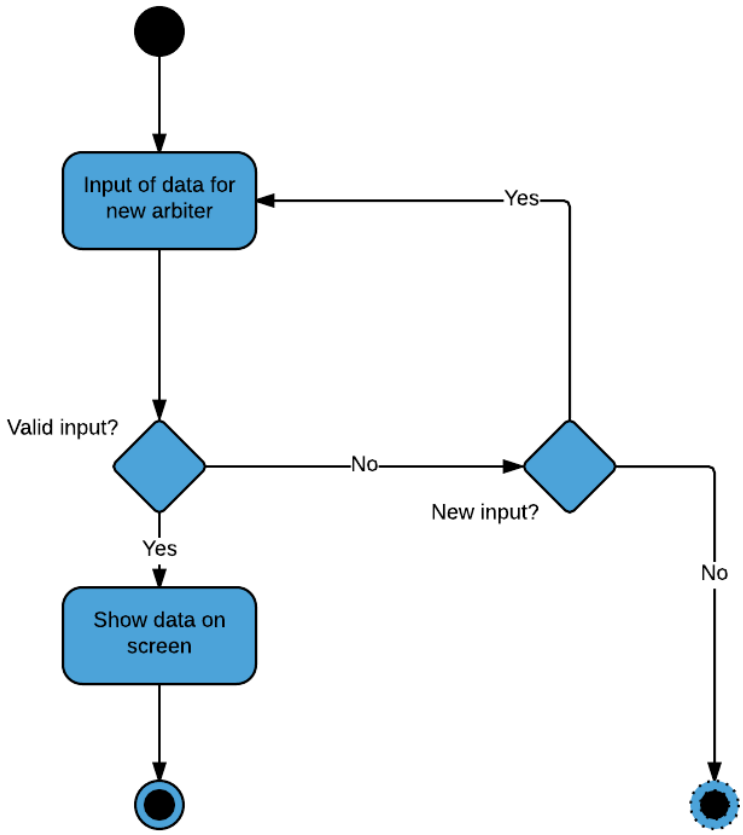
Uppgift 3 - Design och implementation

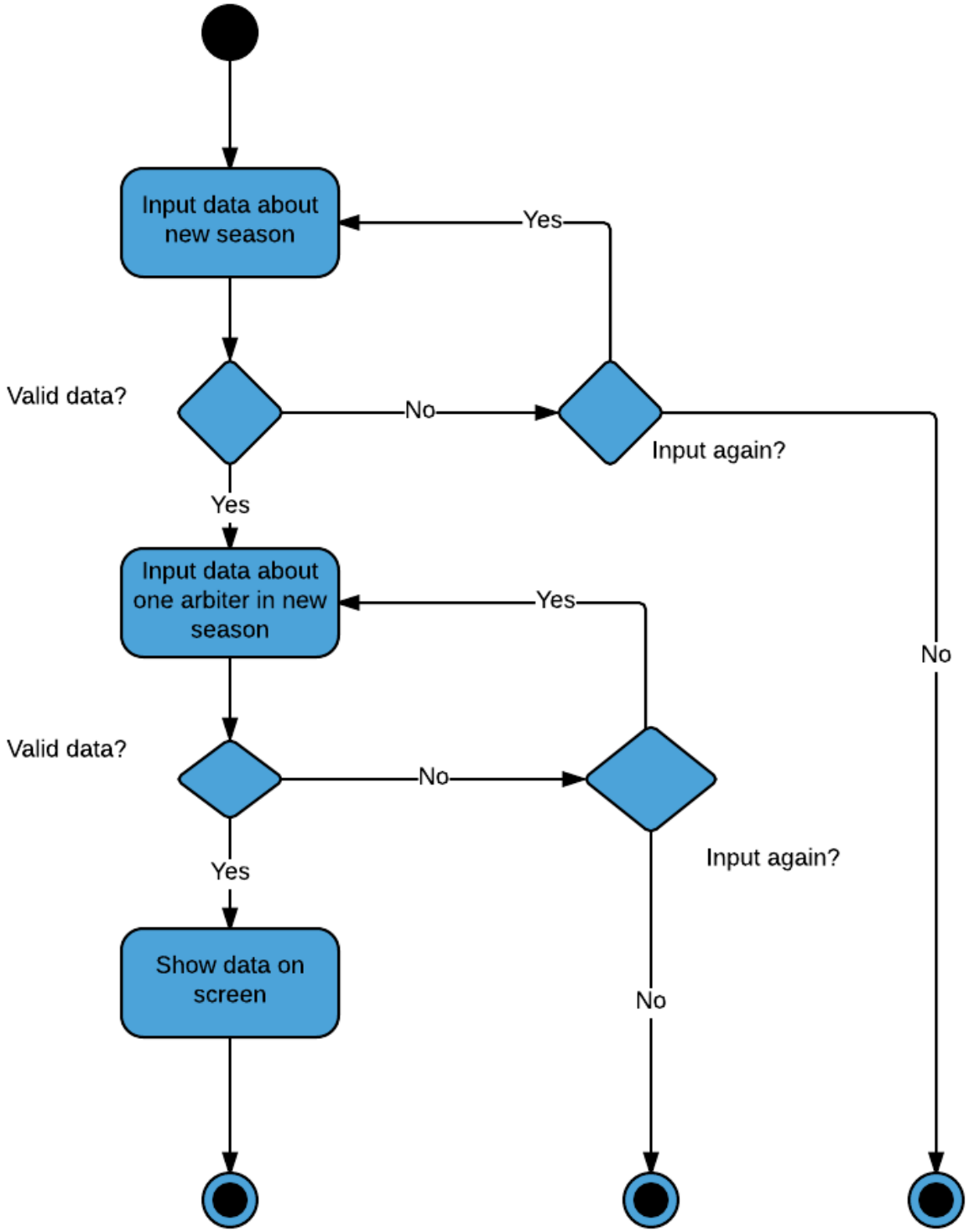
Jag har förändrat mina användningsfall från laboration 2 för att anpassa dessa till laboration 3. Jag har valt att skriva mina klasser och kod i C# för att jag behärskar detta språk bättre än de andra. Jag har dessutom gjort om alternativa flöde, eftersom de inte var korrekta i laboration 2.

Användningsfallsmodell

Användningsfall 4: Skriva in ny domare



Användningsfall 5: Registrera ny säsong



Användningsfall beskrivningar

Användningsfall 4: Skriva in ny domare
Aktör: Administratör
Objekt: Domare

Primärt flöde:

1. Administratör skriver in den nya domarens namn och efternamn
2. Systemet kräver av administratör att mata in en säsong
3. Systemet visar information om ny domare och tillhörande säsong på skärmen

Alternativt flöde:

1. Inmatning av domare gick fel (administratör skriver in ogiltigt värde)
2. Systemet kräver ny inmatning eller möjlighet att avsluta operation

Användningsfall 5: Registrera ny säsong
Aktör: Administratör
Objekt: Säsong

Primärt flöde:

1. Administratör lägger till en ny säsong med namn och år
2. Systemet sparar information om ny säsong i databasen
3. Systemet lägger till en domare för ny säsong
4. Systemet kontrollerar om domaren är behörig
5. Systemet visar information om ny säsong

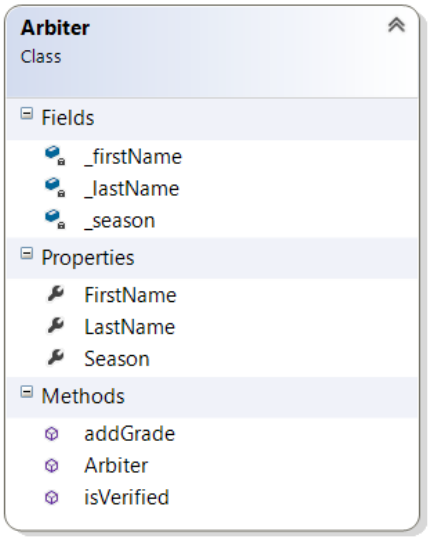
Alternativt flöde:

1. Domaren är inte behörig
2. Systemet meddelar administratör och kräver ny inmatning eller möjlighet att avsluta operationen.

Klasser med beskrivningar

Analys:

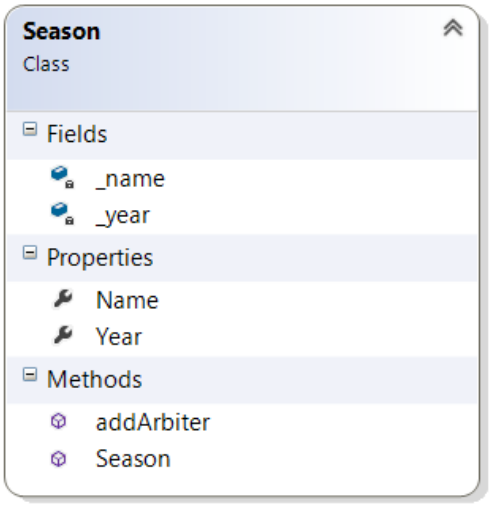
Artbiter klass



Användningsfall 4 har en enkel inmatning av ny domare i systemet. För att genomföra detta, är det enklast om en klass är *Domare*. Denna klass kommer antagligen att ärva från klassen *Person.* Jag har inte implementerat allt detta, för att behålla exempel enkelt. Själva inmatningen görs av administratör. Jag har behandlat bara en del av användningsfallet, och hoppat över delen när administratören loggar in i systemet. Därför valde jag att inte implementera klassen *Administratör* (som också skulle ärva från klassen *Person*). I mitt exempel, använder administratören vanlig *Program* klass för att registrera en ny domare. Domar klassen kan bli mer utvecklad, men jag har valt bara fält som är nödvändiga för en domare.

Klassen innehåller några metoder. En metod är konstruktör metoden, och andra metoder är metoder som hjälper en domare sätta ett betyg och en som avgör om domaren är behörig för att döma en tävling. Av vanliga fälten, innehåller klassen namn, efternamn och säsong. Första två hjälper att identifiera domaren, och sista att tillägga en säsong för domare (igen, förenklade jag lösningen från laboration 2).

Season klass



Användningsfall 5 är lite mer komplicerad än användningfall 4. För att registrera en säsong, måste man registrera namn och under vilket år tävlingar ska ske. Efter detta kan man tilläga domarna i ny säsong, men bara om domaren är behörig för den säsongen. För att genomföra användningsfall 5, behöver vi 3 klasser.
En klass är *Season,* andra är *Arbiter* och sen man behöver en *Program* klass som kommer att anropa metod från båda klasser. Klassen Season har två inkapslade fält med två motsvarande egenskaper, och två metoder. En metod är konstruktör och den andra metoden är den metod som används när man tilldelar ny domare i en säsong.