



Linnéuniversitetet

Kalmar Vaxjö

Laborationsanvisning

Gissa det hemliga talet

Steg 2, laborationsuppgift 1



Författare: Mats Looch

Kurs: Inledande programmering med C#

Kurskod: 1DV402

Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen Inledande programmering med C# vid Linnéuniversitetet.

Du får använda detta verk så här:

Allt innehåll i verket Gissa det hemliga talet av Mats Look, förutom Linnéuniversitetets logotyp, symbol och kopparstick, är licensierad under:



Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens.
<http://creativecommons.org/licenses/by-nc-sa/2.5/se/>

Det betyder att du i icke-kommersiella syften får:

- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp, symbol och/eller kopparstick i din nya version!

Vid all användning måste du ange källan: "Linnéuniversitetet – Inledande programmering med C#" och en länk till <https://coursepress.lnu.se/kurs/inledande-programmering-med-csharp> och till Creative Common-licensen här ovan.

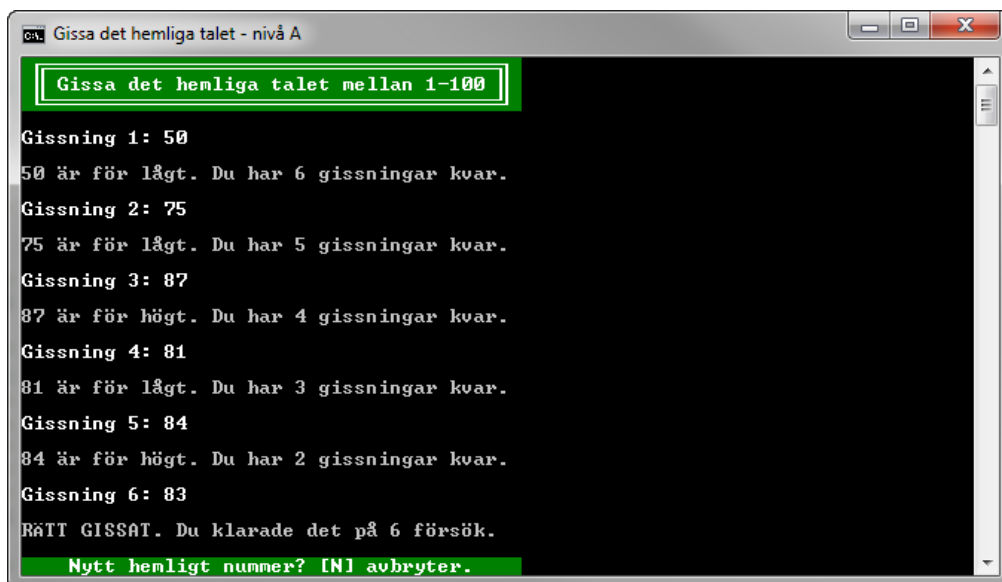
Innehåll

A. Uppgift	5
Problem	5
Det påbörjade projektet	5
Klassen SecretNumber	6
A-krav	8
Läsvärt	9
B. Uppgift	11
Problem	11
Det påbörjade projektet	11
Klassen SecretNumber	12
B-krav	14
Läsvärt	15
C. Uppgift	17
Problem	17
Det påbörjade projektet	17
Den uppräkningsbara typen Outcome	18
Strukturen GuessedNumber	18
Klassen SecretNumber	19
C-krav	20
Läsvärt	21

A. Uppgift

Problem

Skriv färdigt en påbörjad konsolapplikation där användaren ska ha sju försök på sig att gissa ett hemligt tal i det slutna intervallet mellan 1 och 100. Börja med att hämta hem tillhörande projekt och komplettera därefter med klassen `SecretNumber` enligt klassdiagrammet i Figur A.5 så att tester och koden i `Main`-metoden i klassen `Program` fungerar enligt anvisningarna.

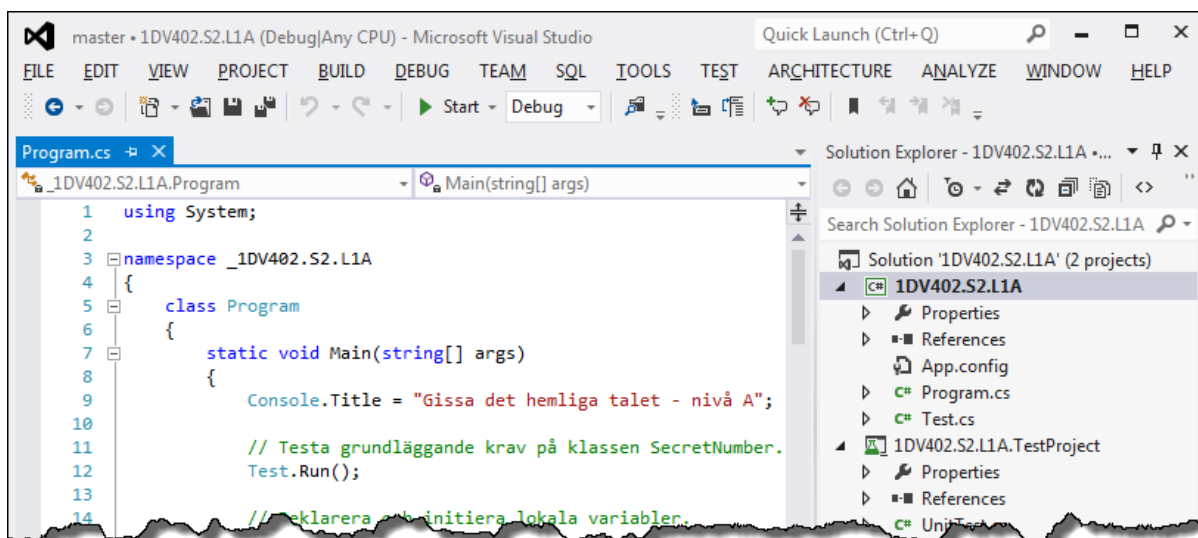


Figur A.1

När en användare gjort en gissning ska resultatet av gissningen presenteras, det vill säga om gissningen var för låg, för hög eller rätt. Har användaren gissat rätt, eller förbrukat samtliga gissningar, ska det inte gå att göra fler gissningar innan ett nytt hemligt tal slumpats.

Det påbörjade projektet

Hämta hem filen `1dv402-s2-11a.zip`, packa upp den och öppna det innehållande projektet i Visual Studio. Projektet innehåller bland annat filerna `Program.cs` och `Test.cs`. Dessa filer innehåller kod som på inget sätt får modifieras. Koden i filerna har till uppgift att testa att klassen `SecretNumber`, som ska implementeras, uppfyller grundläggande krav.

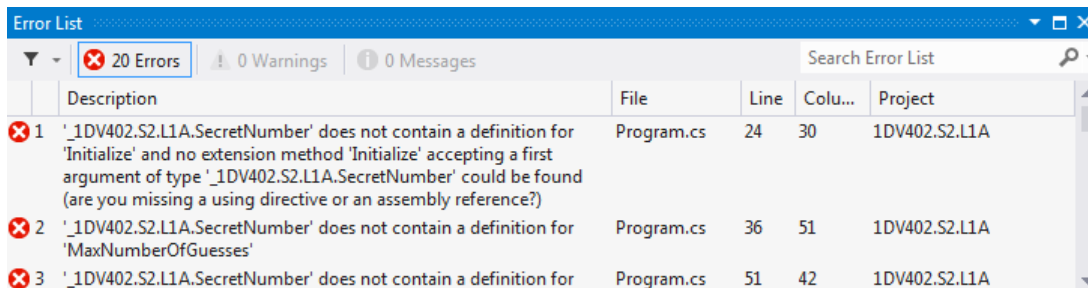


Figur A.2.

Använder du Visual Studio, som kan hantera det bifogade testprojektet, kan du köra de tester som

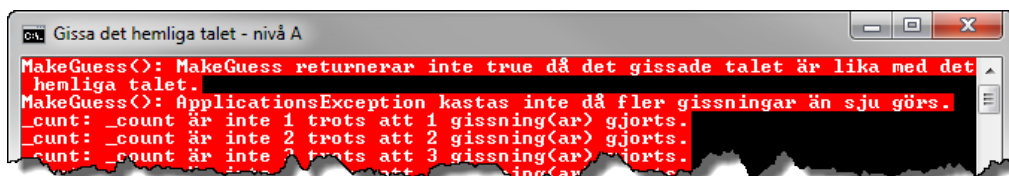
finns i testprojektet 1DV402.S2.L1A.TestProject genom att välja menykommandot **Test ► Run ► All Tests**.

Innan klassen `SecretNumber` lagts till projektet, och delvis implementerats, kommer källkoden inte att kunna kompileras.



Figur A.3. Felmeddelande då klassen `SecretNumber` lagts till men är ofullständigt implementerad.

Då klassen `SecretNumber` är implementerad så långt att den klarar att kompileras anropar metoden `Main()`, i klassen `Program`, den statistiska metoden `Run()` i klassen `Test`. Metoden `Run()` i sin tur anropar ett flertal privata statistiska metoder som testar att grundläggande krav uppfylls av klassen `SecretNumber`. Uppfylls inte alla krav meddelas detta i form av ett eller flera felmeddelande.

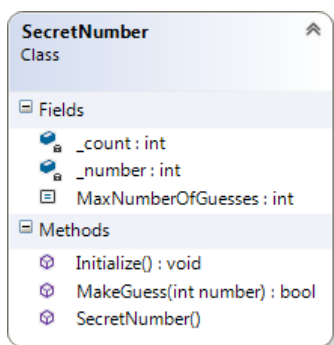


Figur A.4. Felmeddelanden då klassen `SecretNumber` inte är korrekt implementerad.

Först då klassen `SecretNumber` klarar samtliga tester utan fel kan det egentliga programmet starta och användaren kan börja gissa på heltal i det slutna intervallet mellan 1 och 100.

Klassen `SecretNumber`

Klassen måste implementeras så den som minst innehåller medlemmarna enligt klassdiagrammet i Figur A.5 och har den funktionalitet som beskrivs för respektive medlem för att klara samtliga tester.



Figur A.5. Klassdiagram över `SecretNumber`.

Fältet `_count`

Privat fält som räknar antalet gjorda gissningar sedan det hemliga talet slumpades fram.

Fältet `_number`

Privat fält som innehåller det hemliga talet.

Konstanten `MaxNumberOfGuesses`

Publik konstant med värdet 7 som definierar hur många gissningar en användare har på sig att gissa rätt.

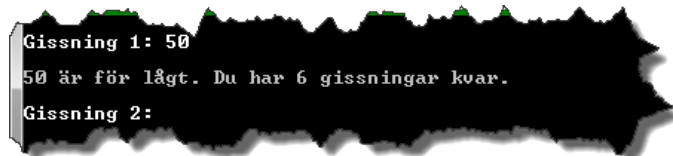
Metoden Initialize

Publik metod som initierar klassens fält.

- `_number` ska tilldelas ett slumpat heltal i det slutna intervallet mellan 1 och 100.
- `_count` ska tilldelas värdet 0.

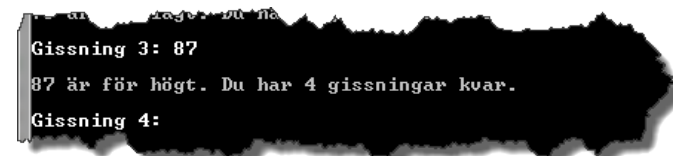
Metoden MakeGuess

Publik metod som anropas för att göra en gissning av det hemliga talet. Beroende om det gissade talets värde, som parametern `number` innehåller, är för högt, lågt eller överensstämmer med det hemliga talet ska lämpliga meddelanden, innehållande det gissade värde samt antalet kvarstående gissningar, skrivas ut.




```
Gissning 1: 50
50 är för lågt. Du har 6 gissningar kvar.
Gissning 2:
```

Figur A.6. Exempel på meddelande efter gissning på ett för lågt värde.



```
Gissning 3: 87
87 är för högt. Du har 4 gissningar kvar.
Gissning 4:
```

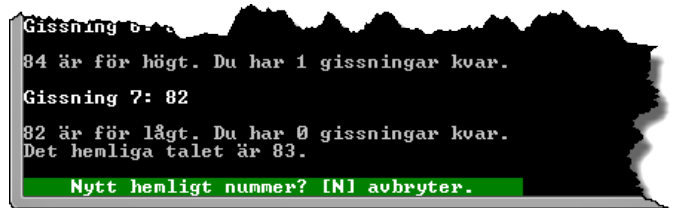
Figur A.7. Exempel på meddelande efter gissning på ett för högt värde.



```
84 är för högt. Du har 2 gissningar kvar.
Gissning 6: 83
RÄTT GISSAT. Du klarade det på 6 försök.
Nytt hemligt nummer? [N] avbryter.
```

Figur A.8. Exempel på meddelande efter gissning på rätt hemligt tal.

Om den sjunde gissningen görs och är felaktig ska användaren meddelas att det inte är några gissningar kvar och vilket det hemliga talet var.¹¹



```
Gissning 7: 82
82 är för lågt. Du har 0 gissningar kvar.
Det hemliga talet är 83.
Nytt hemligt nummer? [N] avbryter.
```

Figur A.9. Användaren misslyckas att gissa rätt hemligt tal på sju försök.

Anropas metoden `MakeGuess()` fler än sju gånger efter varandra innan ett nytt hemligt tal har slumpats fram, genom ett anrop av metoden `Initialize()`, ska metoden `MakeGuess()` kasta ett undantag av typen `ApplicationException`.

Om det vid anrop av metoden `MakeGuess()` skickas med ett argument som inte är i det slutna intervallet mellan 1 och 100 ska metoden, efter att undersökt parameterns värde, kasta ett undantag av typen `ArgumentOutOfRangeException`.

Konstruktorn

Konstruktorn har till uppgift att se till att ett `SecretNumber`-objekt är korrekt initierat. Det innebär att fälten har blivit tilldelade lämpliga värden, vilket enklast görs genom att låta konstruktorn anropa metoden `Initialize()`.

A-krav

1. Koden i `Program.cs` och `Test.cs` måste exekveras och får inte ändras på något sätt.
2. Klassen `SecretNumber` ska vara publik och placerad i en egen fil med namnet `SecretNumber.cs`.
3. Antalet gissningar användaren har på sig att gissa rätt hemligt tal ska vara sju (7) och lagras i en publik namngiven konstant, med namnet `MaxNumberOfGuesses`, i klassen `SecretNumber`.
4. Antalet gissningar som gjorts sedan det hemliga talet slumpats fram ska lagras i det privata fältet `_count` i klassen `SecretNumber`.
5. Det hemliga talet ska lagras i ett privat fält, `_number`, i klassen `SecretNumber`.
6. Konstruktorn i klassen `SecretNumber` måste säkerställa att ett objekt av klassen är korrekt initierat, d.v.s. att fälten har de värden som kan förväntas då ett nytt objekt instansierats och initierats. Speciellt viktigt är att fältet `_number` verkligen har ett värde i det slutna intervallet mellan 1 och 100 för att inte objektets status ska vara ogiltigt efter att konstruktorn exekverats.
7. Metoden `Initialize()` ska göra det möjligt att återställa ett objekt så att fälten `_count` och `_number` har lämpliga värden. Fältet `_count` ska ha värdet 0 och fältet `_number` ska ha ett slumpat värde i det slutna intervallet mellan 1 och 100.
8. Koden som slumpar fram det hemliga talet får bara förekomma en gång. (Bryt inte mot principen DRY, "*Don't Repeat Yourself*".)
9. En gissning ska göras genom att metoden `MakeGuess()` i klassen `SecretNumber` anropas. Metoden ska skriva ut ett lämpligt meddelande i konsolfönstret beroende på resultatet av gissningen. Om användaren gissat rätt hemligt tal ska metoden returnera `true` annars `false`.
 - a) Användaren har sju (7) försök på sig att gissa rätt tal. Det ska inte gå att göra fler gissningar utan att ett nytt hemligt tal slumpas fram. Görs ytterligare försök utöver de stipulerade sju ska ett undantag av typen `ApplicationException` kastas.
 - b) Är det gissade talet inte i det slutna intervallet mellan 1 och 100 ska ett undantag av typen `ArgumentOutOfRangeException` kastas.
 - c) Är det gissade värdet lägre än det hemliga talet ska användaren meddelas att det gissade värdet är för lågt. Även kvarstående antal gissningar som kan göras ska meddelas.
 - d) Är det gissade värdet högre än det hemliga talet ska användaren meddelas att det gissade värdet är för högt. Även kvarstående antal gissningar som kan göras ska meddelas.
 - e) Är det gissade värdet samma som det hemliga talet ska användaren meddelas att det gissade värdet är korrekt. Även antalet gissningar som krävdes ska meddelas.
 - f) Lyckas inte användaren gissa rätt hemligt tal efter sju försök ska det hemliga talet presenteras.

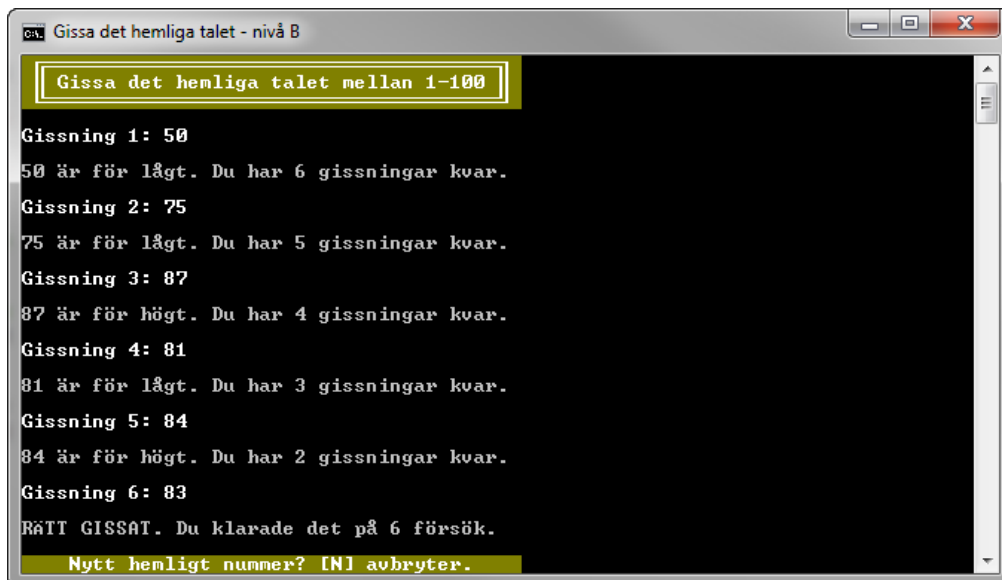
Läsvärt

- Klasser
 - Essential C# 5.0, 209-220.
 - <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
- Åtkomstmodifierare ("*Access Modifiers*")
 - Essential C# 5.0, 227-229.
 - <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.
- Konstruktörer
 - Essential C# 5.0, 244-248, 250-253.
 - <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>
- Konstanter
 - Essential C# 5.0, 267.
 - <http://msdn.microsoft.com/en-us/library/e6w8fe1b.aspx>
- Klassen Random
 - <http://msdn.microsoft.com/en-us/library/ts6se2ek.aspx>

B. Uppgift

Problem

Skriv färdigt en påbörjad konsolapplikation där användaren ska ha sju försök på sig att gissa ett hemligt tal i det slutna intervallet mellan 1 och 100. Börja med att hämta hem tillhörande projekt och komplettera därefter med klassen `SecretNumber` enligt klassdiagrammet i Figur B.5 så att tester och koden i `Main`-metoden i klassen `Program` fungerar enligt anvisningarna.

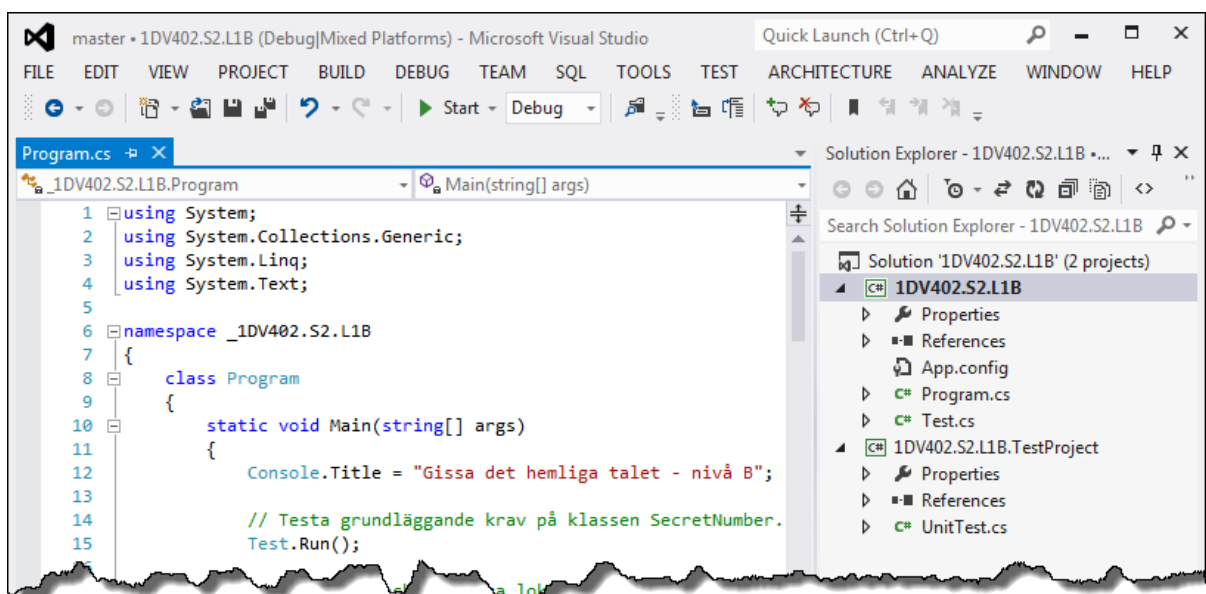


Figur B.1.

När en användare gjort en gissning ska resultatet av gissningen presenteras, det vill säga om gissningen var för låg, för hög eller rätt. Har användaren gissat rätt, eller förbrukat samtliga gissningar, ska det inte gå att göra fler gissningar innan ett nytt hemligt tal slumpats.

Det påbörjade projektet

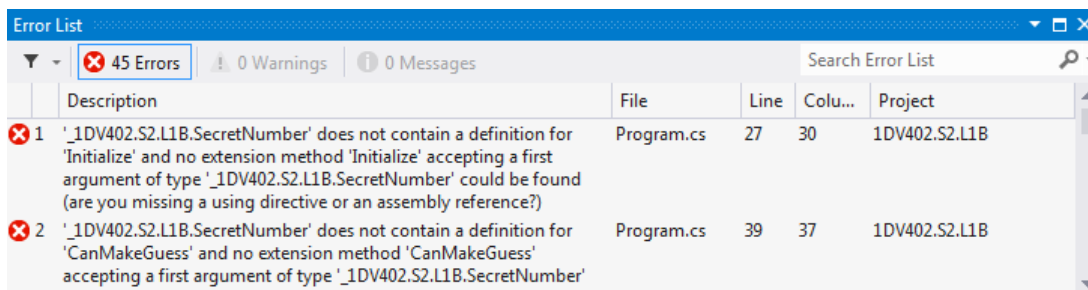
Hämta hem filen `1dv402-s2-11b.zip`, packa upp den och öppna det innehållande projektet i Visual Studio. Projektet innehåller bland annat filerna `Program.cs` och `Test.cs`. Dessa filer innehåller kod som på inget sätt får modifieras. Koden i filerna har till uppgift att testa så den saknade klassen `SecretNumber` uppfyller grundläggande krav.



Figur B.2.

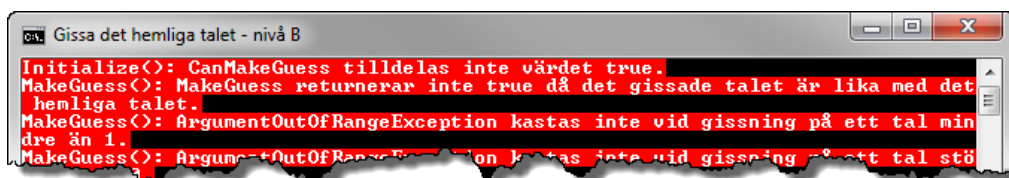
Använder du Visual Studio, som kan hantera det bifogade testprojektet, kan du köra de tester som finns i testprojektet 1DV402.S2.L1B.TestProject genom att välja menykommandot **Test ► Run ► All Tests**.

Innan klassen `SecretNumber` lagts till projektet, och delvis implementerats, kommer källkoden inte att kunna kompileras.



Figur B.3. Felmeddelanden då klassen `SecretNumber` lagts till men är ofullständigt implementerad.

Då klassen `SecretNumber` är implementerad så långt att den klarar att kompileras anropar metoden `Main()`, i klassen `Program`, den statistiska metoden `Run()` i klassen `Test`. Metoden `Run()` i sin tur anropar ett flertal privata statiska metoder som testar att grundläggande krav uppfylls av klassen `SecretNumber`. Uppfylls inte alla krav meddelas detta i form av ett eller flera felmeddelande.

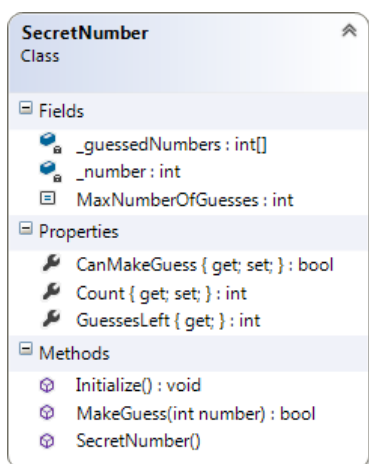


Figur B.4. Felmeddelanden då klassen `SecretNumber` inte är korrekt implementerad.

Först då klassen `SecretNumber` klarar samtliga tester utan fel kan det egentliga programmet starta och användaren kan börja gissa på heltal i det slutna intervallet mellan 1 och 100.

Klassen `SecretNumber`

Klassen måste implementeras så den som minst innehåller medlemmarna enligt klassdiagrammet i Figur A.5 och har den funktionalitet som beskrivs för respektive medlem för att klara samtliga tester.



Figur B.5. Klassdiagram över `SecretNumber`. Observera att set-blocken för egenskaperna `CanMakeGuess` och `Count` är privata.

Fältet `_guessNumbers`

Privat fält, av typen `int[]`, som innehåller gjorda gissningar sedan det hemliga talet slumpats fram.

Fältet `_number`

Privat fält av typen `int` som innehåller det hemliga talet.

Konstanten `MaxNumberOfGuesses`

Publik konstant, av typen `int`, med värdet 7 som definierar hur många gissningar en användare har på sig att gissa rätt.

Egenskapen `CanMakeGuess`

Egenskap, av typen `bool` där `get` är publik och `set` är privat, som håller ordning på om användaren kan gissa eller inte.

Så länge användaren kan göra en gissning ska egenskapen ha värdet `true`. Egenskapen ska ha värdet `false` då en användare förbrukat sju gissningar eller lyckats gissa rätt.

Egenskapen `Count`

Egenskap, av typen `int` där `get` är publik och `set` är privat, som räknar antalet gjorda gissningar sedan det hemliga talet slumpades fram.

Egenskapen `GuessesLeft`

Publik "read-only" egenskap, av typen `int`, som ger hur många gissningar det återstår, d.v.s. differensen mellan maximalt antal tillåtna gissningar och antalet gjorda gissningar.

Metoden `Initialize`

Publik metod som initierar klassens fält och egenskaper.

- Arrayen `_guessedNumbers` refererar till ska återanvändas och måste därför rensas på gjorda gissningar och elementen initieras till standardvärdet för typen `int`.
- `_number` ska tilldelas ett slumpat heltal i det slutna intervallet mellan 1 och 100.
- `CanMakeGuess` ska tilldelas värdet `true`.
- `Count` ska tilldelas värdet 0.

Metoden `MakeGuess`

Publik metod som anropas för att göra en gissning av det hemliga talet. Beroende om det gissade talets värde, som parametern `number` innehåller, är för högt, lågt eller överensstämmer med det hemliga talet ska lämpliga meddelanden, innehållande det gissade värde samt antalet kvarstående gissningar, skrivas ut. En gissning på ett tidigare gissat tal ska inte räknas och användaren ska informeras om att det redan gjorts en gissning på det talet.

```
Gissning 1: 50
50 är för lågt. Du har 6 gissningar kvar.
Gissning 2:
```

Figur B.6. Exempel på meddelande efter gissning på ett för lågt värde.

```
75 är för lågt. Du har 5 gissningar kvar.
Gissning 3: 87
87 är för högt. Du har 4 gissningar kvar.
Gissning 4:
```

Figur B.7. Exempel på meddelande efter gissning på ett för högt värde.

```
Gissning 3: 87
87 är för högt. Du har 4 gissningar kvar.
Gissning 4: 87
Du har redan gissat på 87. Gör om gissningen!
Gissning 4:
```

Figur B.8. Exempel på meddelande efter gissning på ett tal samma som en tidigare gjord gissning.

```
84 är för högt. Du har 2 gissningar kvar.  
Gissning 6: 83  
RÄTT GISSAT. Du klarade det på 6 försök.  
Nytt hemligt nummer? [N] avbryter.
```

Figur B.9. Exempel på meddelande efter gissning på rätt hemligt tal.

Om den sjunde gissningen görs och är felaktig ska användaren meddelas att det inte är några gissningar kvar och vilket det hemliga talet var.

```
84 är för högt. Du har 1 gissningar kvar.  
Gissning 7: 82  
82 är för lågt. Du har 0 gissningar kvar.  
Det hemliga talet är 83.  
Nytt hemligt nummer? [N] avbryter.
```

Figur B.10. Användaren misslyckas att gissa rätt hemligt tal på sju försök.

Anropas metoden `MakeGuess()` fler än sju gånger efter varandra innan ett nytt hemligt tal har slumpats fram, genom ett anrop av metoden `Initialize()`, ska metoden `MakeGuess()` kasta ett undantag av typen `ApplicationException`.

Om det vid anrop av metoden `MakeGuess()` skickas med ett argument som inte är i det slutna intervallet mellan 1 och 100 ska metoden, efter att undersökt parameterns värde, kasta ett undantag av typen `ArgumentOutOfRangeException`.

Konstruktorn

Konstruktorn har till uppgift att se till att ett `SecretNumber`-objekt är korrekt initierat. Det innebär att fält och egenskaper har blivit tilldelade lämpliga värden, vilket enklast görs genom att låta konstruktorn anropa metoden `Initialize()`.

Konstruktorn ska även ansvara för att instansiera arrayen som håller ordning på gjorda gissningar.

B-krav

1. Koden i `Program.cs` och `Test.cs` måste exekveras och får inte ändras på något sätt.
2. Klassen `SecretNumber` ska vara publik och placerad i en egen fil med namnet `SecretNumber.cs`.
3. Antalet gissningar användaren har på sig att gissa rätt hemligt tal ska vara sju (7) och lagras i en publik namngiven konstant, med namnet `MaxNumberOfGuesses`, i klassen `SecretNumber`.
4. Gissningar som gjorts sedan det hemliga talet slumpats fram ska lagras i det privata fältet `_guessedNumbers` i klassen `SecretNumber`. `_guessedNumbers` ska referera till en array med heltal och endast ett objekt av typen `int[]` får instansieras per instans av `SecretNumber`-objekt.
5. Det hemliga talet ska lagras i ett privat fält, `_number`, i klassen `SecretNumber`.
6. Ett `SecretNumber`-objekt ska via egenskapen `CanMakeGuess` hålla ordning på om det är tillåtet/meningsfullt att gissa, d.v.s. anropa metoden `MakeGuess()`, eller inte.
7. Antalet gissningar som gjorts sedan det hemliga talet slumpats fram ska lagras i egenskapen `Count`, där `get`-metoden ska vara publik och `set`-metoden privat, i klassen `SecretNumber`.
8. Egenskapen `GuessesLeft` i klassen `SecretNumber` ska ge kvarstående antal gissningar.
9. Konstruktorn i klassen `SecretNumber` måste säkerställa att ett objekt av klassen är korrekt initierat, d.v.s. att fälten och egenskaperna har de värden som kan förväntas då ett nytt objekt instansierats och initierats. Speciellt viktigt är att fältet `_number` verkligen har ett

värde i det slutna intervallet mellan 1 och 100 för att inte objektets status ska vara ogiltigt efter att konstruktorn exekverats.

10. Metoden `Initialize()` ska göra det möjligt att återställa ett objekt så att fält och egenskaper har lämpliga värden så att en ny gissningsomgång kan göras utan ett nytt `SecretNumber`-objekt behöver instansieras.
11. Koden som slumpar fram det hemliga talet får bara förekomma en gång. (Bryt inte mot principen DRY, "Don't Repeat Yourself".)
12. En gissning ska göras genom att metoden `MakeGuess()` i klassen `SecretNumber` anropas. Metoden ska skriva ut ett lämpligt meddelande i konsolfönstret beroende på resultatet av gissningen. Om användaren gissat rätt hemligt tal ska metoden returnera `true` annars `false`.
 - a) Användaren har sju (7) försök på sig att gissa rätt tal. Det ska inte gå att göra fler gissningar utan att ett nytt hemligt tal slumpas fram. Görs ytterligare försök utöver de stipulerade sju ska ett undantag av typen `ApplicationException` kastas.
 - b) Är det gissade talet inte i det slutna intervallet mellan 1 och 100 ska ett undantag av typen `ArgumentOutOfRangeException` kastas.
 - c) Är det gissade värdet samma som en tidigare gissnings värde ska inte gissningen räknas och användaren informeras om att han/hon redan gissat på värdet.
 - d) Är det gissade värdet lägre än det hemliga talet ska användaren meddelas att det gissade värdet är för lågt. Även kvarstående antal gissningar som kan göras ska meddelas.
 - e) Är det gissade värdet högre än det hemliga talet ska användaren meddelas att det gissade värdet är för högt. Även kvarstående antal gissningar som kan göras ska meddelas.
 - f) Är det gissade värdet samma som det hemliga talet ska användaren meddelas att det gissade värdet är korrekt. Även antalet gissningar som krävdes ska meddelas.
 - g) Lyckas inte användaren gissa rätt hemligt tal efter sju försök ska det hemliga talet presenteras.

Läsvärt

- Klasser
 - Essential C# 5.0, 209-220.
 - <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
- Åtkomstmodifierare ("Access Modifiers")
 - Essential C# 5.0, 227-229.
 - <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.
- Konstruktörer
 - Essential C# 5.0, 244-248, 250-253.
 - <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>
- Egenskaper
 - Essential C# 5.0, 229-235.
 - <http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx>.

- Konstanter
 - Essential C# 5.0, 267.
 - <http://msdn.microsoft.com/en-us/library/e6w8fe1b.aspx>
- Klassen Array
 - <http://msdn.microsoft.com/en-us/library/czz5hkty.aspx>.
- Klassen Random
 - <http://msdn.microsoft.com/en-us/library/ts6se2ek.aspx>

C.Uppgift

Problem

Skriv färdigt en påbörjad konsolapplikation där användaren ska ha sju försök på sig att gissa ett hemligt tal i det slutna intervallet mellan 1 och 100. Börja med att hämta hem tillhörande projekt och komplettera därefter med klassen `SecretNumber` enligt Figur C.5 - Figur C.7 så att tester och koden i `Main`-metoden i klassen `Program` fungerar enligt anvisningarna.

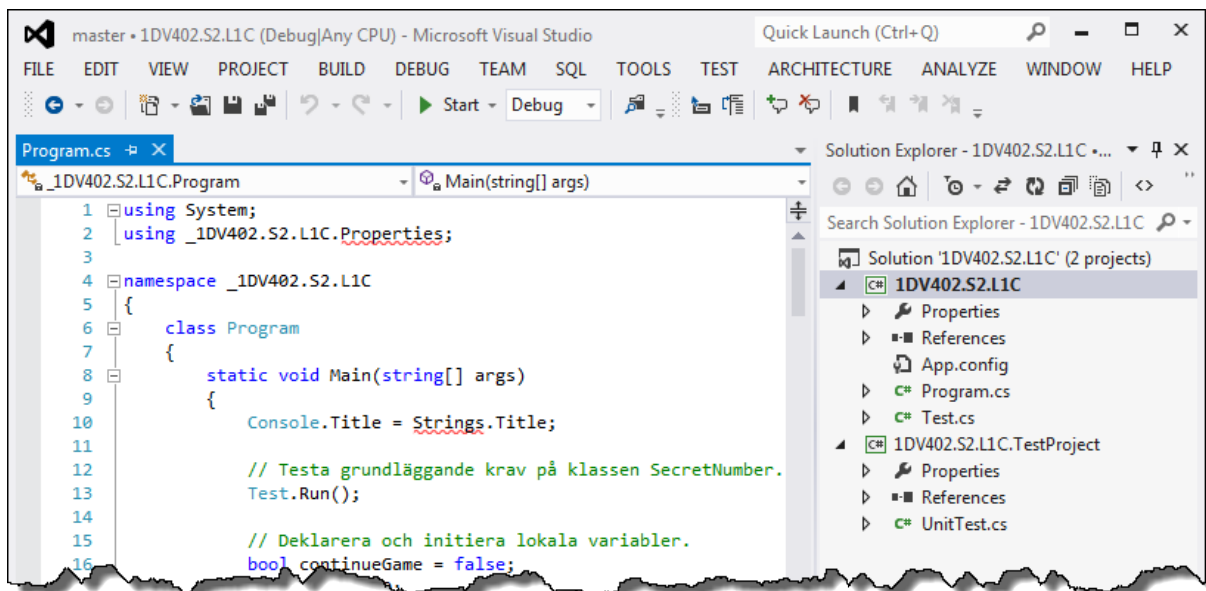


Figur C.1.

När en användare gjort en gissning ska resultatet av gissningen presenteras, det vill säga om gissningen var för låg, för hög eller rätt. Har användaren gissat rätt, eller förbrukat samtliga gissningar, ska det inte gå att göra fler gissningar innan ett nytt hemligt tal slumpats.

Det påbörjade projektet

Hämta hem filen `1dv402-s2-11c.zip`, packa upp den och öppna det innehållande projektet i Visual Studio.

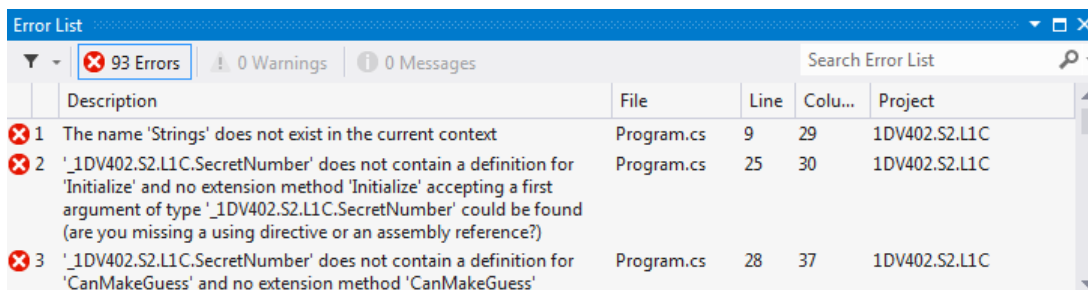


Figur C.2.

Projektet innehåller bland annat filerna `Program.cs` och `Test.cs`. Dessa filer innehåller kod som på inget sätt får modifieras. Koden i filerna har till uppgift att testa så den saknade klassen `SecretNumber` uppfyller grundläggande krav.

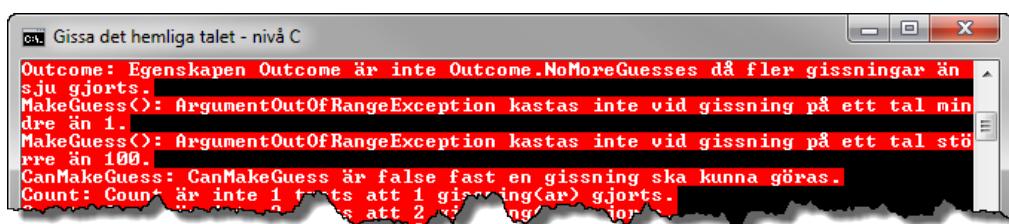
Använder du Visual Studio, som kan hantera det bifogade testprojektet, kan du köra de tester som finns i testprojektet `1DV402.S2.L1C.TestProject` genom att välja menykommandot **Test ► Run ► All Tests**.

Innan klassen `SecretNumber` lagts till projektet, och delvis implementerats, kommer källkoden i projekten inte att kunna kompileras.



Figur C.3. Felmeddelande då klassen SecretNumber lagts till men är ofullständigt implementerad.

Då klassen SecretNumber är implementerad så långt att den klarar att kompileras anropar metoden Main(), i klassen Program, den statistiska metoden Run() i klassen Test. Metoden Run() i sin tur anropar ett flertal privata statiska metoder som testar att grundläggande krav uppfylls av klassen SecretNumber. Uppfylls inte alla krav meddelas detta i form av ett eller flera felmeddelande.

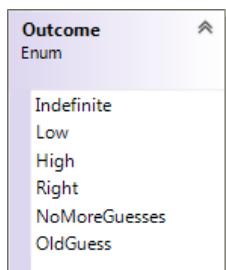


Figur C.4. Felmeddelanden då klassen SercetNumber inte är korrekt implementerad.

Först då klassen SecretNumber klarar samtliga tester utan fel kan det egentliga programmet starta och användaren kan börja gissa på heltal i det slutna intervallet mellan 1 och 100.

Den uppräkningsbara typen Outcome

Typen Outcome används för att representera utfallet av en gissning.

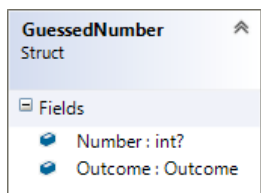


Figur C.5

- Innan en gissning gjort är utfallet obestämt (Indefinite).
- En gissning på ett tal mindre än det hemliga talet är för låg (Low).
- En gissning på ett tal större än det hemliga talet är för hög (High).
- En gissning på ett tal lika med det hemliga talet är för rätt (Right).
- Har sju gissningar redan gjorts och fler försök att gissa görs saknas fler möjligheter att gissa (NoMoreGuesses).
- En gissning på ett tal överstämmande med en tidigare gissning är en gammal gissning (OldGuess).

Strukturen GuessedNumber

Instanser av strukturen GuessedNumber används för att lagra information om genomförda gissningar.



Figur C.6

Fältet Number

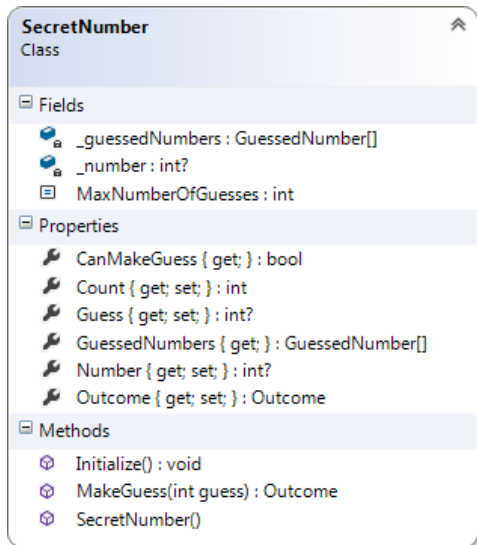
Publikt fält av typen int? som innehåller en gissnings värde.

Fältet Outcome

Publikt fält av typen Outcome som innehåller utfallet av en gissning.

Klassen SecretNumber

Klassen måste implementeras så den som minst innehåller medlemmarna enligt klassdiagrammet i Figur C.7 och har den funktionalitet som beskrivs för respektive medlem för att klara samtliga tester.



Figur C.7. Klassdiagram över SecretNumber. Observera att set-blocken för egenskaperna Count, Guess, Number och Outcome ska vara privata.

Fältet `_guessedNumbers`

Privat fält, av typen `GuessedNumber[]`, som innehåller gjorda gissningar med tillhörande utfall sedan det hemliga talet slumpats fram. Kapslas in av egenskapen `GuessedNumbers`.

Fältet `_number`

Privat fält av typen `int?` som innehåller det hemliga talet. Kapslas in av egenskapen `Number`.

Konstanten `MaxNumberOfGuesses`

Publik konstant, av typen `int`, med värdet 7 som definierar hur många gissningar en användare har på sig att gissa rätt.

Egenskapen `CanMakeGuess`

Egenskap, av typen `bool` där `get` är publik och `set` är privat, som håller ordning på om användaren kan gissa eller inte.

Så länge användaren kan göra en gissning ska egenskapen ha värdet `true`. Egenskapen ska ha värdet `false` då en användare förbrukat sju gissningar eller lyckats gissa rätt.

Egenskapen `Count`

Egenskap, av typen `int`, där `get` är publik och `set` är privat, som räknar antalet gjorda gissningar sedan det hemliga talet slumpades fram.

Egenskapen `Guess`

Egenskap, av typen `int?`, där `get` är publik och `set` är privat, som innehåller den senaste gissningens värde. Innan någon gissning gjorts ska egenskapen ha värdet `null`.

Egenskapen `GuessedNumbers`

”Read-only”-egenskap, av typen `GuessedNumber[]` som ger en referens till en kopia av det privata fältet `_guessedNumbers`. OBS! För att undvika en ”privacy leak” får under inga omständigheter en referens till det privata fältet `_guessedNumbers` returneras av egenskapen.

Egenskapen `Number`

Egenskap, av typen `int?`, där `get` är publik och `set` är privat, som innehåller det hemliga talet. Så länge som det går att gissa ska egenskapen ge värdet `null`. Går det inte att gissa ska egenskapen ge värdet den privata egenskapen `_number` har.

Egenskapen Outcome

Egenskap, av typen `Outcome`, där `get` är publik och `set` är privat, som innehåller utfallet av den senaste gissningen. Innan någon gissning gjorts ska egenskapen ha standardvärdet `Outcome.Indefinite`.

Metoden Initialize

Publik metod som initierar klassens fält och egenskaper.

- Arrayen `_guessedNumbers` refererar till ska återanvändas och måste därför rensas på gjorda gissningar och elementen initieras till standardvärdet för typen `GuessedNumber`.
- `Number` ska tilldelas ett slumpat heltal i det slutna intervallet mellan 1 och 100.
- `Count` ska tilldelas värdet 0.
- `Guess` ska tilldelas värdet `null`.
- `Outcome` ska tilldelas värdet `Outcome.Indefinite`.

Metoden MakeGuess

Publik metod som anropas för att göra en gissning av det hemliga talet. Beroende om det gissade talets värde, som parametern `number` innehåller, är för högt, lågt eller överensstämmer med det hemliga talet ska lämpligt värde av typen `Outcome` returneras. En gissning på ett tidigare gissat tal ska inte räknas.

Om det vid anrop av metoden `MakeGuess()` skickas med ett argument som inte är i det slutna intervallet mellan 1 och 100 ska metoden, efter att undersökt parametrarnas värde, kasta ett undantag av typen `ArgumentOutOfRangeException`.

Konstruktorn

Konstruktorn har till uppgift att se till att ett `SecretNumber`-objekt är korrekt initierat. Det innebär att fält och egenskaper har blivit tilldelade lämpliga värden, vilket enklast görs genom att låta konstruktorn anropa metoden `Initialize()`.

Konstruktorn ska även ansvara för att instansiera arrayen som håller ordning på gjorda gissningar.

C-krav

1. Koden i `Program.cs` och `Test.cs` måste exekveras och får inte ändras på något sätt.
2. Klassen `SecretNumber` ska vara publik placerad i en egen fil med namnet `SecretNumber.cs`. Den uppräkningsbara typen ska även den vara publik och placerad i denna fil men utanför klassdefinitionen.
3. Strukturen `GuessedNumber` ska vara publik och placerad i en egen fil med namnet `GuessedNumber.cs`.
4. Antalet gissningar användaren har på sig att gissa rätt hemligt tal ska vara sju (7) och lagras i en publik namngiven konstant, med namnet `MaxNumberOfGuesses`, i klassen `SecretNumber`.
5. Gissningar som gjorts sedan det hemliga talet slumpats fram ska lagras i det privata fältet `_guessedNumbers` i klassen `SecretNumber`. `_guessedNumbers` ska referera till en array med element av typen `GuessedNumber` och endast ett objekt av typen `GuessedNumber[]` får instansieras per instans av `SecretNumber`-objekt.
6. Det hemliga talet ska lagras i ett privat fält, `_number`, i klassen `SecretNumber`.
7. Egenskapen `CanMakeGuess` ska ge om det är meningsfullt att gissa, d.v.s. anropa metoden `MakeGuess()`, eller inte, d.v.s. om det finns några gissningar kvar och om rätt tal ännu inte gissats.
8. Antalet gissningar som gjorts sedan det hemliga talet slumpats fram ska lagras i egenskapen `Count`, där `get`-metoden ska vara publik och `set`-metoden privat.
9. Konstruktorn i klassen `SecretNumber` måste säkerställa att ett objekt av klassen är korrekt

initierat, d.v.s. att fälten och egenskaperna har de värden som kan förväntas då ett nytt objekt instansierats och initierats. Speciellt viktigt är att fältet `_number` verkligen har ett värde i det slutna intervallet mellan 1 och 100 för att inte objektets status ska vara ogiltigt efter att konstruktorn exekverats.

10. Metoden `Initialize()` ska göra det möjligt att återställa ett objekt så att fält och egenskaper har lämpliga värden så att en ny gissningsomgång kan göras utan ett nytt `SecretNumber`-objekt behöver instansieras.
11. Koden som slumpar fram det hemliga talet får bara förekomma en gång. (Bryt inte mot principen DRY, "*Don't Repeat Yourself*".)
12. En gissning ska göras genom att metoden `MakeGuess()` i klassen `SecretNumber` anropas. Metoden ska returnera lämpligt värde beroende på resultatet av gissningen.
 - a) Är det gissade talet inte i det slutna intervallet mellan 1 och 100 ska ett undantag av typen `ArgumentOutOfRangeException` kastas.
 - b) Användaren har sju (7) försök på sig att gissa rätt tal. Det ska inte gå att göra fler gissningar utan att ett nytt hemligt tal slumpas fram. Görs ytterligare försök utöver de stipulerade sju ska värdet `Outcome.NoMoreGuesses` returneras.
 - c) Är det gissade värdet samma som en tidigare gissnings värde ska inte gissningen räknas och värdet `Outcome.OldGuess` returneras.
 - d) Är det gissade värdet lägre än det hemliga talet ska värdet `Outcome.Low` returneras.
 - e) Är det gissade värdet högre än det hemliga talet ska värdet `Outcome.High` returneras.
 - f) Är det gissade värdet samma som det hemliga talet ska och värdet `Outcome.Right` returneras.
13. Går att genomföra en gissning ska det hemliga talet inte kunna kommas åt via egenskapen `Number`.
14. Går det inte att genomföra en gissning ska det hemliga talet finnas tillgängligt via egenskapen `Number`.
15. Egenskapen `GuessedNumbers` får inte returnera en referens till det privata fältet `_guessedNumbers`.
16. `set`-metoderna för egenskaperna `Count`, `Guess`, `Number` och `Outcome` måste vara privata.

Läsvärt

- Klasser
 - Essential C# 5.0, 209-221.
 - <http://msdn.microsoft.com/en-us/library/0b0thckt.aspx>.
- Åtkomstmodifierare ("*Access Modifiers*")
 - Essential C# 5.0, 227-229.
 - <http://msdn.microsoft.com/en-us/library/ms173121.aspx>.
- Konstruktörer
 - Essential C# 5.0, 244-248, 252-253.
 - <http://msdn.microsoft.com/en-us/library/k6sa6h87.aspx>

- Egenskaper
 - Essential C# 5.0, 229-244.
 - <http://msdn.microsoft.com/en-us/library/x9fsa0sw.aspx>.
- Konstanter
 - Essential C# 5.0, 267.
 - <http://msdn.microsoft.com/en-us/library/e6w8fe1b.aspx>
- Klassen Array
 - <http://msdn.microsoft.com/en-us/library/czz5hkty.aspx>.
- Klassen Random
 - <http://msdn.microsoft.com/en-us/library/ts6se2ek.aspx>