Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Laboration 3

Ett labora tions arbete av

Christopher Holmberg

ch22iu

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Innehållsförteckning

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Uppgift 1 - Planering

Uppgifter	Tid	Verk. Tid
Testplan	2h 15min	1h 25min
Design & Implement	6h 25min	3h 26min
Enhetstestning	5h 30min	5h 15min
Testvits	6h	3h 20min
Integrationstestning	3h 30min	3h 10min
Reflektion	2h	32min

Summering: 25h 40min
***Diff: 8h 32min**
***Verk: 17h 8min**

I slutet utav planeringen kommer jag att räkna på differencen mellan dom olika tiderna som jag har satt vid min planering.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Uppgift 2 - Testplan

1. Introduktion

Testplanen beskriver utvecklingsbolagets arbete och målsättning med tester under utvecklingsarbetet och leveransen av gymnastikligans gymnastiktävlingssystem. Testning planeras löpande under utvecklingsprocessen för att säkra upp leveransen av ett fungerande och driftsäkert system till gymnastikligan.

2. Teststrategi

Under utvecklingsarbetet planeras ett testarbete utföras på flera nivåer i systemet. Initialt planeras modultester av enskilda klasser i den takt de implementeras. I andra hand planeras integrationstester utföras då flera beroende klasser implementerats. Samtliga användningsfall i kravspecifikationen utgör grund för integrationstest vid implementering av beroende klasser.

2.1 Modultester

Modultester utförs med exekverar testkod som utformas för att testa grundläggande funktionalitet hos implementerad kod. Exempelvis kan konstruktorer, enskilda metoder och egenskaper i enskilda klasser testas. Efter lyckat modultest kan klassen integrationstestas med övriga berörda implementerade klasser.

2.2 Integrationstester

Integrationstester utförs senare i utvecklingsskedet när flera va varandra beroende klasser implementerats. Testerna fokuserar på att testa funktionalitet mellan klasser såsom delade uppgifter och anrop mellan klasser.

3. Testidéer

Under testarbetet prioriteras tester baserade på beskrivna användningsfall. Utvecklingsteamet har identifierat ett användningsfall som beskriver poängsättning av gymnaster som extra sårbart och därav fokuserar de första testerna på just detta användningsfall. Med användningsfallet kommer tester säkra upp att poäng kan ges, beräknas och registreras på enskilda gymnaster - ett måste för användandet av levererat system.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

3.1 Avgränsning

Testplanen inkluderar inga system eller användartester utan planerar enbart för modul och integrationstester. Detta efter att kund formulerat en mindre budget för testning än efterfrågat. Utvecklingsbolagets rekommendation är att öka budgetutrymmet och omfattningen av testning och menar att exempelvis användartestning kan adderas i ett senare skede. De tester som innefattas av testplanen återfinns alla i tidigare presenterade användningsfall. Inga tester är planerade för kod eller scenarion som inte redan är beskrivna.

4. Risker

Test utgör i grunden ett underlag för minskade risker men vi har återfunnit ett fåtal risker i testplanen som bör belysas. Ansvarig för testning har en begränsad kompetens som kan komma att påverka slutresultatet. Med satt budget kan vi inte sätta en mer erfaren medarbetare på projektet. Den budget som kund satt för testning anser vi var låg och rekommendationen är att öka budgeten för att tester ska täcka en större del av implementerad kod. Enligt testplanen testas användningsfall men ytterligare testning är nödvändig för att garantera driftsäkerhet.

5. Testprocess

Har listat dom olika delarna i testet så som exekvering, testverktyg, dokument, Testmiljö osv.

5.1 Ansvarsfördelning

För testplanering, exekvering och dokumentation används en student från Linnéuniversitetet.

5.2 Testmiljö

Projektet kommer att planeras och exekveras i Visual Studio 2013 vilket behöver installeras och licensieras på samma dator.

5.3 Testverktyg

För testgenomförande krävs dator med tillgång till projektets källkod. Inga övriga verktyg bedöms nödvändiga men förändringar i testplanen kan leda till förändringar av behov av specifika testverktyg.

5.4 Tidsåtgång och deadline

Testfall ska exekveras och dokumenteras för att kommuniceras till kund senast 2014-12-18. För testprojektet är avsatt 26 timmar och 30 minuter arbetstid.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

6. Resultat & Redovisning

Samtliga godkända resultat kommuniceras med dokumentation, källkod och testdata till gymnastikligan. Dessa resultat kommuniceras tillsammans med övriga delar av testplanen och testfallen för att gymnastikligan ska få full insyn i testprocessen.

6.1 Kriterier för godkänt testresultat

Samtliga testresultat anses godkända först när testdata går att tolka i sin helhet. Om testdata inte anses komplett planeras och exekveras tester till dess att testdata bedöms komplett.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Uppgift 3 - Design & Implementation

Användningsfallet som kommer att testas i laborationen hämtas från laboration 3. För att omfattningen av tester (såväl enhetstester av klasser och integrationstest) ska landa på en lagom nivå för utförandet av laborationen har användningsfallet reviderats. Förändringar som gjorts är att en aktör (jury) avlägsnats samt att primörflödet förkortats för att enbart innehålla aktiviteter vid införande av och senare godkännande av poängbedömning. Förändringarna lämnar tydligare struktur för testning med färre klasser och flöden än ursprungsversionen. Nedan presenteras den versionen av användningsfallet som senare testas.

Under tävling sätter domare poäng. I systemet registreras sedan poängen från samtliga domare som tar bort högsta/lägsta poäng samt beräknar medelvärde på resterande domarpoäng vilket utgör slutgiltig poäng. Sekreterarens är att registrera domarpoäng samt att avslutningsvis (efter att ha presenterats slutgiltig poäng) godkänna slutgiltig poäng. Om slutgiltig poäng ej godkänns bes sekreteraren att på nytt mata in samtliga domares poäng.

Aktör

Sekreterare

Pre-conditions

Det hålls tävling och det deltar certifierade domare. Sekreterare är registrerad i systemet sedan tidigare och inloggad som administratör. Systemet har struktur för att behandla grenar, gymnaster och poängbedömningar.

Primärt flöde

1. Användarfallet påbörjas när samtliga domare gjort sin bedömning av en specifik gymnast
2. Sekreteraren samlar in och för in samtliga poängbedömningar i systemet
3. Systemet tar bort lägsta och högsta poängbedömningen
4. Systemet beräknar ett medelvärde för resterande poängbedömningar
5. Slutgiltig poäng presenteras i systemet
6. Sekreteraren godkänner (kontroll) slutgiltig poäng
7. Systemet: Slutgiltig poäng registreras på enskild gymnast

Alternativa flöden

Om (6) sekreteraren inte godkänner

a. Systemet ber sekreteraren mata in poängbedömningar på nytt (2)
b. Systemet tar bort högsta/lägsta poäng (3)
c. Systemet beräknar medelvärde (4)
d. Systemet presenterar slutgiltig poäng (5)
e. Sekreteraren ges möjlighet att godkänna (6)

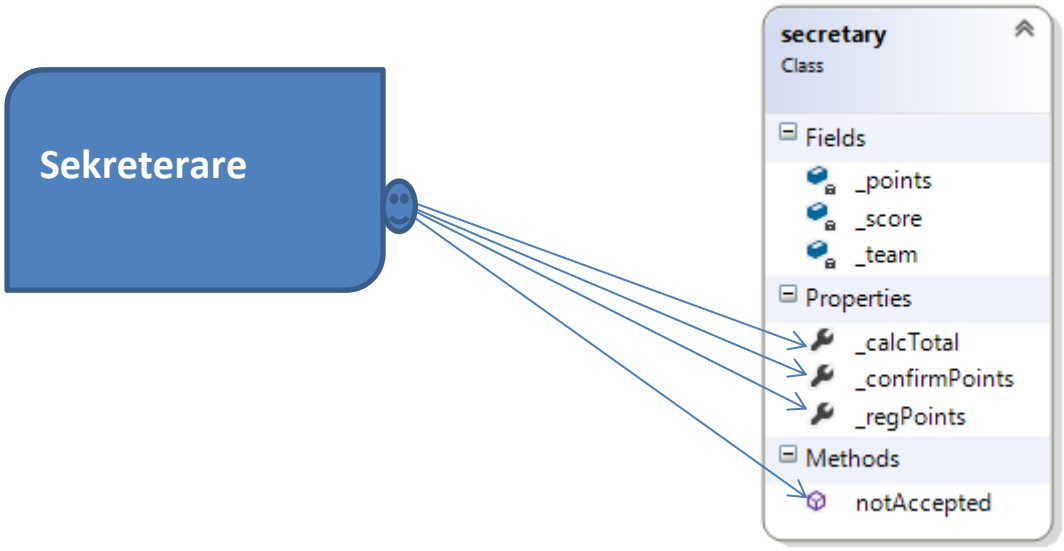
Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Post-conditions

Slutgiltig poäng är korrekt registrerad i systemet och tävlingen fortsätter med ytterligare tävlande och ytterligare poängbedömningar.

Användningsfallsmodell

Aktör

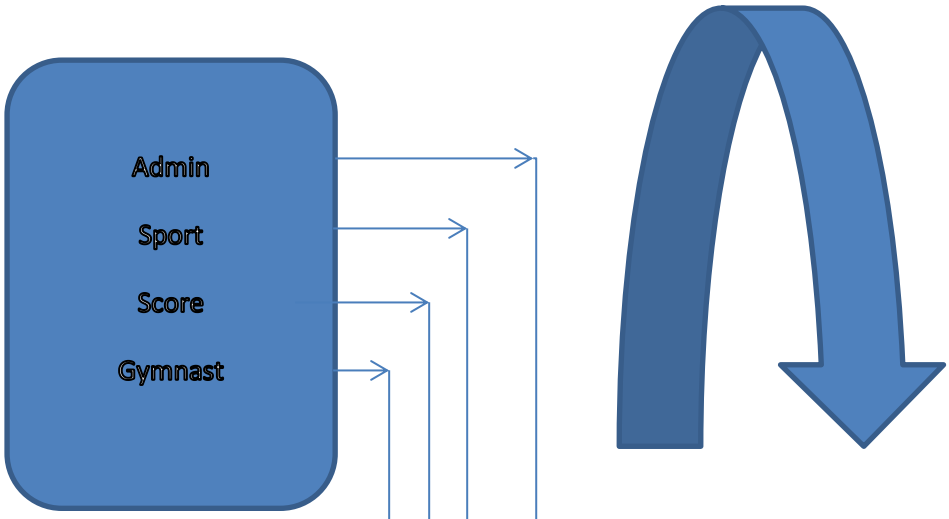


1. Aktören registrerar poängbedömningar (_calcPoints) samt beräknar slutgiltig poäng (_calcTotal)
2. Aktören godkänner poängen. (_confirmPoints)
3. Aktören (sekreteraren) beräknar slutliga poäng och även kollar ännu ej godkända poäng. (notAccepted)

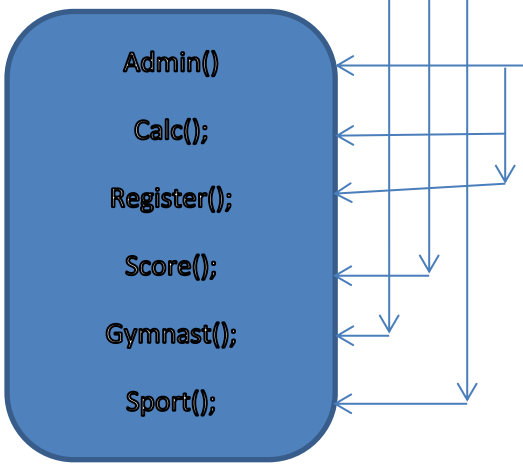
Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Klasser

Använda klasser

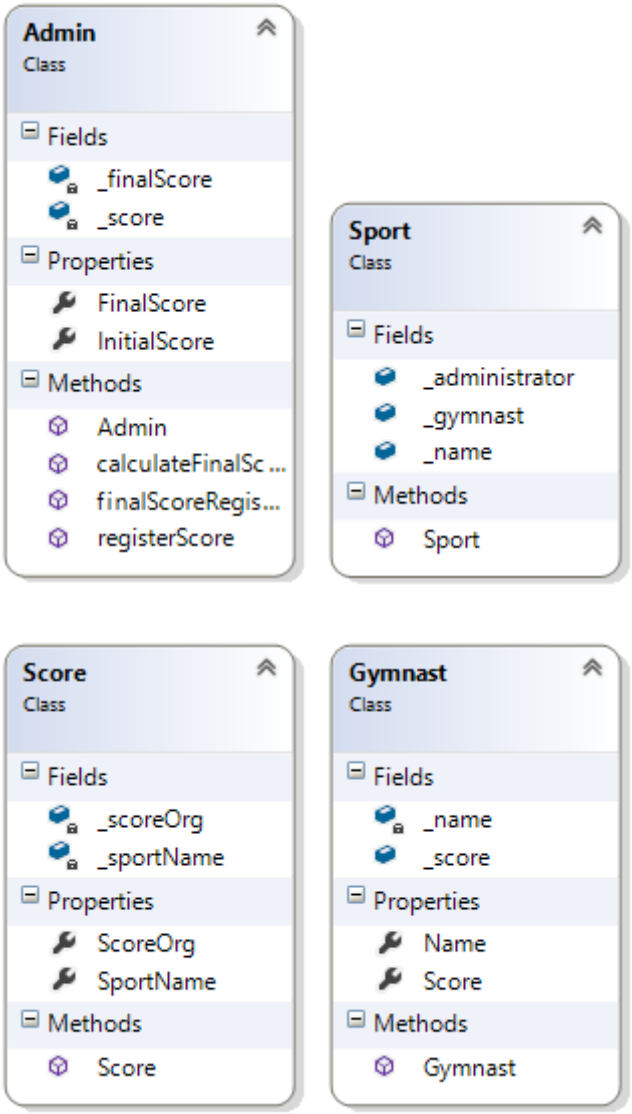


Använda metoder



Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Klassdiagram



Beskrivning:

Vid implementering kan tester påvisa användningsfall under specifik gren. Arkitekturen kan sedan sättas i bruk i ett större perspektiv genom införande av Event klass som kan samla flera Sport klasser för flera grenar under en tävling tillsammans med ytterligare metoder som kan krävas. Meningen är att klassen Sport (gren) i sin tur har objekt av klassen Admin och Gymnast som i sin tur har objekt av klassen Score. Nedan följer specificerade beskrivningar av klasserna och dess fält och metoder.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Sport

Klassen sport (gren) har ett namn, en administratör och en samling gymnaster (beroende på min framgång, eventuellt får tester köras med enbart en gymnast men ambitionen är flera). Denna förenklade gren klass kan i senare releases byggas på med metoder och fält för hantering och registrering av samlade lagpoäng etc. I senare versioner samlar en större event klass flera grenar och styr konstruktionen av Sport objekt.

Admin

Admin klassen innehåller metoder för att registrera poängbedömningar, beräkna slutgiltiga poäng samt registrera dessa. Tanken är att Admin klassen motsvarar en sekreterares roll. För integritet vid användning hålls metoder i största utsträckning privata.

Gymnast

Klassen Gymnast ligger till grund för objekt innehållandes en gymnasts namn och poäng. Första planen var att låta sport klassen innehålla en samling av gymnast objekt men för laborationen skapas bara ett gymnast objekt per gren.

Score

Klassen poäng ligger till grund för poängobjekt som lagrar gren och resultat.

Övrigt

Överblicka på bilderna på sidan 10 samt att all kod är kommenterad i originalfilerna i mitt repository.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Uppgift 4 – Enhetstestning

Testsvit

Identifikation

Testsvit #1A

Modell

Kod

```
public void testsvit()
{
    // Skapar upp ett nytt objekt utav Sport
    Sport sportTest1 = new Sport("SportTest", "TestGymnast");

    // Starta mina tester.
    test1(sportTest1._administrator);
    test2(sportTest1._administrator);
    test3(sportTest1._administrator);
    test4(sportTest1._gymnast._score);
}

// Test case 1 - Antal domare
1 reference
public void test1(Admin adminTest)
{
    // Testdata för anrop
    int[] first = {1, 10, 2, 4, 7, 7};
    int[] second = {1, 10, 2, 4, 7};
    int[] third = {1, 10, 2, 4, 7, 7, 4};

    // Samlar alla anrop i en list
    List<int[]> calls = new List<int[]>();
    calls.Add(first);
    calls.Add(second);
    calls.Add(third);

    // Kör alla anrop i en try catch sats som skriver ut meddelande beroende på utfall
    for (int i = 0; i < 3; i++)
    {
        string call = String.Join(",", calls[i]);
```

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Beskrivning

Testsviten omfattar 4 testfall och 1 testfixtur. Testfallen är samtliga exekverbara modultester med syfte att testa grundläggande funktionalitet i utvalda klasser (Admin och Score). Nedan presenteras en modell över testsvitens sammansättning varefter testfall och fixtur presenteras detaljerat. Slutligen presenteras motivering till sammansättning av testsvit och utvalda testfall.

Personalresurser och ansvarsfördelning

Testutförande för denna laboration är Christopher Holmberg, ch22iu

Schemaläggning

Till sitt förfogande har ansvarig givits en beräknad arbetstid om 9.5 arbetstimmar för planering, exekvering och dokumentation av testsviten. Deadline är satt till 2014-12-15.

Testsvitens pre- & post-conditions

Pre-conditions

Testsviten förutsätter att klasser som ska modultestas är implementerade och åtkomliga för test.

Post-conditions

Efter genomförd testsvit ska det vara möjligt att gå vidare med integrationstestning av implementerade klasser.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testfall 1.1 – Antal domare

Klass

Admin

Beskrivning

Testfallet ämnar testa metoden registerScore samt egenskapen InitialScore. Systemet är tänkt att vara begränsat till enbart 6 domare vilket även ger att varje beräkning av slutpoäng kräver specifikt 6 heltal lagrat i värdefältet _initialScore som sätts av egenskapen InitialScore. Under testet kommer olika sammansättningar av poängbedömningar att testas på metoden för att säkra upp att endast fungerande poängbedömningar används. Under post-conditions listas de värden som under testat anropar metoden tillsammans med förväntade resultat och kommentarer.

Pre-conditions

För att genomföra test krävs en instans av Sport objekt korrekt initierat (med admin objekt). För att uppfylla dessa krav används initialt fixtur (basklass).

Post-conditions

Nedan listas i testet använda anrop med förväntat resultat och kommentar.

Värde: anropar	Resultat	Kommentar
[1, 10, 2, 4, 7, 7]	OK	6 värden
[1, 10, 2, 4, 7]	Undantag kastas	5 värden (för några)
[1, 10, 2, 4, 7, 7, 4]	Undantag kastas	7 värden (för många)

Data required

Testerna kräver tillgång till testfixtur som exekveras först. Därutöver kräver testerna en metod som skickar listade anrop till initierat admin objekts metod samt visualiserar resultatet av varje anrop för testdokumentation.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testfall 1.2 – Poängbedömningar

Klass

Admin

Beskrivning

Testfallet ämnar testa egenskapen InitialScore som innan den sätter värde på _initialScore ska säkerställa att varje värde är i intervallen 1-10. Under testet kommer olika sammansättningar av poängbedömningar att testas på egenskapen för att säkra upp att endast poängbedömningar inom korrekt intervall går igenom egenskapen. Under postconditions listas de värden som under testat anropar egenskapen tillsammans med förväntade resultat och kommentarer.

Pre-conditions

För att genomföra test krävs en instans av Sport objekt korrekt initierat (med admin objekt). För att uppfylla dessa krav används initialt fixtur.

Post-conditions

Nedan listas i testet använda anrop med förväntat resultat och kommentar.

Värde: anropar	Resultat	Kommentar
[1, 10, 2, 4, 7, 7]	OK	6 godkända värden
[1, 10, 2, 7, 7, 7]	Undantag kastas	Innehåller ett högt värde
[1, 10, 2, 4, 7, 7, 4]	Undantag kastas	Innehåller ett lågt värde

Data required

Testerna kräver tillgång till fixtur som exekveras först. Därutöver kräver testerna en metod som skickar listade anrop till egenskap hos initierat admin objekt samt visualiserar resultatet av varje anrop för att möjliggöra testdokumentation.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testfall 1.3 – Total poäng

Klass

Admin

Beskrivning

Testfallet ämnar testa metoden calculateFinalScore som tar en int array och returnerar en double, Metoden tar bort högsta och lägsta värde i array och returnerar medelvärdet på resterande värden. För att testa metoden anropas den men olika värden. Under postconditions listas värden som testet anropar metoden med samt förväntade resultat och kommentarer.

Pre-conditions

För att genomföra test krävs en instans av Sport objekt korrekt initierat (med admin objekt). För att uppfylla dessa krav används initialt fixtur.

Post-conditions

Nedan listas i testet använda anrop med förväntat resultat och kommentar.

Värde: anropar	Resultat	Kommentar
[1, 10, 2, 4, 7, 7]	5.0	Beräkningstest
[1, 7, 2, 8, 7, 3]	4.75	Beräkningstest
[10, 10, 2, 3, 8, 7]	7.0	Beräkningstest

Data required

Testerna kräver tillgång till fixtur som exekveras först. Därutöver kräver testerna en metod som skickar listade anrop till initierat admin objekts metod samt visualiserar resultatet av varje anrop för testdokumentation.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testfall 2.1 – Lägg till poäng (Set)

Klass

Admin

Beskrivning

Testfallet ämnar testa metoden registerScore samt egenskapen InitialScore. Systemet är tänkt att vara begränsat till enbart 6 domare vilket även ger att varje beräkning av slutpoäng kräver specifikt 6 heltal lagrat i värdefältet _initialScore som sätts av egenskapen InitialScore. Under testet kommer olika sammansättningar av poängbedömningar att testas på metoden för att säkra upp att endast fungerande poängbedömningar används. Under post-conditions listas de värden som under testat anropar metoden tillsammans med förväntade resultat och kommentarer.

Pre-conditions

För att genomföra test krävs en instans av Sport objekt korrekt initierat (med admin objekt). För att uppfylla dessa krav används initialt fixtur (basklassen).

Post-conditions

Nedan listas i testet använda anrop med förväntat resultat och kommentar.

Värde: anropar	Resultat	Kommentar
5	5	Double utan dec
4.75	4.75	Double med dec

Data required

Testerna kräver tillgång till fixtur som exekveras först. Därutöver kräver testerna en metod som skickar listade anrop till initierat admin objekts metod samt visualiserar resultatet av varje anrop för testdokumentation.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testsvit

Beskrivning

Testsvit initierar ett Sport objekt vid namn testSport med parametrarna (testSport och TestGymnast). Vid initieringen säkerställer fixturen att förutom ett sport objekt initieras så initieras samtidigt ett Admin objekt, ett Gymnast objekt samt ett Score objekt. Dessa objekt säkerställer att tester kan köras för samtliga testfall i testsviten

Testsvits skapas som en egen klass med en metod fixture som instansierar ett Sportobjekt med ovan beskrivna parametrar.

Du kan även se bilden testgrund.png i bifogad fil. Samt på blad 12.

Sammansättning

Testsvitens sammansättning är genomförd med tre huvuddrag i åtanke. Testsvitens omfång sätts till en nivå som är genomförbar på schemalagd tid. Testsvitens testfall är modultester som utförs på två olika klasser. Testsvitens utfall (förhoppningsvis positivt) ligger till grund för att senare testa samma metoder och egenskaper ur ett integrationsperspektiv. Dessa huvuddrag säkerställer att testfallen blir genomförbara, testresultaten blir användbara som grund för senare tester samt kanske viktigast - laborationskraven uppfylls. Testsvitens omfattning och val av tester har även formats av tidigare presenterade och implementerade klasser (uppgift 3). Helt enkelt är testerna fokuserade att köras på redan implementerad kod för att säkerställa funktionalitet. Ytterligare testning krävs vid implementering av fler klasser i en senare release. Testet kan ses testa grundläggande arkitektur på redan implementerad kod.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Uppgift 5 – Implementera testsviten och kör

Kommenterad kod för testklasserna

För kodning av test och exekvering av desamma har jag valt att bygga en testklass för hela testsviten. Testklassen inkluderas i inlämningen av samtliga källfiler och klassen är döpt till Testsvit1. Jag har även valt att skriva en metod som presenterar testdata i konsollfönster (exekveras i konsol) i samma testklass. Testklassen samlar metod för testfixtur och exekvering av samtliga tester i testsviten samt en metod för varje testfall skriven för att utföra specifika tester.

Inser efter att det blivit 200 rader kod att en uppdelning i flera klasser hade varit att föredra men då allting fungerar fint och ger testaren en bra visuell överblick över genomförda tester väljer jag att köra vidare på modellen.

Dokumentation av de utförda testerna & testdata

```
public void fixture()
{
    //Set-up test fixture
    Sport sportTest = new Sport("TestSport", "TestGymnast");

    //Execute tests
    runTestA01(sportTest._administrator, sportTest._gymnast);
    runTestA01B(sportTest._administrator, sportTest._gymnast);
}
```

Detta är min basklass – fixture som kör mina tester. (Integrationstestning)

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testfall 1.1 – Antal domare

Tillvägagångssätt

Med metoden test1() anropades ett Admin objekts metod registerScore som i sin tur anropar egenskapen score för att sätta värde på fältet _initialScore. Nedan listas i testet använda anrop med förväntat resultat och kommentar.

Värde: anropar	Förväntat resultat	Kommentar
[1, 10, 2, 4, 7, 7]	OK, Sätter _score	6 värden
[1, 10, 2, 4, 7]	Undantag kastas	5 värden (för några)
[1, 10, 2, 4, 7, 7, 4]	Undantag kastas	7 värden (för många)

Testdata

Nedan presenteras testdata från genomförd exekvering av testklass testsvit. Utfallet motsvarade förväntat resultat på samtliga punkter.

Värde: anropar	Förväntat resultat	Faktiskt resultat
[1, 10, 2, 4, 7, 7]	_score = [1, 10, 2, 4, 7, 7]	samma
[1, 10, 2, 4, 7]	Undantag kastas	samma
[1, 10, 2, 4, 7, 7, 4]	Undantag kastas	samma

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testfall 1.2 – Poängbedömningar

Tillvägagångssätt

Med metoden test2() anropades ett Admin objekts metod registerScore som i sin tur anropar egenskapen score för att sätta värde på fältet _score. Nedan listas i testet använda anrop med förväntat resultat och kommentar.

Värde: anropar	Förväntat resultat	Kommentar
[1, 10, 2, 4, 7, 7]	OK, Sätter _score	6 godkända värden
[1, 12, 2, 4, 7, 7]	Undantag kastas	Innehåller ett för högt värde
[1, 10, 2, 0, 7, 7]	Undantag kastas	Innehåller ett lågt värde

Testdata

Nedan presenteras testdata från genomförd exekvering av testklass testsvit. Utfallet motsvarade förväntat resultat på samtliga punkter.

Värde: anropar	Förväntat resultat	Faktiskt resultat
[1, 10, 2, 4, 7, 7]	_score = [1, 10, 2, 4, 7, 7]	samma
[1, 10, 2, 4, 7]	Undantag kastas	Undantag kastas
[1, 10, 2, 4, 7, 7]	Undantag kastas	Undantag kastas

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testfall 1.3 – Total poäng

Tillvägagångssätt

Med metoden test3() anropades ett Admin objekts metod calcFinalScore som beräknar slutpoäng. Med en int array som indata ska metoden sätta _finalScore till medeltalet av arrayens samtliga tal förutom högsta och lägsta.

Värde: anropar	Förväntat resultat	Kommentar
[1, 10, 2, 4, 7, 7]	5.0	Test av beräkning
[1, 7, 2, 8, 7, 3]	4.75	Test av beräkning
[10, 10, 2, 3, 8, 7]	7.0	Test av beräkning

Testdata

Nedan presenteras testdata från genomförd exekvering av testklassen testsvit. Utfallet motsvarade förväntat resultat på samtliga punkter.

Värde: anropar	Förväntat resultat	Faktiskt resultat
[1, 10, 2, 4, 7, 7]	5.0	5.0
[1, 7, 2, 8, 7, 3]	4.75	4.75
[10, 10, 2, 3, 8, 7]	7.0	7.0

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testfall 2.1 – Lägg till poäng (Set)

Tillvägagångssätt

Med metoden test4 anropades ett Admin objekts metod calcFinalScore som beräknar slutpoäng. Med en int array som indata ska metoden sätta _totalscore till medeltalet av arrayens samtliga tal förutom högsta och lägsta.

Värde: anropar	Förväntat resultat	Kommentar
5	5	Double utan decimal
4.75	4.75	Double med decimal

Testdata

Nedan presenteras testdata från genomförd exekvering av testklassen test. Utfallet motsvarade förväntat resultat på samtliga punkter.

Värde: anropar	Förväntat resultat	Faktiskt resultat
5	5	Double utan decimal
4.75	4.75	Double med decimal

Slutsats

Efter påvisad funktionalitet i samtliga modultester anser vi de i systemet implementerade klasserna redo att integrationstestas. Förslagsvis genomförs första integrationstest på användningsfallet presenterat i uppgift 3.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Uppgift 6 – Integrationstestning

Här kommer jag presentera en rätt så stor del utav några testfall, testa majoriteten av implementerad kod för användningsfallet. Stegvis beskrivning för arbetet med att ta fram testfall samt testdata under uppgift 6.

1. Analys av användningsfall

Genom att analysera användningsfallets primärflöde och alternativa flöde sedan sätter jag testfall efter varje punkt. Jag gör en enklare modell över användningsfallens flöden med steg att testa för att förenkla arbetet.

2. Testfallens täckning

Jag säkerställer att testfallen täcker samtliga variationer som kan återfinnas i primärflödet.

3. Planera anrop

Jag planerar data som ska användas i testet för att säkerställa täckning av användningsfallets samtliga utfall.

4. Fixturplanering

Jag planerar testfixtur med hjälp av testfallens pre-conditions och bedömer att samma fixtur som användes till modultester går att applicera på dessa testfall.

5. Kod

Jag skriver testfixtur och testklass. Testfallen använder jag som mall för hur testklassen utformas.

6. Kod igen

Jag går igenom koden för att kolla att jag inte missat något och skriver kod som presenterar testfallens data.

7. Exekvering

Jag exekverar koden och utvinner testdata genom metod i testklassen som skriver ut testresultat i konsolen.

8. Presentation

Jag presenterar datan i tabeller som är lättlästa tillsammans med en kortare beskrivning av tillvägagångssätt under testexekveringen.

Kommenterad kod

Kod finns inlämnad i sin helhet och distribuerades tillsammans med laborationsrapporten. I mappen ”laboration3” samlas all källkod för såväl tester som design och implementation av klasser.

Källkoden för integrationstester återfinns i testklassen Integrationstest.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

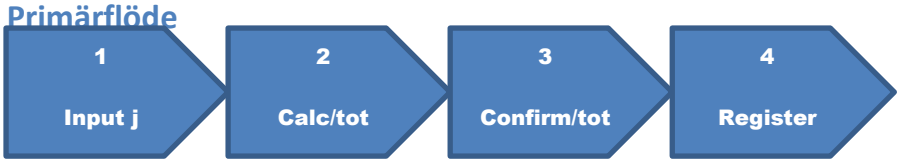
Testfall

Beskrivning

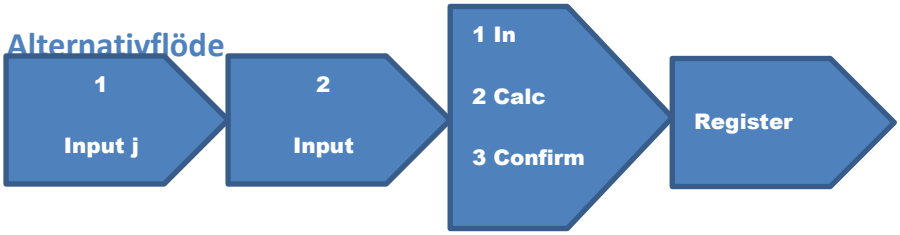
Eftersom modultesterna som genomförts tidigare verifierat olika vägar genom metoderna fokuserar jag i integrationstesterna att säkerställa att metoder och klasser går att köra tillsammans snarare än att utforska samtliga vägar genom varje metod som ju redan verifierats genom genomförda modultester. Integritetstestningen kommer att utföras med två testfall; ett som efterliknar primärflödet och ett som efterliknar alternativflödet. Nedan presenteras en enklare modell som beskriver flödena.

Modell av flöden i användningsfall (primär & alternativflöde)

Primärflöde



Alternativflöde



Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

TestfallA01 - Primärflöde

Beskrivning

Testfallet ämnar testa primärflödet i beskrivet användningsfall. Testfallet kommer att föra in domarbedömningar, beräkna slutpoäng genom att behandla domarbedömningar, presentera och godkänna slutpoängen och registrera poängen på en gymnast. Eftersom testfallet är av typen integrationstest kommer flera metoder och klasser att anropas under samma testfall. Detta för att i huvudsak testa samspelet mellan var för sig fungerande metoder och klasser.

Pre-conditions

För att genomföra test krävs en instans av Sport objekt korrekt initierat (med admin objekt). För att uppfylla dessa krav används initialt fixtur . För första anrop krävs påhittade domarbedömningar i en array. För testet används godkända (inom intervall 1-10 samt totalt 6 st) [7, 6, 10, 8, 5, 8].

Post-conditions

Testfallet förutsätter en lyckad registrering och gymnastiktävlingen fortsätter med ytterligare gymnaster som ska poängbedömas etc. Nedan listas i testet använda anrop med parametrar.

Metod som anropas	Klassen	Parametrar	Kommentar
registerScore	Admin	[7, 6, 10, 8, 5, 8]	Sätter _score
calcScore	Admin	initScore	Beräknar slutpoäng
finalRegScore	Admin	FinalScore, Gymnast, true	Godkänner registering
Score	Gymnast	FinalScore	Registrerar (med tidigare anrop efter godkännande)

Data required

Testerna kräver tillgång till fixtur (basklasssen) som exekveras först. Därutöver kräver testerna en metod som skickar listade anrop till initierade objekts metoder samt visualiserar resultatet av varje anrop för testdokumentation.

Resultat

Testdokumentation och testdata presenteras efter genomfört test.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

TestfallA01B - Alternativflöde

Beskrivning

Testfallet ämnar testa alternativflödet i beskrivet användningsfall. Testfallet kommer att föra in domarbedömningar, beräkna slutpoäng genom att behandla domarbedömningar, presentera och underkänna slutpoängen för att därefter repetera processen och slutligen registrera poängen på en gymnast. Eftersom testfallet är av typen integrationstest kommer flera metoder och klasser att anropas under samma testfall. Detta för att i huvudsak testa samspelet mellan var för sig fungerande metoder och klasser.

Pre-conditions

För att genomföra test krävs en instans av Sport objekt korrekt initierat (med admin objekt). För att uppfylla dessa krav används initialt fixtur . För första anrop krävs påhittade domarbedömningar i en array. För testet används godkända (inom intervall 1-10 samt totalt 6 st) [7, 6, 10, 8, 5, 8].

Post-conditions

Testfallet förutsätter en lyckad registrering och gymnastiktävlingen fortsätter med ytterligare gymnaster som ska poängbedömas etc. Nedan listas i testet använda anrop med parametrar.

Metod som anropas	Klassen	Parametrar	Kommentar
registerScore	Admin	[7, 6, 10, 8, 5, 8]	Sätter _score
calcScore	Admin	initScore	Beräknar slutpoäng
finalRegScore	Admin	FinalScore, Gymnast, true	Underkänner registering
registerScore	Admin	Score	Sätter _score på nytt
calculateFinalScore	Admin	[7, 7, 10, 8, 8, 10]	Beräknar slutpoäng på nytt
finalScoreRegistration	Admin	FinalScore,Gymnast, true	Godkänner registering
Score	Gymnast	FinalScore	Registrerar (med tidigare anrop) efter godkännande.

Data required

Testerna kräver tillgång till fixtur (basklasssen) som exekveras först. Därutöver kräver testerna en metod som skickar listade anrop till initierade objekts metoder samt visualiserar resultatet av varje anrop för testdokumentation.

Resultat

Testdokumentation och testdata presenteras efter genomfört test.

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Testdokumenation & testdata

TestfallA01 - Primärflöde

Tillvägagångssätt

Under testfallet exekverar testklassens kod metoder i implementerade klasser för att simulera användningsfall. Med metoden runTestA01 körs testklassens kod för testfallet. Efter nedan uppställning.

Metod som anropas	Klassen	Parametrar
registerScore	Admin	[7, 6, 10, 8, 5, 8]
calcScore	Admin	initScore
finalRegScore	Admin	FinalScore, Gymnast, true
Score	Gymnast	FinalScore

Testdata

Som stöd för rapportering skriver testklassen ut testutfall efter varje anrop vilka sedan sammanställs till testdatan som presenteras nedan. Samtliga metoder i testklassen bedömdes godkända vilket ger att testet inte upptäckt felaktigheter i testade klasser och metoder.

Metod som anropas	Klassen	Parametrar	Förvänat resultat	Faktiskt resultat
registerScore	Admin	[7, 6, 10, 8, 5, 8]	[7, 6, 10, 8, 5, 8]	[7, 6, 10, 8, 5, 8]
calcScore	Admin	initScore	7,25	7,25
finalRegScore	Admin	FinalScore, Gymnast, true	Godkänd registering	Godkänd registering
registerScore	Gymnast	FinalScore	7,25	7,25

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

TestfallA01B - Alternativflöde

Tillvägagångssätt

Under testfallet exekverar testklassens kod metoder i implementerade klasser för att simulera användningsfall. Med metoden runTestA01 körs testklassens kod för testfallet. Efter nedan uppställning.

Metod som anropas	Klassen	Parametrar
registerScore	Admin	[7, 6, 10, 8, 5, 8]
calcScore	Admin	InitScore
finalRegScore	Admin	FinalScore, Gymnast, false
registerScore	Admin	[7, 7, 10, 8, 8, 10]
calFinalScore	Admin	InititalScore
finalScoreReg	Admin	FinalScore, Gymnast, true
Score	Gymnast	FinalScore

Testdata

Som stöd för rapportering skriver testklassen ut testutfall efter varje anrop vilka sedan sammanställs till testdatan som presenteras nedan. Samtliga metoder i testklassen bedömdes godkända vilket ger att testet inte upptäckt felaktigheter i testade klasser och metoder.

Metod som anropas	Klassen	Parametrar	Förväntat resultat	Faktiskt resultat
registerScore	Admin	[7, 6, 10, 8, 5, 8]	[7, 6, 10, 8, 5, 8]	[7, 6, 10, 8, 5, 8]
calcScore	Admin	InitialScore	7,25	7,25
finalRegScore	Admin	FinalSco, Gym, false	Icke genomförd	Icke genomförd
registerScore	Admin	[7, 7, 10, 8, 8, 10]	[7, 7, 10, 8, 8, 10]	[7, 7, 10, 8, 8, 10]
calculateFinalScore	Admin	InitialScore	8,25	8,25
finalScoreRegistration	Admin	FinalSco, Gym, true	Godkänd reg	Godkänd reg
Score	Gymnast	FinalScore	8,25	8,25

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Uppgift 7 – Reflektion

Svårigheter

Svårast med laborationen - i princip alla uppgifter var att det var svårt att närma sig uppgifterna. Med begränsad kunskap och något luddiga formuleringar sökte jag mycket information i kurslitteratur och online och även om det finns mycket att hitta så skiljer sig materialet åt. En reflektion är att det finns flera olika metoder att ta sig an testning beroende på omfattning, system och organisation. Jag känner mig tämligen nöjd med mina val men är inte helt säker på att jag inte missat någon detalj som borde varit med. Är ju det som är avgränsning helt enkelt och testning upplever jag kräver en hel del.

Funderingar kring testkoden

Testkoden var svår att skriva - egentligen inte med hänsyn till själva koden utan till utformning och exekvering. Det finns bra miljöer gjorda för testning men att sätta sig in i en syntax, IDE och testutformningsmetodik utanför uppgifterna upplever jag som stört omöjligt (kursen är halvfart och vi har en begränsad tid till inlämning). Jag tror definitivt att jag skulle skriva min kod annorlunda vid ett annat tillfälle då jag inser att mina konsol utskrifter äter kodrader och mina testklasser med fördel hade kunnat delas upp. Hade varit enklare att jobba i grupp.

Funderingar kring arbetet

Själva testarbetet i sig upplever jag i grunden väldigt logiskt även om strukturer som vi behöver för ordning och reda komplicerar vetandet. Jag tror att laborationen gett mig en bra första inblick i testning och en respekt för densamma.

Allämnat

Omfattningsmässigt tycker jag att min avgränsning av användningsfall blev skaplig och effektiv tid klockad med rapporten landar på 19h och 30 minuter som jag anser vara lagom, kanske till och med lite i överkant.

Övrigt

Jag tyckte själv att jag hade en hyffsad bra bild över hur långt tid själva arbetet skulle ta. Jag planerade innan och nu i efterhand ser man att man har fått större blick på större projekt. Att en förstudie är väldigt viktigt. *Min tid för hela rapporten samt alla andra delar i sig slutade på 17h 8min.*

Anv: ch22iu
Namn: Christopher Holmberg
Datum: 2014-12-12

Tidslogg över hela projektet

Datum	Tid	Information
2014-12-10	07.30	Planering/Förstudie
2014-12-11	09.00→14.30	Förstudie/TestFall
2014-12-12	10.00→12.00	TestFall
2014-12-13	13.00→17.00	Testvits
2014-12-15	19→01.00	Integration
2014-12-16	22→01.00	Implementation & Kör
2014-12-16	07.30→10.00	Reflektion
2014-12-16	10.00→11.00	Renskrivning - Inlämning

Man fick läsa på en heldel utöver och samla massa information från dom olika föreläsningarna samt att alltid försöka hålla ramen inom den genomtänkta tiden som jag satte från början utav projektet. Jag höll tiderna nästan exakta.