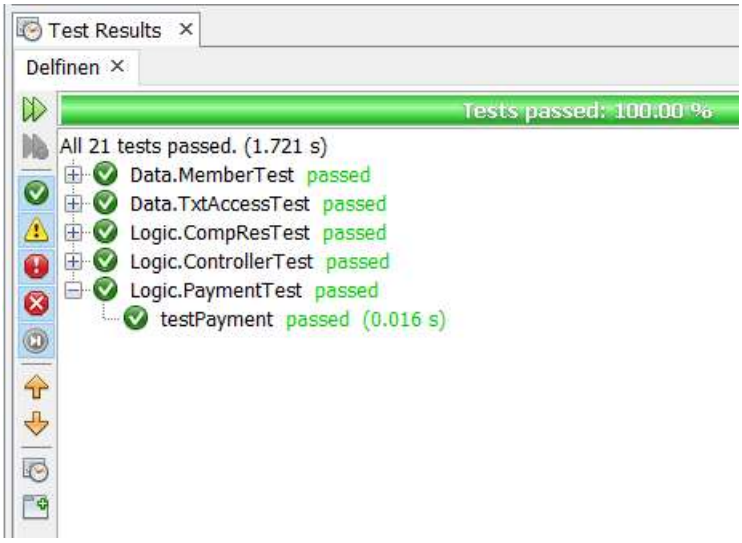
Gruppe 5. Hold A. Delfinen.

I Github kan der findes:
JavaDoc under dist mappen.
UnitTests
UML-Diagram
Use-Case Diagram
README

Automatiserede Unit Tests:
Alt i alt har vi lavet 21 automatiserede JUNIT tests af vores kode. Vi synes vi er kommet godt omkring alt det vigtige i koden, og testene hjalp i et par tilfælde med at gøre koden bedre.



Vi har kørt testene med et Java Plugin kaldet TikiOne JaCoCoverage. Det viser os at vi har god test dækning på de relevante klasser i projektet. En procentmæssig dækning på over 80% var målet.

Logic

Element	Missed Instructions	Cov.	Missed Branches	Cov.
Controller		69%		60%
Payment		63%		0%
Competition		58%		50%
Member		87%		50%
TrainingResults		92%		n/a
CompRes		100%		n/a
Delfinen		100%		n/a
Total	251 of 861	71%	40 of 72	44%

Det der trækker os ned i procent i Logic er Equals() metoder, og i Controller er det fordi vi har 3 forskellige topFive metoder, der har næsten helt ens kode, og vi mente kun det var

relevant at teste den ene. Derudover er alt der er relevant testet.
Vores UI er ikke unit testet, men vi har testet den grundigt manuelt, og fundet nogle fejl som vi desværre ikke har haft tid til at rette op på.

Fejlene består af:
- Hvis der ikke er 5 Stævne Resultater i en given kategori, fx Senior Crawl, og kun 2 fx, så viser den ingen af dem, og vi mener at den burde vise de 2.
- Der kommer ingen reaktion fra programmet hvis man giver den forkert input. Hvis vi havde haft mere tid ville vi have implementeret error messages når der opstod forkert input, som fx når formanden prøver at finde et member på et ID der ikke eksisterer i filerne.

Vi har anvendt GSON JSON til vores Members og Resultater i filerne, og det har sparret os for meget besvær og mulige errors i koden.
Her er vores code coverage af den største klasse i koden:

TxtAccess

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Method
findPayment(String)		79%		83%	1	4	5	17	0	
payment(String)		93%		100%	0	4	3	22	0	
initializePayment(Member)		89%		100%	0	2	3	16	0	
deleteID(String)		88%		100%	0	3	2	14	0	
deleteAllPayments(String)		86%		83%	1	4	3	15	0	
getAllIDs()		81%		100%	0	2	2	8	0	
getOneCompRes(String, Competition, String, int)		90%		50%	4	5	1	9	0	
setIDpath(String)		0%		n/a	1	1	2	2	1	
setMembersPath(String)		0%		n/a	1	1	2	2	1	
setPaymentPath(String)		0%		n/a	1	1	2	2	1	
setTrainingResultsPath(String)		0%		n/a	1	1	2	2	1	
setCompetitionsPath(String)		0%		n/a	1	1	2	2	1	
getHighestID()		91%		75%	1	3	2	11	0	
getMember(String)		92%		75%	1	3	1	6	0	
getMemberByName(String)		92%		75%	1	3	1	6	0	
resetAllFiles()		100%		n/a	0	1	0	27	0	
assignID(Member)		100%		100%	0	2	0	9	0	
deleteMember(String)		100%		100%	0	3	0	9	0	
compResToFile(CompRes)		100%		n/a	0	1	0	8	0	
TxtAccess()		100%		n/a	0	1	0	8	0	
getCompRes(String)		100%		n/a	0	1	0	6	0	
getMembers()		100%		n/a	0	1	0	5	0	
getTrainingResults()		100%		n/a	0	1	0	5	0	
getCompetitions()		100%		n/a	0	1	0	5	0	
juniorOrSenior(Member)		100%		100%	0	3	0	7	0	
setCompRes(String, List)		100%		n/a	0	1	0	2	0	
setMembers(List)		100%		n/a	0	1	0	2	0	
setTrainingResults(List)		100%		n/a	0	1	0	2	0	
setCompetition(List)		100%		n/a	0	1	0	2	0	
Total	85 of 920	91%	9 of 56	84%	14	57	33	231	5	2

Den er dækket godt, og det er vigtigt når den er så essentiel for programmet, som også tydeligt kan ses i vores UML model.

Exception handling:

changeMember(String id, boolean status,
String discipline)

MakeTrainingResult(String id, int distance, double time,
LocalDate date)

Hvis brugeren indtaster et ikke eksisterende ID i både ChangeMember og MakeTrainingResult metoderne, crasher systemet ikke men brugeren bliver ikke

gjort opmærksom på at han ikke har fundet et eksisterende medlem og den indtastede information bliver ikke brugt til noget. Til fremtidige ændringer skulle denne fejl fanges og brugeren skal gøres opmærksom på det forkert indtastede ID.

FindTopFiveId
FindTopFiveIdComp

Hvis brugeren beder om at se top 5 men der er mindre end 5 resultater registreret viser programmet ikke nogen resultater. Find TopFiveId burde fange denne fejl og vise den mængde der er uanset om det er under 5.

Tilgengæld bliver denne fejl fanget når man søger efter resultater ud fra et ID.

I vores textReader/writer har vi try catch til at håndtere problemer forbundet med at læse eller skrive til og fra vores filer.

Overordnet set har vi sat vores system op så der ikke er noget der kan få det til at crache. Men til videre arbejde vile vi informere vores brugere bedre om når de laver fejl så de er opmærksomme på forkert indberettet data.

Vedligeholdelsesvenlighed

Method Detail

makeMember

```
public void makeMember(java.lang.String name,
                       java.time.LocalDate birthDay,
                       java.lang.String city,
                       java.lang.String email,
                       java.lang.String number,
                       boolean status,
                       java.lang.String disciplin)
```

Make a Member.

Parameters:

`name` - Full name

`birthDay` - Birthday

`city` - City where they reside.

`email` - Email

`number` - Phone Number.

`status` - Status - Are they Active or Passive Members of Delfinen.

`disciplin` - Their Disciplin. "" empty String if they are not a competition swimmer. "Crawl", "BackCrawl", "Butterfly", "Breast" if they are a Competition Swimmer.

Vores controller indeholder metoder som vi har navngivet efter hvad deres primær funktion i vores program indebærer, så det er nemt for medprogrammører at kalde dem. F.eks. vores makeMember() metode som opretter et medlem med given parameter, navn, fødselsdag, by, disciplin også videre. Nedenunder ses det at vi ved hjælp af javadoc giver brugeren et godt indblik i hvordan de gører brug af medtoderne.

```java
/**
 * Finds top five Swimmers on a Team.
 *
 * For more comments, check method below. It basically does the same thing.
 *
 * @param distance The Distance you want to filter by.
 * @param Id ID of the Member
 * @return Returns Array of the best 5 Members ID for that Distance.
 */
public String[] FindTopFiveId(int distance, String Id) {
```

Vi har dog derudover en metode som har et navn som er en smule misledende da den gør det samme som en anden metode dog returnerer den et array med top fem svømmer og derved opfylder navngivning, og vores Javadoc forklarer udmærket brugen af metode med kommentarer.

Modulært design:

Vi har forsøgt at gøre vores metoder så fokuserede og nemme at overskue som muligt. Når man designer sit program er det vigtigt at hver metode har et navn der tydeligt beskriver funktionen.

Et af målene var at kunne ændre i en metode uden at ødelægge resten af programmet. Så GUI, Logic-laget og Data-laget kunne arbejde selvstændigt og blive opdateret uden at påvirke de andre lag. Interfaces kunne være gode til at sikre dette, fordi de garanterer at metodenavnene, returtyperne og parametrene er faste.

Det ville også gøre det muligt at erstatte en .txt baseret filsystem med en Database, uden at skulle ændre i hverken Logic-laget eller GUI-laget.

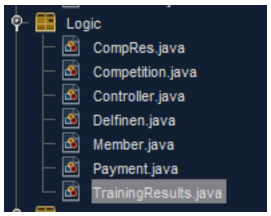
Genbrug (Modulært design, hjælpeklasser)

Modulært design:

Hjælpeklasser:

I Delfin har vi 5 hjælpeklasser som er under Logic pakken.

billede 1.



Hvilket er CompRes, Competition, Member, Payment og TrainningsResults fra billede 1.

CompRes

CompRes er en hjælpeklasse som bliver kaldt hvis træneren vil indskrive konkurrence resultater,

som tager imod et id, stævne og hvad plads de kom på i konkurrencen.

Competition

Competition bliver brugt til at oprette en konkurrence og brugt i CompRes til at registrer resultater.

Member

Member bliver brugt til at oprette et nyt medlem til klubben, samt giver dem et id-nummer.

Member inde holder metoder til at få et medlems alder og andre informationer.

Payment

Payment bliver brugt i henhold til at sende input data fra brugergrænsefladen til andre klasser.

Payment bliver også brugt til blandt andet at finde ud af hvor meget en person skylder til klubben.

TrainningsResults

TrainningsResults bliver kaldt når der bliver registreret nyt trænings resultater fra bruger grænseoverfladen. Den tager imod id'et af personen som har lavet et nyt resultat, samt distance, tid og dato.