Iterationsplanering 1DV404

Benny Carlsson bc222az

Bakgrund och ProblemBeskrivning

Gymnastiktävlingssystemet
Systemet är tänkt att ha ett antal tävlingar för en säsong och lag sa kunna registrera sig för att sedan anmäla sig till tävlingar. En sekreterare ska sedan efter varje tävling kunna lägga in poäng för varje lag.
Det ska sedan gå att se resultat för säsongen.

Användare

Lagledare - Registrerar lag
Sekreterare - Tilldelar poäng för varje lag i varje tävling
Åskådare - Person som vill se resultaten

Slutmål

En applikation där man skall kunna:
-Registrera ett lag
-Logga in som sekreterare
-Logga in som ett registrerat lag
-Anmäla sitt registrerade lag till en tävling
-Ge poäng till ett lag som sekreterare
-Se poäng för säsongen
-Se poäng för varje lag
-Se poäng för varje tävling

Sammanlagt ska hela genomförandet ta 32 timmar.
Varje iterationerna planeras ta ca 8 timmar alltså sammanlagt 24timmars implementering.

Delmål för varje iteration

Iteration 1:
Uppskattad tid: 10,5h
Design och implementation:
Registrering - Man skall kunna registrera ett lag. Registrerat lag lagras i en textfil.
Inloggning - Man skall kunna logga in som sekreterare eller som ett registrerat lag.

Iteration 2:
Uppskattad tid: 10h
Design och implementation:
Anmäl lag - Man skall kunna anmäla ett lag till alla tävlingar som kommer finnas tillgängliga under säsongens gång.
Ange poäng som sekreterare - Som sekreterare skall man kunna ge poäng till diverse lag för varje tävling. poängen sparas i en textfil.

Iteration 3:
Uppskattad tid: 5h
Design och implementation:
Visa resultat -
Resultat för enskild tävling
Resultat för enskilt lag
Resultat för hela säsongen

Iteration 1

Tidsplanering

Ett lag ska kunna registrera sig med ett lagnamn: 2h
Verklig tid: 2h
Lagnamn som registrerats ska sparas i en textfil: 2h
Verklig tid: 2.5h
Lagnamn som finns ska inte kunna registreras igen: 1h
Verklig tid: 1h
Sekreterare ska kunna logga in som sekreterare 1.5h
Verklig tid 0.5h
Registrerat lag ska kunna logga in med sitt lagnamn 1.5h
Verklig tid: 1h
Testimplementation 2h
Verklig tid 3.5h
Sammanlagt: 10h
Verklig Sammanlagd tid: 10.5h

Kravanalys

Ett lag ska kunna registrera sig
Ett lag ska kunna logga in
En sekreterare ska kunna logga in
Registrerat lag ska sparas i ett textdokument
Registrerat lagnamn ska inte kunna registreras igen

Testning

Ett lagnamn som redan är registrerad ska inte kunna registreras igen
Ett lagnamn får inte vara “sekreterare” eller “0”
Ett lagnamn som **inte** redan finns ska gå igenom registreringen
Inloggning som “sekreterare” ska gå igenom
Inloggning med existerande lagnamn ska gå igenom
Inloggning med ej registrerad lagnamn ska inte gå igenom

Design och implementation

Två klasser skapades för att ta hand om kraven. **Login** och **Register**.
Login klassen består av följande metoder
tryLoginTeam() som anropas när ett lag försöker logga in.

tryLoginSecretary() som anropas när sekreteraren försöker logga in.
loginvalidator() som anropas för att kolla så att ingen föröker logga in med “0” eller “null”.
secretaryLoggedIn() anropas för att kolla om sekreteraren är inloggad.
teamLoggedIn() anropas för att kolla om ett lag är inloggad.
teamNameLoggedIn() anropas för att kolla vilket lag som är inloggad.

Klassen **FileHandler** har ett flertal metoder som tar hand om allt som har med textfilerna där all information sparas att göra.

Reflektion

Planerad tid: 10h
Faktiskt tid: 10.5h
Efter lagt ner 10.5 timmar på första iterationen har jag tyvärr inte lyckats åstadkomma det jag förväntade mig eftersom jag stötte på några motgångar.
Att komma igång med testning i eclipse tog längre tid än förväntat och efter handledning insåg jag att jag lagt ner onödig tid på “view” delen av uppgiften då en meny eller fungerande exekverande applikation inte var nödvändigt för projektet utan jag ska lägga om fokus på själva applikationens funktioner och sedan göra testmetoder för dessa.
Detta har lett till att efter första iterationen måste jag strukturera och lägga om min kod för att anpassa det efter den nya tillkomna informationen.
Jag valde att avsluta iteration 1 eftersom jag uppnåt tidschemat men att föra över kvarstående uppgifter och krav till iteration 2 och senare försöka ta igen för förlorad tid.

Uppnådda Mål:

Ett lag ska kunna registrera sig
Ett lag kan registrera sig.
Ett lag ska kunna logga in
Ett registrerat lag kan logga in med sitt lagnamn
En sekreterare ska kunna logga in
Sekreterare kan logga in men användarnamnet “sekreterare”
Registrerat lag ska sparas i ett textdokument
När ett lag registreras sparas detta permanent i ett textdokument
Registrerat lagnamn ska inte kunna registreras igen
Ett redan registrerat lagnamn kan inte registreras igen även upptagna namn som “0” eller “sekreterare” nekas registrering.

Ej uppnådda Mål:

Testningskraven skickas över till Iteration 2.
Struktur av koden skriven i iteration 1 skickas över till iteration 2 för att bättre anpassas till projektet, dock är all funktionalitet i koden implementerad.

Testning:

Test1 - Test Register - Försök registrera användare
Test input 1: ”sekreterare” **Förväntat resultat:** False **Faktiskt resultat:** False
Test input 2: ”0” **Förväntat resultat:** False **Faktiskt resultat:** False
Test input 3: ”lag1” **Förväntat resultat:** True **Faktiskt resultat:** True
Test input 4: ” ” **Förväntat resultat:** False **Faktiskt resultat:** False
Test input 5: ”” **Förväntat resultat:** False **Faktiskt resultat:** False

Iteration 2

Tidsplanering

Strukturera om kod: 1.5h
Verklig tid: 1h
Hårdkorda in säsongens tävlingar 0,5h
Verklig tid: 0,5h
Anmäla lag till en tävling 2h
Verklig tid 3h
Sekreterare ska få upp alla tävlingar 0.5h
Verklig tid 0.5h
Sekreterare ska få upp alla lag som är anmälda till tävlingar 1h
Verklig tid 1h
Sekreterare ska kunna ge poäng till lag per tävling 2h
Verklig tid 2.5h
Alla poäng ska sparas till textfil 2h
verklig tid 2h
Testimplementation 3h
Verklig tid 2h
Sammanlagt: 11.5h
Verklig sammanlagd tid 12.5

Kravanalys

Alla tävlingar för säsongen ska finnas tillgängliga
Ett lag ska kunna anmäla sig till alla tävlingar
Anmäld lag till en tävling ska sparas i en textfil
Sekreterare ska kunna ge poäng till lag per tävling
Alla poäng ska sparas i textfil

Design och implementation

Klassen **Contests** skapades för att uppfylla kraven och har hand om allt med tävlingarna att göra.
String[] contentList är en array som består av alla tävlingar för säsongen.
registerTeamToContest() metoden registrera lag till tävlingar

registerTeamToContestForTest() är till för testning och är en kopia av **registerTeamToContest()** metoden, men retunerar boolean istället för spara registrering till textfil.
givePoints() metoden anropas för att ge poäng till lag
checkIfContestExist() metoden kollar om tävlingen existerar i **contestList**
checkIfTeamAlreadyRegisteredToContest() metoden kollar om ett lag redan är registrerad till en tävling.
checkIfTeamIsRegistered() metoden kollar så att ett lag existerar.

Uppnådda mål:

Alla tävlingar finns tillgängliga
Ett lag kan anmäla sig till tävlingar
Alla anmälningar till tävlingar lagras i ett text dokument
Det går att ge poäng till varje lag->tävling
Alla poäng för lag och tävling sparas i en textfil

Testning

Ett redan anmält lag ska inte kunna anmäla sig igen till samma tävling.
Test input: “lag1”, “contest1”, **Förväntat resultat:** false **Faktist resultat:** false

Kolla så att tävlingen någon försöker anmäla sitt lag till existerar
Test input1:“contest1”, **Förväntat resultat:** true **Faktist resultat:** true
Test input2: “nonExistingContest” **Förväntat resultat:** false **Faktist resultat:** false

Testa registrera lag
Test input: “lag5”, “contest2”, **Förväntat resultat:** true**Faktist resultat:** true

Testa ge poäng till lag
Test input: “lag1”, “contest1”, “1” **Förväntat resultat:** true **Faktist resultat:** true

Testa dra av poäng från lag
Test input: “lag1”, “contest1”, “-1” **Förväntat resultat:** true **Faktist resultat:** true

Testa ge poäng till lag men fel tävling
Test input: “nonExistingContest,”lag1”, “1” **Förväntat resultat:** false**Faktist resultat:** false

Teste ge poäng till lag som inte finns men tävling som finns
Test input: “contest1,”nonExistingTem”, “1” **Förväntat resultat:** false**Faktist resultat:** false

Testa logga in som registrerat lag
Test input: “lag1” **Förväntat resultat:** true **Faktist resultat:** true

Testa logga in som icke registrerat lag
Test input: “ejRegistreratlag” **Förväntat resultat:** false**Faktist resultat:** false

Testa logga in som sekreterare
Test input: “sekreterare” **Förväntat resultat:** true **Faktist resultat:** true

Reflektion

Planerad tid: 11.5h
Verklig tid: 12.5h
Gick över tidsschemat med 1 timme vilket var väldigt bra med tanke på att jag förde över några krav och tester från iteration 1. Inget i iteration 2 blev lidande och jag han med det jag hade planerat. Insåg även att jag kunde dra ner på tiden på iteration 3 då den mest handlar om att hämta ut data och med den struktur och kod jag har nu kommer det krävas väldigt lite implementation för att upfylla kraven för iteration 3. Även testning med jUnit i eclipse fungerar väldigt väl för denna uppgift och efter de första testimplementationerna har resten flytit på.

Iteration 3

Tidsplanering

Resultat för enskild tävling 2h
Verklig tid: 1h
Resultat för enskilt lag 1.5h
Verklig tid: 1h;
Resultat för alla lag hela säsongen 1.5h
Verklig tid: 1.5h
TestningImplementation: 1h
verklig tid: 0,5h
Sammanlagt 6h
Verkliga sammanlagda tid: 4h

Kravanalys

Man ska kunna få alla poäng för alla lag per enskild tävling
Man ska kunna få poäng för alla tävlingar per lag
Man ska få upp alla poäng för alla lag på hela säsongen

Design och implementation

Implementerade klassen **GetResults** för att upfylla kraven med metoderna:
getResultsForTeam() som retunerar lista med resultat för specifikt lag.
getResultsForContest() som retunerar lista med resultat för alla lag registrerade till specifik tävling
getResultsForSeason() som retunerar lista med totala poäng för alla lag registrerade för säsongen

Testning

Få tillbaka lista för alla lag i en tävling
Test input:“contest1” **Förväntat resultat:** list not null, list.size > 1 **Faktist resultat:** list not null, list.size > 1
Få tillbaka lista med alla tävlingar för ett lag
Test input1:“lag1” **Förväntat resultat:** list not null, list.size > 1 **Faktist resultat:** list not null, list.size > 1
Få tillbaka lista med alla lag för säsongen
Förväntat resultat: list not null, list.size > 1 **Faktist resultat:** list not null, list.size > 1

Uppnådda mål:

Man kan få en lista med alla lag och poäng för specifik tävling.
Man kan få en list med alla tävlingar och poäng för specifikt lag.
Man kan få en lista med alla lag och deras totala poäng för hela säsongen.

Reflektion

Planerad tid: 6h
Verklig tid: 4h
Sista iteration gick snabbar än planerat och implementation av de nya kraven var relativt enkelt då den mesta koden som behövdes för att hämta data från textfilerna redan fanns för återanvändning efter dom andra iterationerna. Allt jag behövde göra var att implementera en ny klass (**GetResults)** med metoder som hämtade data från färdiga metoder i klassen FileHandler som redan fanns implementerad från tidigare iteration, sen behövde jag bara behandla datat efter behov.

Slutreflektion

Svårigheter - Motgångar

Det svåraste med uppgiften var planeringen av tid och dokumentationen.
Att försöka planera hur lång tid ett mål kommer ta är alltid svårt men jag tyckte att jag gissade rätt så bra. Dock var planen att dela upp tiden jämnt mellan alla iterationer men desto längre in på projektet jag kom insåg jag att iteration 3 skulle gå snabbare och snabbare och för varje iteration kunde jag dra ner på tiden för varje uppsatt mål.

Den andra motgången kom halvägs in på iteration 1 då jag fick ny information efter att gått på handledningstillfälle. Jag fick senare i iteration ändra lite på min kodstruktur då jag insåg att strukturen jag hade nu var överflödig och skulle vara svår att testa.

Vad som gått bra

Det som gått bra med uppgiften var programmeringen, det flöt på och var relativt enkelt. Efter en del googlande och påläsning samt lite handledning var jUnit som jag använde mig av för testning väldigt lätt och bra för testning efter de första testerna flöt även detta på.