Andreas, Jackie, Mathias, Mohamad

Titel: Delfinen

Gruppe: 6

Dato: 27-05-2021

Delfinen

Indledning

Vi har prøvet at følge MCV modellen.

Der er lagt tanker i hvor en scanner skulle ligge i forhold til MVC - efter nogen forsøg er scanneren endt i en controller. Argumentet for dette er, at når det bliver eventbaseret vil det være controller, der håndterer eventsne og vi har lidt legede med den tanke i starten af opgaven.

Vi har haft GRASP i tankerne - især low coupling og high cohesion i form af bl.a. ansvarsfordeling mellem klasser og hvilke klasser der kender hinanden. Ud fra vedlagte UML-diagram vil vi mene, at vi er kommet et langt stykke af vejen, men der er klart plads til forbedringer. Vi har nogen klasser med meget ansvar, som godt kunne være mindre klasser. Se evt. MemberController - selvom dens ansvar egentligt kun er medlemshåndtering kommer der lidt andet ind også.

Afgrænsning

Vi vil mene, at vi har fået løst alle use-cases, som vi har fundet i opgaven samt lidt ekstra.

Vi har ikke kunne finde nogen åbenlyse problemer i programmet. Det er rimeligt robust i forhold til input fra brugeren.

Data bliver gemt løbende, så hvis programmet skulle crashe, er det forhåbentligt med minimalt tab af data.

Medlemmer og Medlemskaber

I vores implementering er medlemmer og medlemskabet separeret. Et medlemskab indeholder data, om hvorvidt det er aktivt eller passivt - og om det er betalt eller ej. Grundtanken er, at man i teorien således godt kan være medlem uden at have et medlemskab - dog vil man altid have mindst et i vores implementering.

Når medlemskaber udløber, skal man aktivt sende et fornyelse anmodning, som fornyer og opretter en betalingsanmodning. Dette vil oprette et nyt medlemskab på medlemmet af samme type (aktivt/passivt), som vil stå som ubetalt.

Medlemmer burde ikke skulle slettes, da de er bundet på resultater. Resultater ønsker vi at beholde, selvom medlemmet ikke ønsker at være med i klubben længere. Vi har derfor tilføjet en anonymiseringsmetode for medlemmer, der ønsker at blive udmeldt.

Betaling

I forhold til betalings flowet benytter vi pt. den fil som vi opretter når vi laver fornyelser af medlemskab. Tanken er her, at betaling foregår eksternt og at det eksterne system sender et svar tilbage til os med godkendte betalinger. Herefter sætter vi medlemskabet til betalt.

Træning og Stævne

Vi har valgt at lave en enkelt klasse til dette, da begge indeholder samme information. Klassen hedder SwimEvent. Flowet for begge er at oprette et SwimEvent og derefter tilføje resultater til dem. Dette skyldes at vi vurderede at træninger ligesom stævner også foregår på bestemte datoer og tidspunkter.

Aldersgrænse

Alder skal være mellem 1 og 150. Dette er gjort for ikke at tilføje skøre datoer, hvor medlemmer kan være fx 1000 år gamle.

Abstract classes og interface

Vi har ikke benyttes os af selvimplementerede interface. Vi har implementeret Serializable og Comparable. Vi har ikke fundet et sted hvor vi syntes, at vi har haft behov for et hjemmelavet interface, da abstrakt klasser løste vores problem.

Use Cases Delfinen

Create Member

1. The system prompts the user a unique ID
   a. If nothing is entered, the system assigns a unique id
2. system prompts for name, everything is accepted
3. The system prompts for gender
4. The system prompts for a birthdate
   a. Gives junior or senior membership
5. The system prompts for an email and phone
6. The system prompts for competitiveness level (Casual or competitor)
7. The system assigns a creation date
8. The system updates the member's file/DB

View Member info

1. The user must choose which member he wants to view
   a. The User searches on ID or name
      - If the user searches on ID, and it exists, the user is directed to the "member view"
      - If there is more than one match, the matches are displayed, and the user must choose by picking a corresponding number
2. The system displays member information (Name, Age, etc.)
3. The system displays different options
   a. Update Member info
   b. Anonymize the member

Update Member info

1. The system displays "Update member" options
   a. Everything except ID is displayed
2. The user chooses what he/she wants to change
3. The user gets directed to the "Change" menu
4. The user is prompted if he/she is sure that they want to commit the change
   a. If yes, the information is updated and saved to the file
   b. If not, the user is redirected back to the "members options" menu

Update Payments

1. The user receives a list with paid memberships, and the list contains a Member ID
2. The user verifies that everything looks OK
   a. If everything looks OK, the user can accept, and the memberships are updated

b. If something is not OK, the file should be moved to a backup file with a prefix like "error" + timestamp, and this must be handled manually by the finance department

Income Predictions

1. The system displays the income pr. Month the next 12 months; it shows a total as well.

Expiring memberships

1. The system displays a list of passing memberships for the next 30 days
2. The user can choose to add or remove members from the list
3. The user gets prompted if he wishes to send a request for renewal of membership
   a. If yes, the system generates a file containing the member ID for the finance department
   b. If no, the user is directed to the main menu

Create Competition

1. The user gets prompted for competition Start date, name, and time

View Competition

1. Same logic as view member but for competitions

Register competition results

1. The user gets prompted for user ID
2. The user chooses discipline
   a. The user decides distance based on discipline
3. User enters time
4. The user gets prompted if he wishes to add another time and field for the same user
   a. If yes, repeat 2-4 from above
   b. If no, the user gets redirected to the competition's menu

Display top 5

1. User chooses discipline
   a. User chooses distance
2. The system displays the top 5 within the different splits