Design och implementationsdokument

Iteration 1

Användningsfall 1 - Samtliga aktörer registrerar ett konto

Jag kommer att skapa en klass som heter ”Register”. Denna klass kommer att innehålla en metod som heter VerifyRegister() och som tar tre parametrar - förnamn, efternamn och medlemstyp. Det kommer även finnas en metod som heter CreatePassword() som slumpar fram en sträng. Metoden VerifyRegister() fungerar som en administratör som returnerar inloggningsuppgifter till användaren om registreringen var rätt utförd. Om registreringen inte var rätt utförd så ombeds användaren försöka på nytt. Metoden skapar ett användarnamn av det ifyllda för och efternamnet och anropar CreatePassword() för att skapa ett nytt lösenord.

Klasser och metoder finns att se i Visual Studio-projektet ”Gymnastikligan 2014”.

Motivering
Jag tänkte att detta var ett smart sätt att göra på då inte tid finns till att designa ett ”riktigt” registreringssystem med en riktig administratör som godkänner registreringen och sedan mejlar ut inloggningsuppgifterna. När det gäller metoden VerifyRegister() anser jag att för och efternamn kan få vara precis vad som helst medan typ av medlem måste väljas korrekt, och det är just därför det är den parametern som undersöks i metoden. Metoden CreatePassword() valde jag att ha eftersom jag vill att nya lösenord alltid ska slumpas fram.

Iteration 2

Användningsfall 1 - Samtliga aktörer loggar in på sidan

Jag kommer att skapa en klass som heter ”Login”. Denna klass kommer att innehålla en metod som heter VerifyLogin() och som tar två parametrar - användarnamn och lösenord.
Metoden VerifyLogin() undersöker användarens ifyllda användarnamn och lösenord. Om de är korrekta så loggas användaren in, och om de inte är korrekta så ombeds användaren försöka igen.

Klasser och metoder finns att se i Visual Studio-projektet ”Gymnastikligan 2014”.

Motivering
Då jag i en tidigare labb har skapat en form av inloggningsscenario så var min första tanke att använda mig utav den, men eftersom jag har mer tid över från iteration 1 än planerat samt att jag själv tycker den lösningen blev lite grötig så gör jag nu på ett nytt sätt istället och försöker göra det mer tydligt. I denna laboration kommer användarnamnet ”användarnamn” och lösenordet ”lösenord” vara de enda inloggningsuppgifterna som fungerar som korrekta inloggningsuppgifter.

Iteration 3

Användningsfall 1 - Aktör lagledare registrerar ett lag

Jag kommer att skapa en klass som heter ”TeamRegister”. Denna klass kommer att innehålla en metod som heter VerifyTeamRegister() och som tar fyra parametrar – lagnamn, lagledarens namn, föreningens namn och ortsnamn.
Metoden VerifyTeamRegister() fungerar som en administratör som returnerar bekräftelseinformation till användaren om registreringen var rätt utförd. Om registreringen inte var rätt utförd så ombeds användaren försöka på nytt.

Klasser och metoder finns att se i Visual Studio-projektet ”Gymnastikligan 2014”.

Motivering
Anledning till att jag gör på detta sätt är för att registrera lag fungerar på liknande sätt som att registrera ett konto. Det är ett antal uppgifter som behöver fyllas i och det behöver skickas iväg till en administratör för godkännande. Metoden VerifyTeamRegister() godkänner alla fält så länge de inte är tomma.
Då programmet i nuläget inte kan särskilja vilken aktör det är som registrerar sig för ett konto så har jag valt att i detta skede lägga till ett tredje alternativ vid inloggningstillfället - ”logga in som lagledare”. Detta eftersom det var det bästa sättet jag kunde komma på för att simulera en inloggning från en aktör med lagledare-status utan att från grunden ha byggt ett sådant registreringssystem.