muligt at registrere løbende træningstider, samt at printe en liste over de 5 dygtigste svømmere i hver disciplin.

Dette er gjort vha. Use cases, systemsekvens diagrammer, et sekvensdiagram, en domænemodel og et klassediagram.

DEL 3: SWC

Vi har i konstruktionen taget udgangspunkt i to superklasser: **medlem** og **ansat**.
Disse klasser indeholder hhv. alle indmeldte svømmere, og de tre aktører.

Begge disse klasser har tilknyttet flere underklasser, hvorfra de modtager objekter som løbende bliver føjet til lister.
Vi har i vores kode lavet metoder, som kan søge i disse lister efter betalingsstatus, disciplin, bedste træningstid m.v.

For at gøre dette, har vi benyttet os af 2-dimensionelle arrays og ArrayLists, for at kunne indeksere medlemmerne, og ændre i de dertilhørende tokens.

Vi kan derfor nu præsentere et fungerende program, der lever op til de krav som kunden har stillet.

DELKONKLUSION: ARBEJDSPROCESS

Af Casper Frost Andersen

Vi har i løbet af dette projekt benyttet os af en iterativ arbejdsproces, hvilket betyder, at vi hver gang en del af opgaven er blevet udviklet eller færdiggjort, reflekterer gruppen over, om det passer ind i opgaven, og i det overordnede billede.

Vi har arbejdet ud fra nogle grundprincipper, der lyder som følgende:

- Forventningsafstemning:
    - At sætte realistiske mål
    - At kende hinandens kompetencer
    - At uddelegere arbejdsopgaver i henhold til ovenstående (eksempelvis at en dygtig programmør arbejder sammen med en dygtig designer)
- Kommunikation – løbende møder hvor vi gør status over, hvad vi har lavet indtil videre, hvor vi er i opgaven, hvor vi vil hen, og om vi følger arbejdsplanen.
- Oplæg – løbende gennemgang af kode, diagrammer og layout, så alle har samme forståelse for forløbet.

Med henvisning til bilagene, fremgår det, at vi løbende har tilføjet arbejdsopgaver for at imødekomme vores mål. Hvis vi har været forud for faseplanen, har vi justeret derefter, og vice versa.

LITTERATURLISTE

Hele gruppen

Craig Larman: Applying UML and Patterns: An introduction to Object-Oriented Analysis and Design and Iterative Development – Third edition (2004)

Building Java Programs

ITO-bogen

www.stackoverflow.com

søgemaskinen Google.

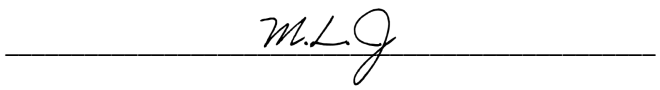
GLOSSARY

Af Martin Løseth Jensen

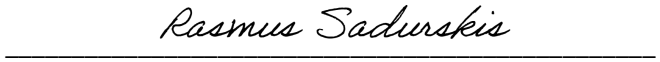
Navn	**Alias**	**Mærkat**	**Beskrivelse**
Formand		Aktør	Administrativ ansat
Kasserer		Aktør	En person der holder styr på betalinger fra medlem.
Træner	Traener	Aktør	En der er sammen med medlem og gør dem bedre.
Uses		Assosiation	Bruge.
Record		Assosiation	Gemmes
Pay		Assosiation	Betaling
Initialize		Assosiation	Opret.
Attends		Assosiation	Deltager
Results		Assosiation	Resultat
Contained-in		Assosiation	Indeholdes i.
Contain		Assosiation	Indeholder.
Pensionist		Object	Medlem over 60
Motionist		Object	En ikke-konkurrerende svømmer
Stævne	Staevne	Object	Begivenhed som de dygtigste konkurrencesvømmere deltager i
Kontingent		Object	Medlem skal betale dette for at forblive i klubben.
Medlem		Object	En person som er med i klubben.
Konkurrence		Object	En forsamling der konkurrer i forskellige disciplin.
Hold		Object	Mange antal medlem, som udgør et hold.
Aktivitetsform		Object	Den form for aktivitet man udøver
Disciplin		Object	En aktivitet, som der konkurreres i.

s

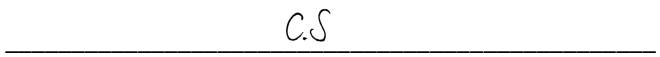
UNDERSKRIFTER:



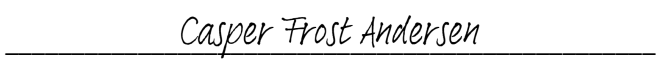
Martin Løseth Jensen: 07-12-2017



Rasmus Sadurskis: 07-12-2017



Christian Strunge: 07-12-2017



Casper Frost Andersen: 07-12-2017