

## ITERATION 2

EN JURY VILL MATA IN SINA POÄNG FÖR VIDARE BERÄKNINGAR

### SPECIFICERAD PLANERING

KRAV	PLANERAD TID	VERKLIG TID
Analysera användningsfall	15 Minuter	15 Minuter
Specificera	30 Minuter	30 Minuter

DESIGN / IMPLEMENTATION	PLANERAD TID	VERKLIG TID
Identifiera klasser	30 Minuter	30 Minuter
Specificera klasser	1 Timma 15 Minuter	1 Timma
Implementation	1 Timma 30 Minuter	1 Timma 15 Minuter

TEST	PLANERAD TID	VERKLIG TID
Test design	45 Minuter	30 Minuter
Test specifikation	45 Minuter	30 Minuter
Test implementation	1 Timma 15 Minuter	1 Timma 15 Minuter
Test exekvering	30 Minuter	45 Minuter

REFLEKTION	PLANERAD TID	VERKLIG TID
Reflektera	45 Minuter	1 Timma

TOTAL PLANERAD TID: 8 TIMMAR

TOTAL VERKLIG TID: 7 TIMMAR 30 MINUTER

# DESIGN OCH IMPLEMENTATION

## FLÖDESSCHEMA

### PRE-Villkor

1. Juryn skall ha ett gemensamt registrerat användarkonto i systemet.
2. Juryn skall vara inloggad i systemet.

### POST-Villkor

1. Gymnasternas medelvärde och lagets poäng i den valda tävlingsgrenen skall vara uträknade.
2. De uträknade poängen skall ligga lagrade i systemet.

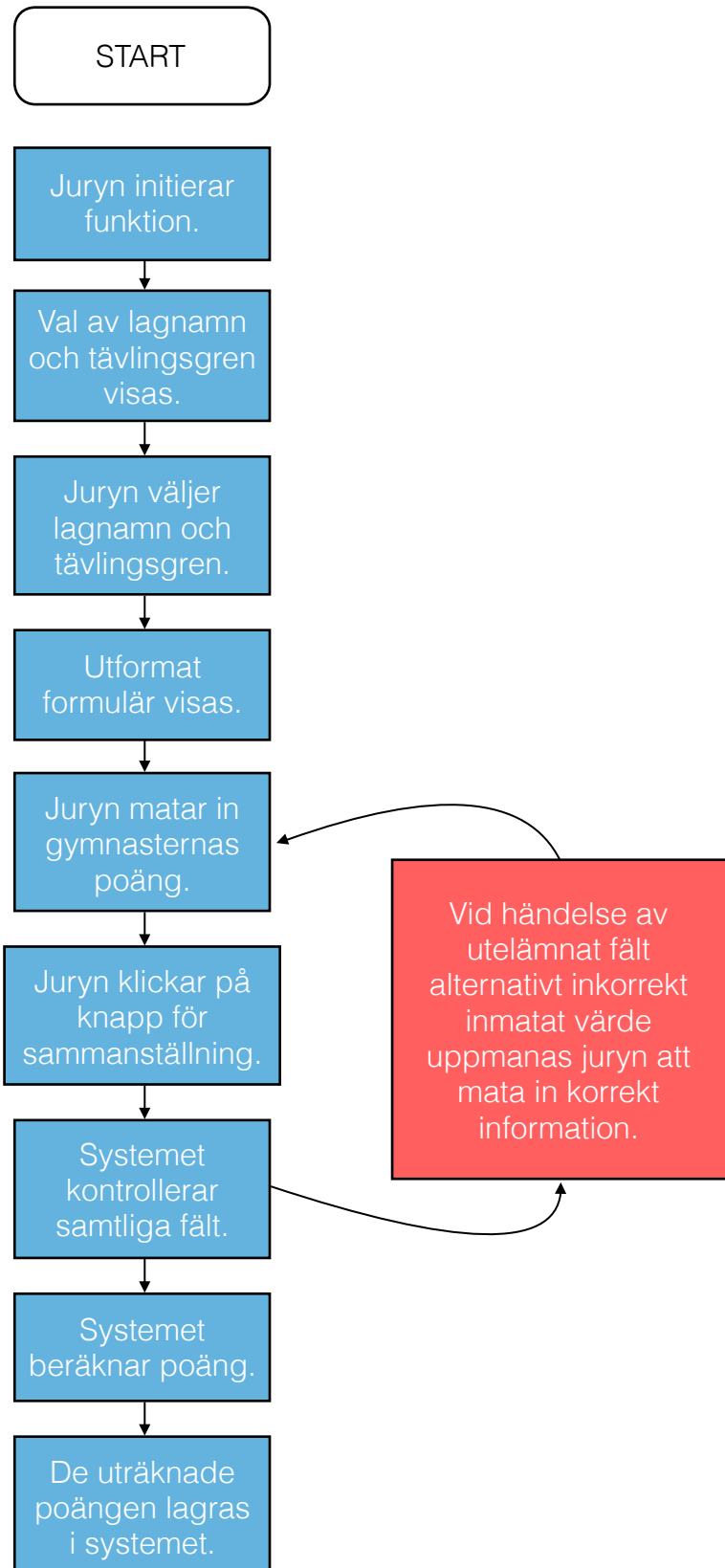
### PRIMÄRT FLÖDE

1. Juryn navigerar sig till och initierar funktionen för poänginmatning.
2. Fönster med val av lagnamn och tävlingsgren visas för juryn.
3. Juryn väljer lagnamn och tävlingsgren.
4. Utformat formulär för inmatning av gymnasternas poängbedömningar visas för juryn.
5. Juryn matar in gymnasternas poäng i de givna fälten i formuläret.
6. När samtliga poäng är inmatade klickar juryn på knapp för sammanställning.
7. Systemet kontrollerar samtliga fält.
8. Systemet beräknar gymnasternas medelvärde och lagets poäng.
9. De uträknade poängen med tillhörande information rörande lagnamn och tävlingsgren lagras i systemet.

### SEKUNDÄRT FLÖDE

- 7A. Vid händelse av utelämnat fält alternativt inmatning av negativt värde, värde utanför poängskalan eller ogiltiga tecken markeras detta och juryn uppmanas mata in korrekt information.

## FLÖDESDIAGRAM



## KLASSER MED BESKRIVNINGAR

**UserAccount**  
Class

Fields

authority

logged

Properties

Authority

Logged

Methods

UserAccount (+ 1 overload)

### UserAccount

Klassens uppgift är att validera huruvida användaren är inloggad i systemet och således tillåten att initiera funktionen. Beroende på användarens status skrivs lämpliga meddelanden ut. I denna iteration nyttjas enbart fältet *logged* med dess tillhörande egenskap *Logged*.

Fältet *authority* med dess tillhörande egenskap *Authority* beskrivs i den tredje och avslutande iterationen.

**Form**  
Class

Fields

givenPoint

gymnastPoint1

gymnastPoint2

gymnastPoint3

gymnastPoint4

gymnastPoint5

Properties

GivenPoint

GymnastPoint1

GymnastPoint2

GymnastPoint3

GymnastPoint4

GymnastPoint5

Methods

CalculateAveragePoint

CalculateTeamPoint

Form (+ 1 overload)

SetColor

### Form

Klassens uppgift är att dels validera värdena på de poäng juryn tilldelar varje enskild gymnast, och dels beräkningar gällande medelpoäng och lagpoäng.

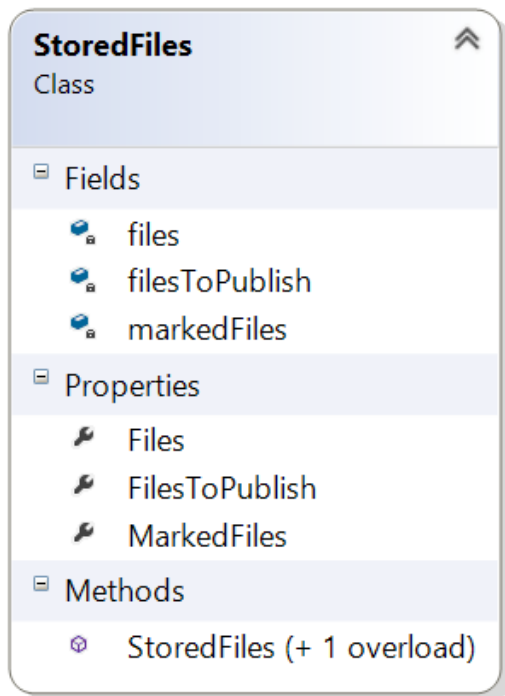
Det övre fältet/egenskapen representerar en domares poängtilldelning för enskild gymnast. Den validerar och fastställer en domares givna poäng för varje specifik gymnast och gren. Värdet på inmatat poäng får endast vara 1-10. Vid inmatning av otillåtet värde kastas ett undantag med beskrivande felmeddelande.

De nedre fem fälten/egenskaperna representerar (beroende på testfall) dels en gymnasts fem tilldelade poäng i en specifik gren, och dels fem gymnasters redan uträknade medelpoäng. Detta var ett medvetet val jag gjorde för att spara in antalet variabler och mängden kod. I det förstnämnda fallet tilldelas en gymnast fem poäng för att därefter validera värdena samt kalla på metod för uträkning av medelpoäng. I det sistnämnda fallet utgår jag ifrån redan tillsatta medelpoäng och kallar därefter på metod för uträkning av lagpoäng.

*CalculateAveragePoint* räknar ut en gymnasts medelpoäng.

*CalculateTeamPoint* räknar ut lagpoängen.

*SetColor* fastställer textfärg i consolfönstret.



## StoredFiles

Klassens uppgift är att validera huruvida ett formulär med resultat har skickats in och lagrats i systemet. I denna iteration nyttjas enbart fältet *files* med dess tillhörande egenskap *Files*.

I denna klass har jag fastställt en gräns på maximalt 100 000 lagrade filer. Vid händelse av att användaren försöker skicka in ett dokument som skulle passera denna gräns kastas ett undantag med beskrivande felmeddelande.

De resterande fälten med tillhörande egenskaper beskrivs i den tredje och avslutande iterationen.

# ENHETSTESTNING

## TESTSVIT

### Enhetstest för metoderna i klassen UserAccount

Testfall	Vad skall testas	Testdata	Förväntat resultat	Resultat
Lyckad initiering av funktion.	Kontrollera om användaren är inloggad.	logged = true	Initiering lyckas.	Initiering lyckas.
Misslyckad initiering av funktion.	Kontrollera om användaren är inloggad.	logged = false	Initiering misslyckas.	Initiering misslyckas.

### Enhetstest för metoderna i klassen Form

Testfall	Vad skall testas	Testdata	Förväntat resultat	Resultat
Lyckad tilldelning av poäng.	Kontrollera att värdet ligger inom tillåtet intervall.	givenPoint = 8	Tilldelning lyckas.	Tilldelning lyckas.
Misslyckad tilldelning av poäng #1.	Kontrollera att värdet ligger inom tillåtet intervall.	givenPoint = 12	Tilldelning misslyckas.	Tilldelning misslyckas. Felmeddelande skrivs ut.
Misslyckad tilldelning av poäng #2.	Kontrollera att värdet ligger inom tillåtet intervall.	givenPoint = 0	Tilldelning misslyckas.	Tilldelning misslyckas. Felmeddelande skrivs ut.
Lyckad tilldelning av poäng, och uträkning av medelpoäng.	Kontrollera att poängfälten tilldelats korrekta värden, och beräkning av medelvärde initieras och utförs.	gymnastPoint1 = 9 gymnastPoint1 = 3 gymnastPoint1 = 5 gymnastPoint1 = 7 gymnastPoint1 = 5	Tilldelning och kontroll av fält lyckas. Medelvärde räknas ut och presenteras.	Tilldelning och kontroll av fält lyckas. Medelvärde räknas ut och presenteras.  Medelvärde blir 5.
Misslyckad tilldelning av poäng, och därav uträkning av medelpoäng.	Kontrollera att poängfälten tilldelats korrekta värden, och beräkning av medelvärde initieras och utförs.	gymnastPoint1 = 9 gymnastPoint1 = 0 gymnastPoint1 = 0 gymnastPoint1 = 7 gymnastPoint1 = 11	Tilldelning och kontroll av fält misslyckas. Medelvärde beräknas inte.	Tilldelning och kontroll av fält misslyckas. Medelvärde beräknas inte.  Felmeddelande skrivs ut.
Lyckad tilldelning av medelpoäng, och uträkning av lagpoäng.	Kontrollera att poängfälten tilldelats korrekta värden, och beräkning av lagpoäng initieras och utförs.	gymnastPoint1 = 7 gymnastPoint1 = 4 gymnastPoint1 = 2 gymnastPoint1 = 9 gymnastPoint1 = 6	Tilldelning och kontroll av fält lyckas. Lagpoäng räknas ut och presenteras.	Tilldelning och kontroll av fält lyckas. Lagpoäng räknas ut och presenteras.  Lagpoäng blir 28.

Misslyckad tilldelning av medelpoäng, och därav uträkning av lagpoäng.	Kontrollera att poängfälten tilldelats korrekta värden, och beräkning av lagpoäng initieras och utförs.	gymnastPoint1 = 0 gymnastPoint1 = 4 gymnastPoint1 = 12 gymnastPoint1 = 9 gymnastPoint1 = 6	Tilldelning och kontroll av fält misslyckas. Lagpoäng beräknas inte.	Tilldelning och kontroll av fält misslyckas. Lagpoäng beräknas inte. Felmeddelande skrivs ut.
--	---	--	--	---

### Enhetstest för metoderna i klassen StoredFiles

Testfall	Vad skall testas	Testdata	Förväntat resultat	Resultat
Lyckad lagring av dokument i systemet.	Kontrollera att dokument lagras korrekt genom att öka "databas-variabel" med 1.	files = 99999 + 1 (files = "databas-variabel")	Lagring av dokument lyckas. Bekräftande meddelande skrivs ut.	Lagring av dokument lyckas. Bekräftande meddelande skrivs ut.
Misslyckad lagring av dokument i systemet.	Kontrollera att dokument lagras korrekt genom att öka "databas-variabel" med 1.	files = 100000 + 1 (files = "databas-variabel")	Lagring av dokument misslyckas. Felmeddelande skrivs ut.	Lagring av dokument misslyckas. Felmeddelande skrivs ut.

## MOTIVERING

Testsviten är utformad på ett tydligt och överskådligt vis, som vid eventuell utökning av exempelvis domare och/eller gymnaster dessutom möjliggör för vidare påfyllning av ytterligare enhetstester och testfall. Testsviten innehåller tre enhetstester med tillhörande testfall vilka verifierar och dokumenterar samtliga fält, egenskaper och metoder av mitt tillämpade användningsfall. Testsvitens samtliga testfall består av fem kolumner som innehåller beskrivande och förklarande information kring vad som skall utföras i varje enskilt test. Planerad testdata presenteras samt förväntade och slutgiltiga resultat. De dokumenterade testfallen anser jag, utefter den grad formuläret är utformat, kontrollerar flertalet tänkbara utfall av de metoder som finns med i denna iterations användningsfall.

## REFLEKTION

I denna andra iteration beslutade jag att vidareutveckla mitt användningsfall för inmatning av poäng. Likt min föregående iteration var strukturen i flödesschemat och flödesdiagrammet för detta användningsfall redan väldokumenterade, så det krävdes enbart en vidareutveckling av vad exakt som skall tillåtas av en domare att mata in i systemet. Validering huruvida en användare är inloggad eller inte återkommer även i denna iteration. Eftersom detta är ett PRE-villkor för att kunna nyttja funktionen ansåg jag det nödvändigt att implementera en sådan kontroll. Jag har begränsat de värden en domare kan mata in för en gymnast. Skalan jag använder mig av är 1-10, således kan exempelvis inte negativt tal inmatas. Utöver denna begränsning har jag även infört en fullständig kontroll av inmatade fält innan uträkning av såväl medelpoäng som lagpoäng initieras. Jag beslutade att använda mig av samma fem variabler för mina uträkningar av dessa medel- och lagpoäng, då det annars hade resulterat i alldeles för många deklarerade variabler och upprepning av kod. Avslutningsvis kontrollerar jag huruvida dokumentet innehållandes resultaten lagras korrekt i systemet. Jag deklarerar en variabel i egen klass vilken jag låter agera "databas-variabel", och som jag vid slutförd kontroll av inmatade fält ökar med ett. Dock har jag begränsat antal lagrade filer till 100 000 vilket leder till att ett felmeddelande kastas vid händelse av att detta tal passeras. Denna kontroll valde jag att införa då det dels står som ett POST-villkor i mitt flödesschema, men också för att klassen kommer nyttjas till den avslutande iterationen. Sammanfattningsvis är jag nöjd över min specificerade planering för denna iteration. Jag granskade den föregående iterationens planering och utförde lämpliga justeringar utifrån den samt erfarenheter. Detta resulterade i att jag slutförde min iteration en halvtimma tidigare än planerat.