

## Planera

### Uppgift 2

Kolla upp hur testplaner ser ut och om det finns eventuella mallar.

5 timmar

### Uppgift 3

Låna arbete från laboration 2 och implementera koden i php.

Edit: började om hela koden för att utföra den i c# då det är lättare att genomföra uppgifterna senare.

6 timmar

### Uppgift 4

Hitta information och eventuella mallar för unit testning och testsviter

6 timmar

### Uppgift 5

Leta upp exempel på testkoder till c#

4 timmar

### Uppgift 6

Väldigt oklar uppgift men verkar som att det är en upprepande uppgift

4 timmar

### Uppgift 7

1 timme

Total planerad tid: 28 timmar.

## Test plan

### Syfte

Huvudsyftet med testplanen är att testa systemet som Gymnastikligan använder och försöka uppfylla deras krav.  
Samt att beskriva vilka test metoder som kommer att användas.

### Test uppdrag

Testerna kommer omfatta funktionella-, prestanda- och säkerhetskrav i användningsfallen.

#### Mål

Målet är att skapa en testplan om hur testarbetet ska genomföras på ett strategiskt sätt och därmed kunna identifiera eventuella risker och eliminera dem. Målet är även att få ett system med bra prestanda samt att det är funktionellt.

#### Risker

Systemet innehåller känslig information om privatpersoner och därmed kommer systemet att kräva en hög grad av säkerhetstänk för att inte kränka våra intressenter eller bryta mot personuppgiftslagen.  
Systemet innehåller även resultatpoäng som kräver hög säkerhet för att undvika obehöriga att ändra poängställningen.

### Teststrategi

Testerna kommer att utföras manuellt i form av dynamisk och statisk testning  
Inga automatiserade system finns att tillgå till projektet.

#### Kvalitetsegenskaper

För att uppnå önskad kvalitet kommer testerna att följas med en checklista för att inte glömma eller lämna kvar några fel i systemet.

#### Enhetstester

För att enklare hitta fel i programmet kommer spårnings koder och loggning läggas in bland koderna.

## **Testprocess**

Testerna sker löpande tillsammans med utvecklingen av projektet för att enkelt och snabbt kunna hitta de buggar som uppstår i systemet. Annars kan eventuella små problem i systemet växa till större och blir svårare att åtgärda.

## **Testmiljö**

Testning av de senaste webbläsarna och operativsystemen för att kunna kolla användarupplevelserna samt funktionalitet av systemet.

## **Dokumentering**

Alla testresultat dokumenteras skriftligt i statusrapporter som i slutändan kommer samlas i en slutrapport. Slutrapporten kommer vara tillgänglig för de som jobbar med projektet

## **Kriterier för godkända test.**

Ett test anses vara godkänt om de stämmer överens med förväntade resultat i dokumentationen. Om de inte stämmer överens är testet inte godkänt enligt våra kriterier.

## Användningsfallsmodell

### Aktörer:

#### Gymnastik klubbar

De olika klubbföreningarna kan besöka webbsidan och registrera sig genom ett formulär som de ska fylla i och skicka iväg.

De kan registrera nya lag med medlemmar samt registrera vilka grenar, ålder och kön de tillhör.

De kan även ta bort lag som de har registrerat.

#### Lag ledare

De vägleder och ger ut information till gymnasterna.

De har samma användningsfall i systemet som gymnastik klubbarna.

#### Sekreterare

Handhåller Tävlingspoängen och räkna ut medelpoängen.

Sekreteraren kan skriva in och ändra poängen i systemet.

#### Administratör

Administratören har behörighet i hela systemet och har i uppgift att uppdatera och redigera scheman, lagen, statistiken samt användarna i systemet.

De bekräftar även nya registreringar som kommer in.

#### Huvud domare

Huvuddomarens uppgift är att informera de andra domarna vilka lag de ska döma samt vilka tider som gäller.

Huvuddomaren kan logga in och ändra i schemat, exempelvis vilka domare som ska döma vilka lag samt tider.

#### Domare

Domarna har till uppgift att tilldela poäng till gymnasterna och ser till att reglerna följs. De har också till uppgift att lämna över poängen till sekreteraren.

#### Gymnasterna

Gymnasterna kan inte registrera sig i systemet utan kan bara gå in och kolla scheman, statistik, namn på domare och namn på lag som ska tävla.

## Användningsfalls beskrivningar

### Registrera användare.

Registreringen är ett formulär som fylls i på webbsidan och lagras i databasen men innan användaren kan logga in måste administratören godkänna registreringen och dela ut specifika rättigheter till användaren. När administratören har bekräftat registreringen skickas ett mail till användaren att registreringen är genomförd. Vid registrering finns det ett par fält användaren måste fylla i korrekt.

För och efternamn

- Lämpligt reguljärt uttryck för att kolla om det är ett riktigt namn.
- Får inte lämnas blankt.

Personnummer

- Kolla så personnumret är valid.

Vilken typ av roll

- En lista med olika roller exempelvis lagledare, sekreterare.
- Ett val är ett krav.

Mail

- Kolla så angiven mail inte redan finns.
- Lämpligt reguljärt uttryck för att kolla om det är en riktig mail.

Lösenord

- Minst antal tecken på lösenordet.
- Special tecken eller siffra ska ingå.
- Skriva in lösenordet igen

Namn på klubben

- Får inte lämnas blankt.

Ett robotfilter

- Kollar att robotfiltret är rätt ifyllt.

Ifall användaren hoppa över eller missar ett fält kommer fältet att indikeras med en röd list runt fältet.

Om något av dessa fält inte fylls i korrekt kommer formuläret inte skickas iväg och en ruta med text kommer informera användaren vart felet är och ligger. Användaren kan sedan rätta till eventuella fel och skicka iväg formuläret på nytt.

## **Beskriv pre- och postvillkor, primära och alternativa flöden**

### **Previllkor:**

- Användaren har gått in på webbsidan och hittat registrerings länken.

### **Postvillkor:**

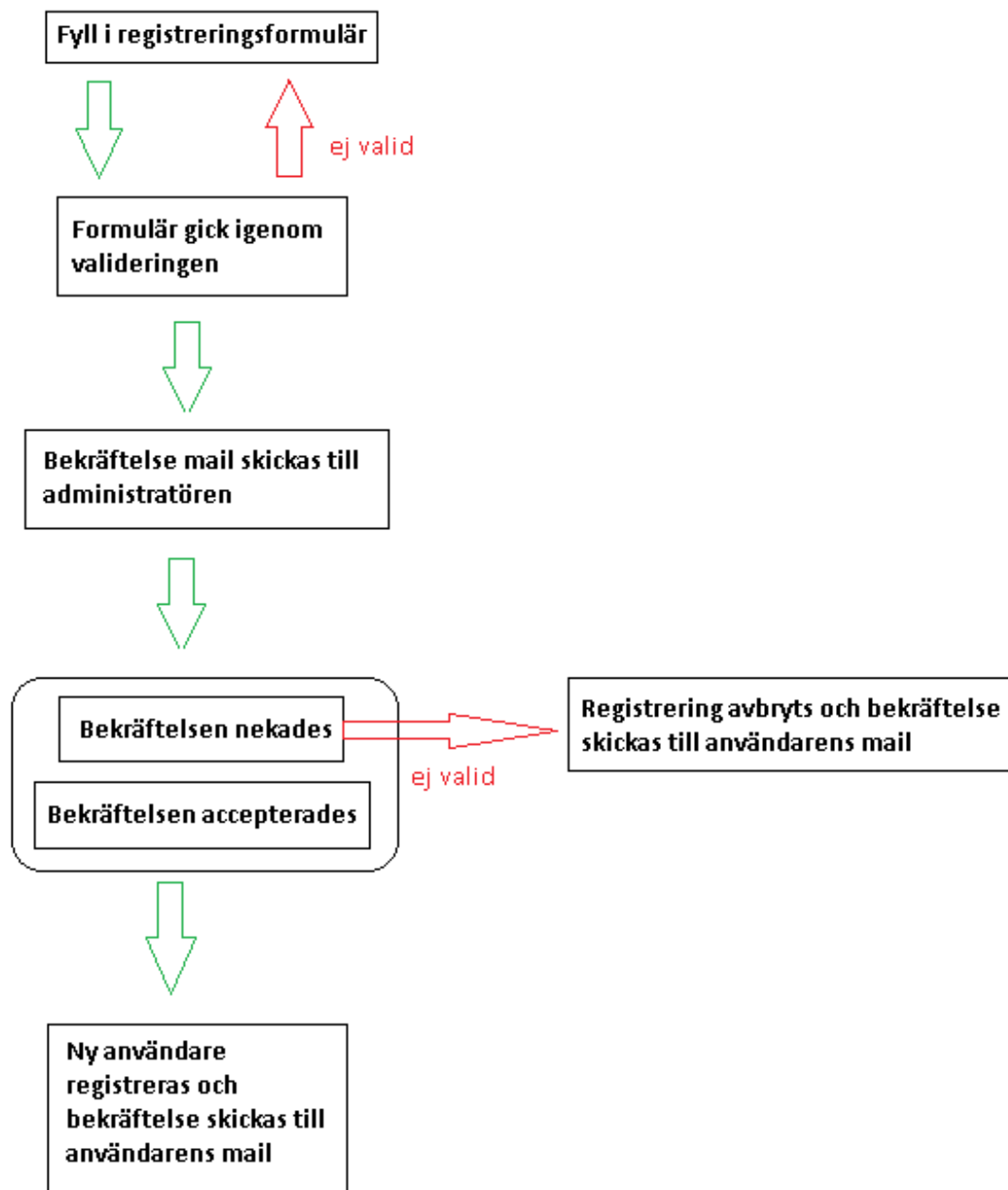
- Registrering av användare är lyckad.
- Alla data som är ifyllt har skickats till databasen.

### **Primära:**

- Bekräftelse på en lyckad registrering (Den perfekta världen)
- Användaren får en bekräftelse till sin mail.

### **Alternativa flöden:**

- Fält som inte är korrekt ifyllt visas genom ett meddelande.
- Fält som är tomma indikeras.



## **Registrera poäng.**

Sekreterarna kan logga in på webbsidan och lägga in poäng i systemet.

Genom att gå in på kontrollpanelen visas en lista med alla lagen i bokstavsordning. Listan går även att filtrera genom kön, åldersgrupp och gren, det går även att filtrera bort flera alternativ.

En sökfunktion finns även att tillgå som kan söka efter lag och medlemmar.

Genom att klicka på ett lag i listan genereras en lista upp med lagets namn.

I listan med lagets medlemmar går det att lägga in poäng medlemsvis genom att klicka på "lägg till poäng" som befinner sig till höger vid medlemens namn.

Vid klick på "lägg till poäng" knappen genereras ett fält som man kan skriva in poängen i men inte genom text från tangentbordet utan man får klicka sig fram genom pilar.

Genom att klicka sig fram genom pilarna blir det svårare att skriva in fel så som text eller punkter och bindestreck.

Poängen läggs sig bredvid "lägg till poäng" knappen som kommer ändras till "ändra poäng" och om sekreteraren väljer att klicka på knappen igen kommer det nuvarande poängen att ändras.

De poäng som har lagts till hos personerna kommer läggas till i lagets totala poäng.

## **Beskriv pre och post villkor, primära och alternativa flöden**

Pre villkor:

- Sekreteraren har registrerat sig i systemet
- Administratören har godkänt bekräftelsen
- Sekreteraren har gått in på systemet och navigerats till registrera poäng sidan

Post villkor:

- Registrering av poäng är lyckad
- Alla data som är ifyllt har skickats till databasen

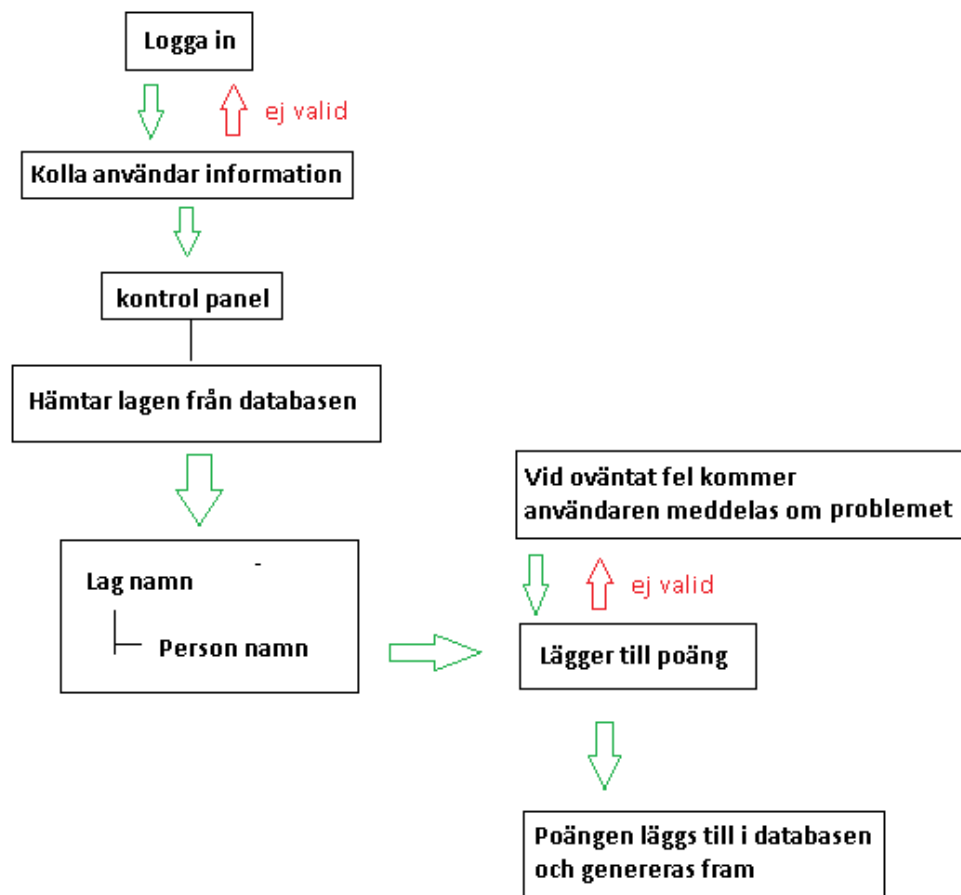
Primära

- Bekräftelse på en lyckad registrering (Den perfekta världen)
- Användaren får en bekräftelse

Alternativa flöden

- Fält som inte är korrekt ifyllt visas genom ett meddelande
- Fält som är tomma indikeras





Klasser med beskrivningar

## Klass Register

Klassen är avsedd för aktörer som registrera sig i systemet innehållande fält, metoder, egenskaper och konstruktorer.

### Fält

**string \_username;**

Innehåller namn på användaren.

**string \_password;**

Innehåller användarens lösenord.

**string \_email;**

Innehåller användarens email.

### Egenskaper

#### String Username

Ger fältet \_username ett värde av typen string.

#### String Password

Ger fältet \_password ett värde av typen string samt går igenom metoden passValid().

#### String Email

Ger fältet \_email ett värde av typen string samt går igenom metoden emailValid().

### Konstruktorer

#### Register()

Skapa ett nytt objekt av klassen Register.  
Inparametrar är samtliga fält.

### Metoder

#### emailValid()

Kollar om \_email är en valid mail genom reguljära uttryck.

#### passValid()

Kollar så att \_password innehåller mellan 4 till 8 tecken samt inkludera minst en siffra genom reguljära uttryck.

#### Registered()

Kollar så alla fält har fått ett värde tilldelat och där med returnera true annars false för att se om registreringen har lyckats.

### Klass Secretary

Klassen är avsedd för sekreterare som registrera poäng i systemet innehållande fält, metoder, egenskaper och konstruktorer.

## **Fält**

**string \_name;**

Innehåller namn på gymnasten.

**double \_score;**

Innehåller gymnastens poäng.

**double \_teamScore;**

Innehåller lagets poäng.

## **Egenskaper**

**String Name**

Ger fältet \_name ett värde av typen string.

**double Score**

Ger fältet \_score ett värde av typen double.

## **Konstruktorer**

**Secretary()**

Skapa ett nytt objekt av klassen Secretary.

Inparametrar är fälten \_name och \_score

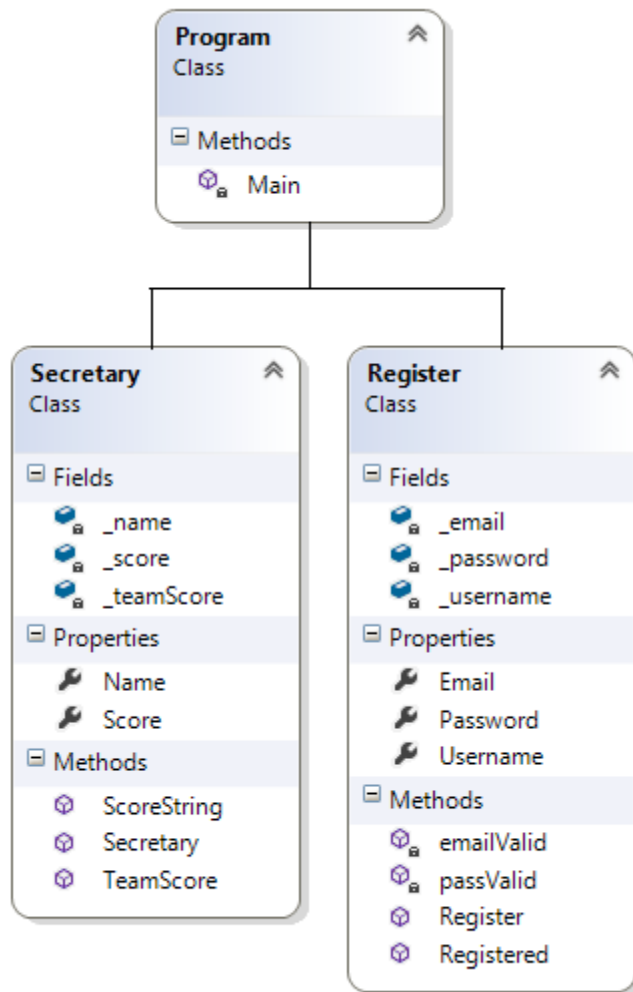
## **Metoder**

**ScoreString()**

Returnera fälten \_name samt \_score

**TeamScore()**

Lägger till gymnastens poäng \_score i lagets poäng \_teamScore och returnera \_teamScore



## Enhetstestning

### Testdokumentation

Testningen går ut på att säkerställa att alla delar i klassen fungera som planerat och uppnår dokumenterade krav. Testningen kommer även se till att minimera eventuella buggar.

De klasser som ska testas är Register och Secretary.

Testsviterna som kommer användas är testsvit 1 och testsvit 2 nedanför.

### Testsvit 1- Register

Typ av test	Testdata	Metod	Kommentar	Resultat
Kontroll av inloggning	Namn = "Åke" Lösenord = "pass12" Email ="roy.lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktör som innehåller tre parametrar.	Förväntat resultat. Kräver valid email	
Kontroll av email	Email ="roy@lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktör som innehåller parametern email.	Förväntat resultat email är valid	
Kontroll av inloggning	Namn = "Roy" Lösenord = "Brun1" Email ="roy@lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktör som innehåller tre parametrar.	Förvänta att testet godkänns	
Kontroll av inloggning	Namn = "" Lösenord = "" Email = ""	Skapa en instans av klassen Register och skapa en ny konstruktör som innehåller tomma parametrar.	Förväntat resultat Kräver ifyllda parametrar	

## Testsvit 2- Secretary

Typ av test	Test data	Metod	kommentar	Resultat
Lägga till poäng	Namn = "Roy" Poäng = 5	Skapa en instans av klassen Secretary och skapa en ny konstruktör som innehåller två parametrar.	Förväntat reultat Testet godkänns	
Lägga till poäng	Namn = "Roy Lnu" Poäng = 10.4	Skapa en instans av klassen Secretary och skapa en ny konstruktör som innehåller två parametrar innehållande decimal tal och efternamn	Förväntat reultat Testet godkänns	
Testar metoderna	Namn = "" Poäng = 0	Skapa en instans av klassen Secretary och skapa en ny konstruktör som innehåller två parameter var den ena har tomt namn	Finns inga regler för tomma inparamtrar Förväntar godkänt	
Lägga till poäng	Namn = "3" Poäng = "lnu"	Skapa en instans av klassen Secretary och skapa en ny konstruktör som innehåller två parametrar i fel ordning.	Förväntat resultat Kan inte använda string i Poäng eller siffror i Namn	

### Sammanfattning.

Klasserna är uppdelade sviter och de testas främst genom deras konstruktörer samt metoder för de täcker funktionaliteten i hela klassen/klasserna.

## Implementera testsviten och kör

### Dokumentation.

Jag har valt att skapa två klasser med namnen TestRegister samt TestSecretary för vardera användningsfall som innehåller olika tester. Testerna körs och returnera om testet är godkänd eller inte godkänt genom en try- catch sats.

Samtliga tester är inkapslade i en metod som innehåller en try- catch sats för att utgöra om testet blir godkänt eller inte och där efter hämtar en metod som färgkodar testerna genom boolenskt värde sant eller falskt.

Testerna ger en bra visuell bild av samtliga tester i konsol fönstret.

### Förbättringar

Det som kan förbättras i koden är ett reguljärt uttryck till namn metoden i poäng klassen eller en if sats som inte godkänner tom sträng.

Om poängen få ge värdet 0 kan jag godkänna då det är möjligt att få 0 poäng i de flesta sporter.

### Testsvit 1- TestRegister

Typ av test	Test data	Metod	kommentar	Resultat
Kontroll av inloggning	Namn = "Åke" Lösenord = "pass12" Email ="roy.lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktor som innehåller tre parametrar.	Förväntat resultat. Kräver valid email	Misslyckat
Kontroll av email	Email ="roy@lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktor som innehåller parametern email.	Förväntat resultat Kollar om email är valid	Godkänt
Kontroll av inloggning	Namn = "Roy" Lösenord = "Brun1" Email ="roy@lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktor som innehåller tre parametrar.	Förväntat resultat	Godkänt
Kontroll av inloggning	Namn = "" Lösenord = "" Email = ""	Skapa en instans av klassen Register och skapa en ny konstruktor som innehåller tomma parametrar.	Förväntat resultat Kräver ifyllda parametrar	Misslyckat

### Testsvit 2- TestSecretary

Typ av test	Test data	Metod	kommentar	Resultat
Lägga till poäng	Namn = "Roy" Poäng = 5	Skapa en instans av klassen Secretary och skapa en ny konstruktor som innehåller två parametrar.	Förväntat reultat	Godkänt
Lägga till poäng	Namn = "Roy Lnu" Poäng = 10.4	Skapa en instans av klassen Secretary och skapa en ny konstruktor som innehåller två parametrar innehållande decimal tal och efternamn	Förväntat reultat	Godkänt
Testar metoderna	Namn = "" Poäng = 0	Skapa en instans av klassen Secretary och skapa en ny konstruktor som innehåller två parametrar var den ena har tomt namn	Åtgärd av tomma namn fält	Godkänt
Lägga till poäng	Namn = "3" Poäng = "lnu"	Skapa en instans av klassen Secretary och skapa en ny konstruktor som innehåller två parametrar i fel ordning.	Förväntat reultat Kan inte använda string i Poäng eller siffror i Namn	Misslyckad





## Integrationstestning

### Stegvis beskrivning för hur du arbetade fram testfallen och testdatan.

För att registrera en användare krävs det att valid data av namn-, lösenord-, och email matas in,

eller för poängregistrering krävs valid data för namn och poäng.

Jag valde att kontrollera hur registrering och poängregistrering betede sig när de fick data inmatade som systemet skulle godkänna och inte godkänna.

Jag delade upp de olika testfallen i klasser (Registrering, poängregistrering) och kappslade in varje testfall i en metod och skapade en ny instans objekt av varje testfall.

För att kontrollera om testfallet är godkänt eller inte godkänt använder jag mig av try-catch satser för att få ut ett resultat med hjälp av en annan metod som returnera boolesk värde sant eller falskt.

Alla reultat renderas ut i konsolfönstret.

En visuel bild av hur en testmetod ser ut.

```
1reference  
public static void TestEmail()  
{  
    var person2 = new Register();  
    try  
    {  
        person2.Email = "roy@hotmail.com";  
        Message("Test lyckades! Email lades till", false);  
    }  
    catch (Exception)  
    { Message("Test misslyckades", true); }  
}
```

Bilden testar om en ny användare kan mata in email adressen [roy@hotmail.se](mailto:roy@hotmail.se)

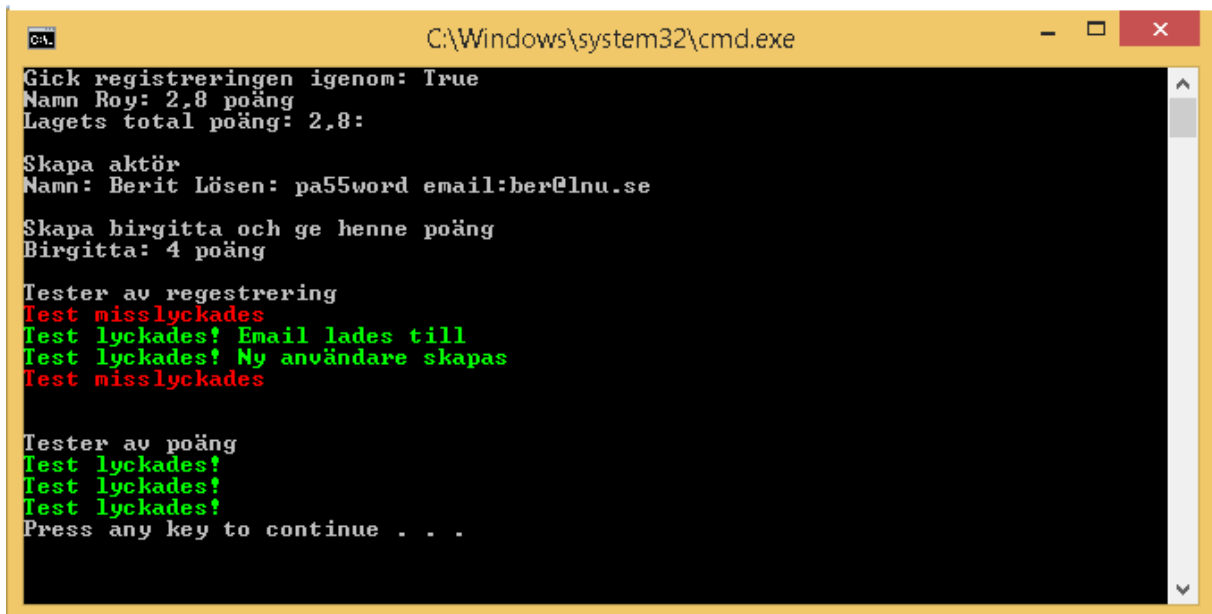
Testet blev godkänt.

Resterande tester är lika med olika typer av data.

Samtliga tester hämtar metoden Message för att få rätt färgbeteckning.

```
static void Message(string message, bool Error)
{
    if (Error)
    {
        Console.ForegroundColor = ConsoleColor.Red;
    }
    else
    {
        Console.ForegroundColor = ConsoleColor.Green;
    }
    Console.WriteLine(message);
    Console.ResetColor();
}
```

En visuell bild av alla tester som ligger i samma ordning som testsviterna.



```
C:\Windows\system32\cmd.exe
Gick registreringen igenom: True
Namn Roy: 2,8 poäng
Lagets total poäng: 2,8:

Skapa aktör
Namn: Berit Lösen: pa55word email:ber@lnu.se

Skapa birgitta och ge henne poäng
Birgitta: 4 poäng

Tester av registrering
Test misslyckades
Test lyckades! Email lades till
Test lyckades! Ny användare skapas
Test misslyckades

Tester av poäng
Test lyckades!
Test lyckades!
Test lyckades!
Press any key to continue . . .
```

## Test Data

### Test av skapa nya instanser från båda användningsfallen

Typ av test	Test data	Metod	kommentar	Resultat
Skapa gymnast och ge henne namn och poäng	Namn = "Birgitta" Poäng = 4.0	Skapa en instans av klassen Secretary och skapa en ny användare och tilldela användaren poäng	Förväntat resultat.	Godkänt
Skapa en ny aktör och tilldela samtliga fält	Namn = "Berit" Lösenord = "pa55sword" Email = "ber@lnu.se"	Skapa en instans av klassen Register och skapa en ny aktör och tilldela aktören värden	Förväntat resultat.	Godkänt

### Testsvit 1- Register

Typ av test	Test data	Metod	kommentar	Resultat
Kontroll av inloggning	Namn = "Åke" Lösenord = "pass12" Email = "roy.lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktör som innehåller tre parametrar.	Förväntat resultat. Kräver valid email	Misslyckat
Kontroll av email	Email = "roy@lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktör som innehåller parametern email.	Förväntat resultat Kollar om email är valid	Godkänt
Kontroll av inloggning	Namn = "Roy" Lösenord = "Brun1" Email = "roy@lnu.se"	Skapa en instans av klassen Register och skapa en ny konstruktör som innehåller tre parametrar.	Förväntat resultat	Godkänt
Kontroll av inloggning	Namn = "" Lösenord = "" Email = ""	Skapa en instans av klassen Register och skapa en ny konstruktör som innehåller tomma parametrar.	Förväntat resultat Kräver ifyllda parametrar	Misslyckat

## Testsvit 2- Secretary

Typ av test	Test data	Metod	kommentar	Resultat
Lägga till poäng	Namn = "Roy" Poäng = 5	Skapa en instans av klassen Secretary och skapa en ny konstruktör som innehåller två parametrar.	Förväntat reultat	Godkänt
Lägga till poäng	Namn = "Roy Lnu" Poäng = 10.4	Skapa en instans av klassen Secretary och skapa en ny konstruktör som innehåller två parametrar innehållande decimal tal och efternamn	Förväntat reultat	Godkänt
Testar metoderna	Namn = "" Poäng = 0	Skapa en instans av klassen Secretary och skapa en ny konstruktör som innehåller två parameter var den ena har tomt namn	Åtgärd av tomma namn fält	Godkänt
Lägga till poäng	Namn = "3" Poäng = "Inu"	Skapa en instans av klassen Secretary och skapa en ny konstruktör som innehåller två parametrar i fel ordning.	Förväntat reultat Kan inte använda string i Poäng eller siffror i Namn	Misslyckad

## Reflektion

Jag tyckte det var väldigt svårt att genomföra uppgifterna för de flesta begreppen var nya och man fick ständigt googla upp dem som inte var särskilt lätt. Det var inte heller helt lätt att veta om man har hittat rätt information då begreppen är väldigt omfattande.

Det var väldigt svårt att riktigt veta vad man skulle göra men med hjälp av Emil och klasskamrater kunde man få en bättre förklaring än vad uppgifterna beskrev.

Fortfarande saknar jag länkar till uppgifterna för att få en förklaring på vad man ska göra eller få en förklaring på vissa begrepp. Det samma gäller föregående laboration.

Uppgift 6 var väldigt luddigt beskrivet och måste beskrivas om till framtida studenter.

Val av programmering språk var fritt men uppgifterna blir väldigt svåra att tyda om man inte använder sig av språk som inte använder sig av klasser. Uppgifterna kanske kunde formuleras om så det passar andra språk bättre.