1DV404 – HT2014
Iterativ mjukvaruutveckling
Laboration 4 - Projekt

av Da-Sing Trinh
dt222cc@student.lnu.se

Gymnastiktävlingssystem

Iterationsplanering

Målet med dokumentet är att definera de funktionalitet som ska implementeras. Målet med iterationerna är att implementera så mycket funktionalitet kring Gymnastiktävlingssystemet som möjligt.

Iteration 1

Vad:

- Implementering av registrering av användare/gymnast

Mål:

- Förmågan att registrera sig och förmågan för systemet att hantera det
  - Klass Person
  - Klass Validate

Tid: 9 timmar

Iteration 2

Vad:

- Implementering av registrering av lag

Mål:

- Förmågan att registrera sitt lag och förmågan för systemet att hantera det
  - Klass Team
  - Klass League

Tid: 10 timmar

Iteration 3

Vad:

- Implementering av registera sig till lag

Mål:

- Förmågan att registrera sig till ett lag och förmågan för systemet att hantera det
  - Fortsätt med klassen Team

Delmål:

- Snygga till all källkod, göra förbättringar om jag kommer på något

Tid: 11 timmar

Kort motivering/reflektion

Min motivering till varför iterationsplaneringen är så kort som den är för jag tyckte att de behövdes inte var så pass långa + att jag hade försökt att hålla den kort, liksom antingen i en sida eller två sidor så att det ser prydligt ut. Man skulle ju ha "vad", "mål" och "tid" om jag hade förstått rätt.

Iteration 1

Bakgrund:

Implementering av registrering av ny användare/gymnast.

Planering:

Vad:	Uppgift:	Tid:
Krav	Analysera användningsfall, Specificera krav	30 min 30 min
Design/Implementation	Identifiera klasser, Specificera klasser, Implementation	45 min 1 h 2 h
Test	Test design, Test specification, Test implementation, Test exekvering,	45 min 1 h 1 h 30 min
Reflektion	Reflektera	1 h
		9 h

Flöden:

Initiering: Användaren klickar på registrera ny användare/gymnast

Primärt flöde:

1. Sidan för registrering av ny användare presenteras
2. Aktören fyller i fält och trycker sedan på "Registrera" knappen
3. Verifierar data
4. Registrera data och lämplig meddelande presenteras för aktören som indikerar att registreringen är klart

Alternativ flöde:

Om i steg 3 i primära flödet misslyckar verifieringen...

1. Presenteras ett meddelande som indikerar att registreringen misslyckades
2. Vidare till steg 2 i primära flödet.

Krav:

- Kontrollera att de inmatade värden är korrekta med regex
  - E-post
    - Kontrollera om användarens e-post är redan registrerad
  - Ålder
  - Lösenord (två fält, om båda fälten är samma)

Design:

Klass: Person	Klass: Validate
firstName: String, lastName: String, email: String, password: String	email: String, pw1: String, pw2: String isVerified: bool
	validateData(data): isVerified validateEmail(email): isVerified validatePW(pw1, pw2): isVerified

Test: Klassen Person och Validate

Test	Metod	Testfall	Förväntad resultat
#1	Lägg till ny person	Förnamn: Sing Efternamn: Trinh E-post: dt222cc@student.lnu.se Lösenord: 1234	true
#2	Validering av data, korrekt format	E-post: dt222cc@student.lnu.se Lösenord 1: 1234 Lösenord 2: 1234	true
#3	Validering av data, fel email	1. dt222ccstudent.lnu.se 2. @gmail.com 3. dt222cc@ 4. ""	false
#4	Validering av data, fel lösenord	1. 1234, 2345 2. 1234, "" 3. "", 1234 4. "", ""	false

Testdata:

Test	Testansvarig	Metod	Förväntad resultat	Verklig resultat	Godkänt/ Underkänt
#1	dt222cc	Lägg till ny person	true	true	Godkänt
#2	dt222cc	Validering av data, korrekt format	true	true	Godkänt
#3	dt222cc	Validering av data, fel email	false	false	Godkänt
#4	dt222cc	Validering av data, fel lösenord	false	false	Godkänt

Källkod:

Källkod finns i iterationsmappen.
https://github.com/dt222cc/1dv404-dt222cc-laborationer/tree/master/Laboration%204/iteration1

Iteration 2

Bakgrund:

Implementering av registrering av nytt lag

Planering:

Vad:	Uppgift:	Tid:
Krav	Analysera användningsfall, Specificera	30 min 30 min
Design/Implementation	Identifiera klasser, Specificera klasser, Implementation	45 min 45 min 2 h
Test	Test design, Test specification, Test implementation, Test exekvering,	45 min 45 min 2 h 30 min
Reflektion	Reflektera	1 h 30 min
Extra	Revidera	1 h
		10 h

Flöden:

Initiering: Användaren klickar på registrera lag
Pre-villkor: Användaren är inloggad

Primärt flöde:

1. Sidan för registrering av nytt lag presenteras
2. Aktören fyller i fält och trycker sedan på "Registrera" knappen
3. Verifierar data
4. Laget registreras och lämplig meddelande presenteras för aktören som indikerar att registreringen är klart

Alternativ flöde:

Om i steg 3 i primära flödet misslyckar verifieringen...

1. Presenteras ett meddelande som indikerar att registreringen misslyckades.
2. Vidare till steg 2 i primära flödet

Post-villkor: Laget registrerad

Krav:

- Kontrollera att de inmatade värden är korrekta med t.ex regex
  - Namn (tillgänglig, tom)

Design:

Klass: League	Klass: Team
teams: [*]	teamName: String teamLeader: Person
addTeam() removeTeam()	

Test: Klassen Team och League

Test	Metod	Testfall	Förväntad resultat
#1	Lägg till nytt lag, tillgänglig lag namn	Lag namn: YaY IF (String) Lag ledare: Sing Trinh (Person)	Laget registrerad
#2	Lägg till nytt lag, upptaget lag namn	Lag namn: YaW IF (String) Lag ledare: Sing Trinh (Person)	Fel meddelande
#3	Lägg till nytt lag, tom namn	Lag namn: "" (String/Undefined) Lag ledare: Sing Trinh (Person)	Fel meddelande
#4	Ta bort lag	Lag namn: YaW IF (String)	Laget borttaget

Testdata:

Test	Testansvarig	Metod	Förväntad resultat	Verklig resultat	Godkänt/ Underkänt
#1	dt222cc	Lägg till nytt lag, tillgänglig lag namn	Laget registrerad	Laget registrerad	Godkänt
#2	dt222cc	Lägg till nytt lag, upptaget lag namn	Fel meddelande	Fel meddelande	Godkänt
#3	dt222cc	Lägg till nytt lag, tom namn	Fel meddelande	Fel meddelande	Godkänt
#4	dt222cc	Ta bort lag	Laget borttaget	Laget borttaget	Godkänt

Källkod:

Källkod finns i iterationsmappen.
https://github.com/dt222cc/1dv404-dt222cc-laborationer/tree/master/Laboration%204/iteration2

Kort motivering/reflektion:

Efter att ha arbetat med klassen team/league så fick jag göra ett par ändringar. Liksom det blir inte alltid så som man hade tänkt sig. När man arbetar så lägger man märke på saker och ting som inte behövs eller annat som man behöver ha med. Detta innebar att jag fick ta bort en del funktionalitet som inte behövdes. Klasserna blev mindre och jag la märke på att jag kunde göra om det jag skulle göra i iteration 3 lite. Med iteration 3 så skulle jag lägga till funtionalitetet att lägga till gymnaster till ett lag. Jag la märke på att jag kunde jobba "vidare" med det i klassen team.

En annan grej är att eftersom omfattningen är inte så stor så blir iterationerna kortare än förväntat, transitionen mellan iterationerna är nu mer om jag blir klar så går jag vidare nästa iteration.

Iteration 3

Bakgrund:
Implementering av registrera gymnast till lag.

Planering:

Vad:	Uppgift:	Tid:
Krav	Analysera användningsfall, Specificera	30 min 30 min
Design/Implementation	Identifiera klasser, Specificera klasser, Implementation	45 min 45 min 2 h
Test	Test design, Test specification, Test implementation, Test exekvering,	45 min 45 min 2 h 30 min
Revidering	Revidera	1 h
Reflektion	Reflektera	1 h 30 min
		11 h

Flöden:
Initiering: Lagledare/administratör väljer att registrera ny medlem.
Pre-villkor: Lagledare/administratör har kollat igenom en lista med personer som har skickat in en intresseanmälan. Aktören är dessutom inloggad.

Primärt flöde:
1. Aktören väljer att han/hon vill registrera ny medlem.
2. Aktören fyller i fält från en lista (om den nya medlemens information).
3. Aktören skickar in datan.
4. En ny medlem registreras och lämplig meddelande presenteras för aktören som indikerar att registreringen är klart.

Alternativ flöde:
Om i steg 3 i primära flödet misslyckar verifieringen...
1. Presenteras ett meddelande som indikerar att registreringen misslyckades, med anledningen till varför det misslyckades.
2. Vidare till steg 2 i primära flödet

Post-villkor: Ny medlem registrerad till laget.

Krav:
- Bara registrerade gymnaster kan skicka intresseanmälan till lag. Innebär att alla medlemar i ett lag är registrerade gymnaster.
- Kontrollera om gymnastern är redan registrerad.

Design:

Klass: League	Klass: CreateObj	Klass: Validate

Teams: [team,team]
gymnasts: [person,person]
	person: {
	pcn: String, pw: String
	fName: String, lName: String,
	email: String,
	}
	team: {
	teamName: String,
	teamLeader: String,
	members: [member, member]
	}
	member: {
	pcn: String, fName: String,
	lName: String, email: String,
	}
		result: String

addTeam() removeTeam() addGymnast() removeGymnast() addMember() removeMember()	addGymnast() addTeam() addMember()	validatePerson() validatePCN() validateEmail() validatePW()

Tester:

Test	Metod	Testfall	Förväntad resultat
#1	addMember(), ny medlem	PCN: 111122334444 Lag: YaW IF	Ny medlem registrerad.
#2	addMember(), redan medlem	PCN: 199302271078 Lag: YaW IF	Personen är redan en medlem.
#3	removeMember(), avregistrera medlem	PCN: 199302271078 Lag: YaW IF	Medlem, avregistrerad.
#4	addGymnast(), registrera ny person	199312241078, Da-Sing, Trinh, dasing.trinh1@gmail.com 1111, 1111	Personen registrerades utan problem.
#5	addGymnast(), redan registrerad	199302271078, Da-Sing, Trinh, dasing.trinh1@gmail.com 1111, 1111	Personnummer, redan registrerad!
#6	addGymnast, tomma fälter	PCN, Förnamn, Efternamn, Epost, Pw1, Pw2, Pw1&2	Tom fält!

Testdata:

Test	Metod	Förväntad resultat	Verklig resultat	Godkänt/ Underkänt
#1	addMember(), ny medlem	Ny medlem registrerad.	Ny medlem registrerad.	Godkänt
#2	addMember(), redan medlem	Personen är redan en medlem.	Personen är redan en medlem.	Godkänt
#3	removeMember(), avregistrera medlem	Medlem, avregistrerad	Medlem, avregistrerad	Godkänt
#4	addGymnast(), registrera ny person	Personen registrerades utan problem.	Personen registrerades utan problem.	Godkänt
#5	addGymnast(), redan registrerad	Personnummer, redan registrerad!	Personnummer, redan registrerad!	Godkänt
#6	addGymnast(), tomma fälter	Tom fält! (för alla tester)	PCN, Pw1, Pw2 och Pw1&2: Tom fält! Förnamn & Efternamn: Personen registrerades utan problem. E-post: E-post adress, fel format!	Underkänt (fixa namn, justera epost i testkoden)

Testansvarig: Da-Sing Trinh, dt222cc@student.lnu.se

**** Update: Test #6 åtgärdad.**
La till validateName() i klassen Validate och justerade vilka argument som skickas (+ fName, lName)

Källkod:
Källkod finns i iterationsmappen.
https://github.com/dt222cc/1dv404-dt222cc-laborationer/tree/master/Laboration%204/iteration3

Reflektioner:
Det saknas lite små funktionaliteter som felhantering (personnummer, namn: tom fält).
Det är väl acceptabelt med det? :D
Jag tyckte att det var roligt att koda och testa koden, så som jag hade gjort (dock kanske inte på rätt sätt). Hursomhelt, detta gällde då när man hade valt/hittat något som inte är allt för krångligt att arbeta med. Jag hade kanske nämnt det tidigare att det tog en del tid för att hitta något man ska arbeta med som man är bekväm med.

Det var inte så skoj med att behöva dokumentera allt. Det är något som jag håller på att jobba med. Jag hoppas att det är inte så mycket fel med mitt "projekt". Jag har gjort några korta reflektionen/motiveringar tidigare, efter olika moment i projektet. Mer reflektioner på nästa sida.

Mer reflektioner

Till att börja med, tiden. Jag planerade in 9 timmar för den första iterationen, 10 för den andra och sedan 11 timmar för den tredje. Jag tänkte först att jag börjar med ett enklare moment och sedan ökar jag svårighets graden lite grann.
Det blev inte precis så som jag hade tänkt mig, det tog mest tid i början. Sedan gick det smidigare (med iteration 2).
Anledningen till att tiderna var relativt hela var pga små "reflektioner/funderingar" för att jämna ut tiden. Liksom om man hade arbetat i 36 minuter, försök fylla på till 40 minuter. Om jag kommer över 40 minuter, kör på 50 minuter, o.s.v
Sedan så finns det faktorer där man behövde kolla up om man kunde göra så som man hade tänkt, kolla up referenser och så.
I de flesta fall gick det utan problem. Denna tid uteslöt jag från tidsloggen. Dessutom så är tidsloggen inte helt precis/noggrann (pga pauser, glömma kolla tiden, etc).

När det gällde testningen så fick jag revidera ett par gånger i början för att få koden att funka, lägga till ett par saker och så med "return". Sedan när man fick till hur man skulle göra testningen gick det smidigare för nästa iteration.
Det gick alltså bättre nu än i laboration 3. Med mer erfarenhet och kunskap. Ett bättre flöde blev det. Det finns dock fortfarande områden man inte har så bra koll på utan man fick använda det man kunde och köra på ett vis som man var bekväm med helt enkelt.