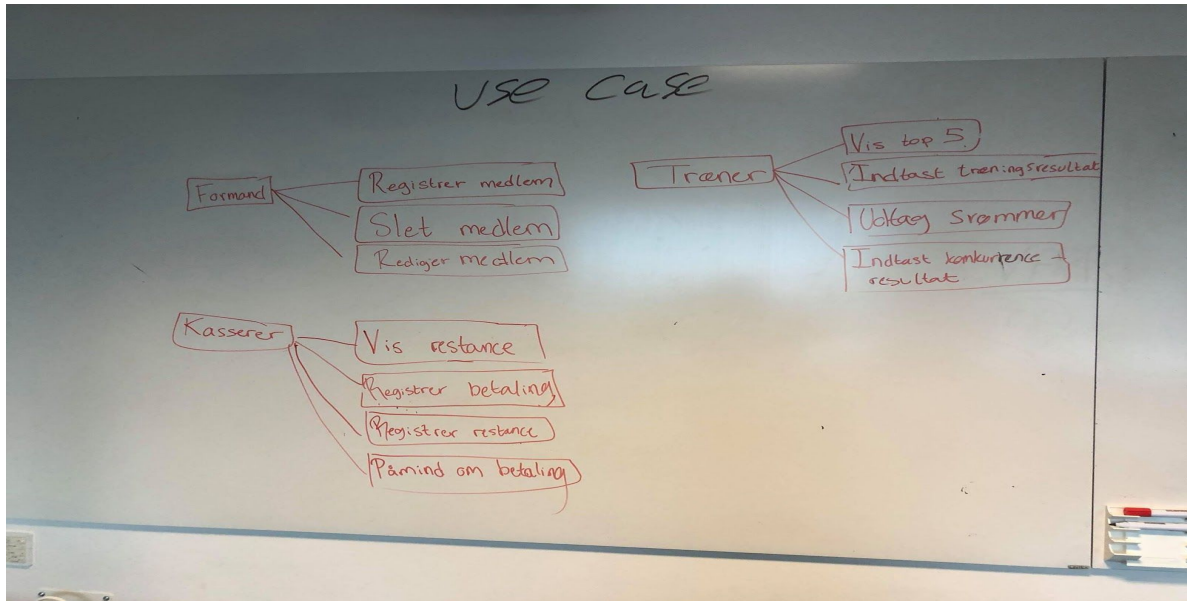


## Programmets funktionaliteter

Alle funktionaliteter er blevet implementeret i programmet på nær “slet medlem” og “påmind om betaling” (af anbefaling fra vores lærer).

### Registrer og rediger medlem (Formand)



Til venstre registrerer formanden et nyt medlem. Vi har lavet “Discipline” som enum i vores kode, så man kun kan vælge de 4 discipliner der er angivet på skærbilledet. Man skal vælge aktivitetsform og holdtype, og i tilfælde af at der ikke vælges hold, disciplin, eller aktivitet, håndterer vi det med Exception handling.

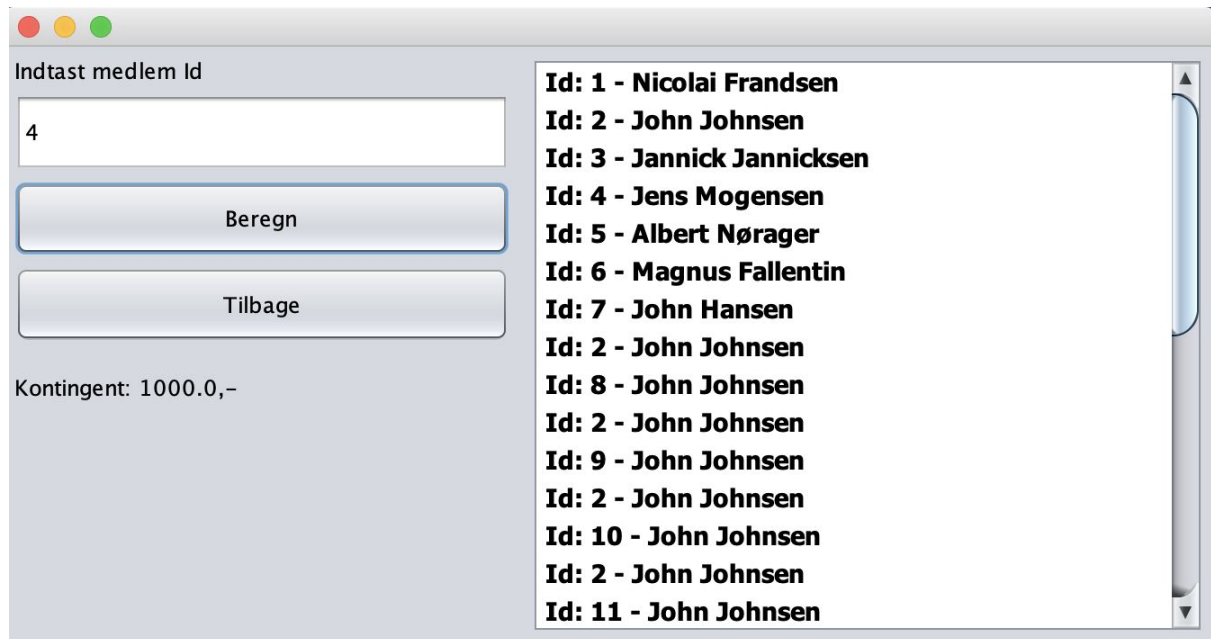
Til højre redigerer vi et medlem. Da vi på forhånd har medlemmer i tekstfiler, henter vi et givent medlems oplysninger ved at indtaste medlemmets ID. Et medlem redigeres i editMember metoden, ved at alle oplysninger sættes til at være tomme, og derefter kan man tildele medlemmet nye oplysninger via nyt input.

Fornavn	Efternavn	Disciplin
<input type="text"/>	<input type="text"/>	<input type="radio"/> Crawl <input type="radio"/> Brystsvømning <input type="radio"/> Butterfly <input type="radio"/> Rygcrawl
Fødselsdag		
<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value=""/>
Aktivitetsform	Holdtype	
<input type="radio"/> Aktivt medlemsskab <input type="radio"/> Passivt medlemsskab	<input type="radio"/> Motionist <input type="radio"/> Konkurrence	<input type="button" value="Tilbage"/> <input type="button" value="Registrer"/>

Indtast medlemmets ID		
<input type="text"/>		
Fornavn	Efternavn	Disciplin
<input type="text"/>	<input type="text"/>	<input type="radio"/> Crawl <input type="radio"/> Brystsvømning <input type="radio"/> Butterfly <input type="radio"/> Rygcrawl
Fødselsdag		
<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1900"/>
Aktivitetsform	Holdtype	
<input type="radio"/> Aktivt medlemsskab <input type="radio"/> Passivt medlemsskab	<input type="radio"/> Motionist <input type="radio"/> Konkurrence	<input type="button" value="Tilbage"/> <input type="button" value="Ændr"/>

## Beregn kontingent (Kasserer)

Her beregnes kontingent for det givne medlem. Vi har i kildekoden metoder til at beregne den korrekte pris afhængig af medlemmets alder og hvorvidt medlemmet er aktivt eller passivt.



Indtast medlem Id

4

Beregn

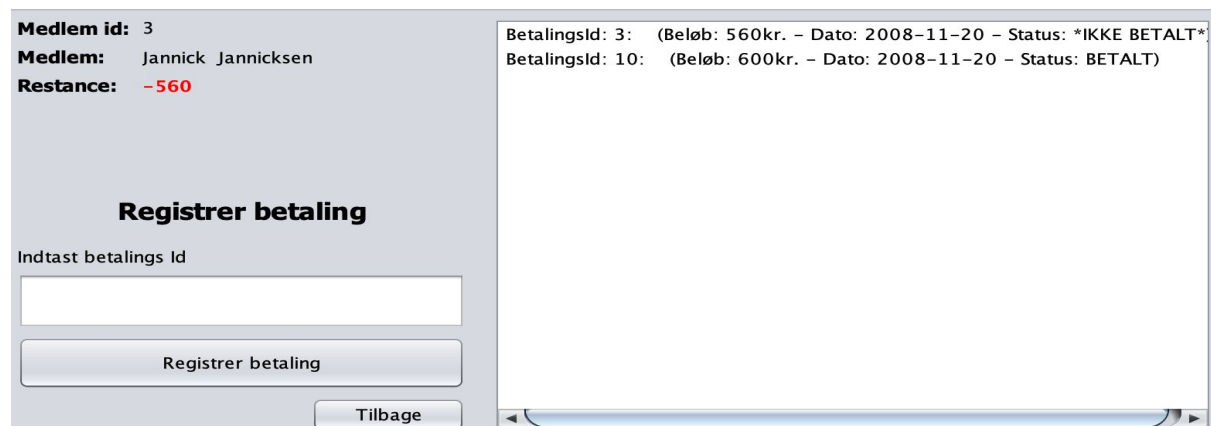
Tilbage

Kontingent: 1000.0,-

- Id: 1 - Nicolai Frandsen**
- Id: 2 - John Johnsen**
- Id: 3 - Jannick Jannicksen**
- Id: 4 - Jens Mogensen**
- Id: 5 - Albert Nørager**
- Id: 6 - Magnus Fallentin**
- Id: 7 - John Hansen**
- Id: 2 - John Johnsen**
- Id: 8 - John Johnsen**
- Id: 2 - John Johnsen**
- Id: 9 - John Johnsen**
- Id: 2 - John Johnsen**
- Id: 10 - John Johnsen**
- Id: 2 - John Johnsen**
- Id: 11 - John Johnsen**

## Vis restance og registrer betaling

Her vises medlemmets ID, navn, og restance. Hvis der er en restance, er status sat til ikke betalt. Hvis der så skal registreres en betaling fra det givne medlem, indtast betalingsid, som er tilfældigt, i feltet.



**Medlem id:** 3  
**Medlem:** Jannick Jannicksen  
**Restance:** -560

**Registrer betaling**

Indtast betalings Id

Registrer betaling

Tilbage

BetalingsId: 3: (Beløb: 560kr. - Dato: 2008-11-20 - Status: \*IKKE BETALT\*)  
BetalingsId: 10: (Beløb: 600kr. - Dato: 2008-11-20 - Status: BETALT)

## Top 5 & udtag svømmer (Træner)

Til venstre kan træneren se en oversigt over top 5 inden for en given svømmedisciplin. Medlemmet kendes på ID og navn. Til højre kan træneren udtage svømmere på enten junior eller seniorholdet.

**Top 5 Crawl**

Id: 5 – Albert Nørager – tid: 22.59  
 Id: 2 – Kurt Kurtsen – tid: 33.56  
 Id: 4 – Jens Mogensen – tid: 34.0  
 Id: 1 – Nicolai Frandsen – tid: 40.56  
 Id: 6 – Magnus Fallentin – tid: 43.0

**Tilbage**

Indtast medlemmets ID

**Udtag**

**Tilbage**

**Senior:**

Id: 2 - Kurt Kurtsen  
 Id: 5 - Albert Nørager  
 Id: 6 - Magnus Fallentin  
 Id: 1 - Nicolai Frandsen  
 Id: 7 - John Hansen

**Junior:**

Id: 4 - Jens Mogensen

## Træningsresultat og konkurrenceresultat

Til venstre registreres træningsresultater for medlemmer i en af disciplinerne. Til højre indtastes resultater i konkurrencer. Her angiver træneren svømmerens placering, tid, og hvilket stævne der er tale om.

**Medlem: 2**

Vælg disciplin:

☒ Crawl  
☐ Brystsvømning  
☐ Butterfly  
☐ Rygcrawl

Indtast resultat:  
 Tid (pr 50m)

Vælg dato:

1 1 1900

**Registrer resultat**

**Tilbage**

**Medlem: 5**

Stævne

Placering

Tid (pr 50 m)

Dato:

1 1 1900

**Registrer**

**Tilbage**

Nedenfor er et eksempel på exception handling. Her sikrer vi, at der skal vælges en disciplin, hvis medlemmet skal være konkurrencesvømmer, ellers kommer der en fejlmeddelelse om, at man skal vælge en disciplin. Hvis det er en motionist, kommer der en fejlmeddelelse, hvis man vælger en disciplin, da motionister ikke skal vælge en disciplin.

Nederst kaster vi en `IllegalArgumentException` hvis der indtastes et tomt resultat, og en `Exception`, hvis der indtastes negative værdier.

```

try {
    if (crawl) {
        dis = CRAWL;
    }
    if (bryst) {
        dis = BRYST;
    }
    if (butterfly) {
        dis = BUTTERFLY;
    }
    if (rygcrawl) {
        dis = RYGCRAWL;
    }

    if (!crawl && !bryst && !butterfly && !rygcrawl && competitive == true) {
        throw new Exception("Vælg venligst din disciplin");
    }
    if ((crawl || bryst || butterfly || rygcrawl) && motionist == true) {
        throw new Exception("Motionister må ikke vælge en disciplin");
    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(this, e.getMessage());
}

try {
    time = (Double) Double.parseDouble(this.jTextField1.getText());

    if (this.jTextField1 == null) {
        throw new IllegalArgumentException("Int not found");
    }

    if (time < 0) {
        throw new Exception("Negative input inserted");
    }
}

```

```

@Test
public void getMember() {
    int id = PresidentFile.getLatestId();
    Member actual = ctrl.createMember(id, firstName, lastName, activity, age, birthDate, dis, mteam);
    ctrl.printMember(actual);
    assertNotNull(actual);
    assertEquals(actual.getFirstName(), ctrl.getMember(id + 1).getFirstName());
    assertEquals(actual.getLastName(), ctrl.getMember(id + 1).getLastName());
    assertEquals(actual.getId(), ctrl.getMember(id + 1).getId());
}

@Test
public void getMembers() {
    assertNotNull(ctrl.getMember(1));
    assertEquals(1200, ctrl.getContingent(1), 0);

    assertNotNull(ctrl.getMember(2));
    assertEquals(500, ctrl.getContingent(2), 0);

    assertNotNull(ctrl.getMember(5));
    assertEquals(1600, ctrl.getContingent(5), 0);

    assertNotNull(ctrl.getMember(4));
    assertEquals(1000, ctrl.getContingent(4), 0);
}

@Test
public void printTrainingResult() {
    ctrl.printTrainingResult("/Users/sinanjasar/Desktop/delfinentxt/BestCrawlResults.txt", "/Users/sinanjasar/Desktop/");

    Scanner x = null;
    try {


```

Test Results

delfinen.test.ControllerTest

Tests passed: 100,00 %

All 7 tests passed. (0,36 s)

▶  delfinen.test.ControllerTest passed

Som det fremgår af det ovenstående har vi lavet Junit Test på de mest relevante metoder.

Vi har ligeledes lavet JavaDoc og her vedlagt et udsnit med eksempler.

```
/**
 * Retrieves Member id
 * @param id
 * @return the member with the given id
 */
public static Member getMember(int id) {
    return CoachFile.getMember(id);
}

/**
 * Retrieves the filepath the method goes through
 * @param filePath
 * @return ArrayList including all members from the filepath
 */
public ArrayList<Member> getMembers(String filePath) {
    return CoachFile.getMembers(filePath);
}

/**
 * Retrieves id of the member you are looking for
 * @param id
 * @return the contingent of member with the given id
 */
public double getContingent(int id){
    Member m = getMember(id);
    int age = m.getAge();
    String status = m.getActivity();

    if(status.equals("active")){
        if(age < 18) return 1000;
        if(age >= 18 && age <= 60) return 1600;
        if(age > 60) return 1600 * 0.75;
    }
}
```