Linnéuniversitetet
Kalmar Växjö

Övningsuppgift

# Statistik över heltal

Steg 2

*Författare:* Mats Loock

*Kurs:* Inledande programmering med C#

*Kurskod:*1DV402

Linnéuniversitetet
Kalmar Växjö

# 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:



## 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.

Linnéuniversitetet
Kalmar Växjö

# Innehåll

Uppgift _____ 5
Problem _____ 5
Mål _____ 7
Tips _____ 7
Lösning _____ 8

# Uppgift

## Problem

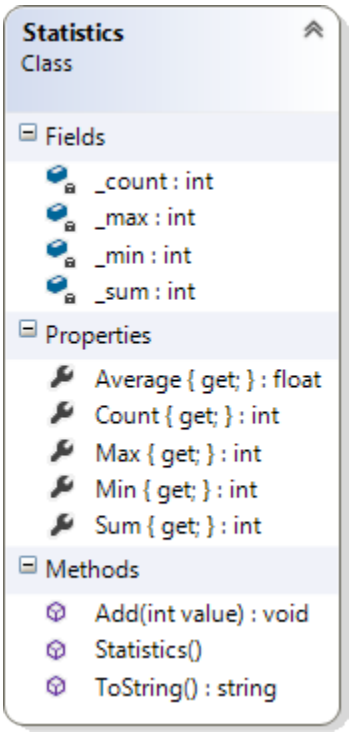
Ett program ber användaren mata in två serier om tio heltal vardera. Programmet ska presentera det största och det minsta av de inmatade heltalen för respektive serie. De olika seriernas summor och medelvärden ska även presenteras. Vidare ska det bestämmas vilken av serierna som har högst medelvärde.

Då de två serierna ska behandlas på exakt samma sätt är det bäst att skapa en klass, som du lämpligen kallar `Statistics`, vars uppgift blir att hålla ordning på datat.

Klassen ska inte lagar varje enskilt heltal, utan bara den information som krävs för att kunna bestämma ett medelvärde, samt högsta och lägsta heltal. För att bestämma medelvärdet krävs att information om summan av heltalen och antalet heltal som bildat summan.

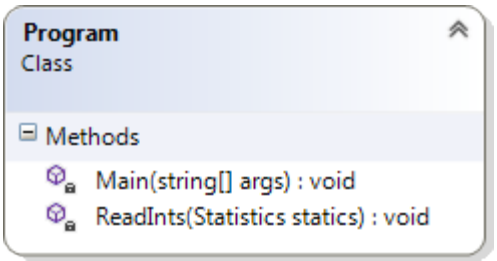
För att enkelt kunna presentera ett objekts status i form av en sträng ska du överskugga metoden `ToString` (som ju deklareras av basklassen `Object`).

Studera klassdiagram och koden som testkör klassen så du kan implementera klassen `Statistics` så att koden i klassen `Program` kan kompileras och exekveras utan att du gör några ändringar.




Figur 1. Klassdiagram över klassen Statistics.

Vad respektive fält, egenskap och metod i klassen `Statistics` ska göra torde vara uppenbart. Poängteras bör att du inte får använda arrayer för att lösa uppgiften.




Figur 2. Klassdiagram över klassen Program.

Program.cs

StatisticsOnIntegers.Program

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace StatisticsOnIntegers
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Statistics statics1 = new Statistics();
14             Statistics statics2 = new Statistics();
15
16             Console.WriteLine("Ange tio heltal till den första serien: ");
17             ReadInts(statics1);
18             Console.WriteLine(statics1);
19
20             Console.WriteLine("Ange tio heltal till den andra serien: ");
21             ReadInts(statics2);
22             Console.WriteLine(statics2);
23
24             Console.WriteLine("Den {0} serien har högst medelvärde.",
25                 statics1.Average > statics2.Average ? "första" : "andra");
26         }
27
28         private static void ReadInts(Statistics statics)
29         {
30             int value;
31             for (int i = 0; i < 10; ++i)
32             {
33                 while (!int.TryParse(Console.ReadLine(), out value))
34                 {
35                     Console.WriteLine("Du måste ange ett heltal!");
36                 }
37                 statics.Add(value);
38             }
39         }
40     }
41 }
```

**Figur 3. Implementation av klassen Program vars kod testar klassen Statistics.**

Istället för att dubblera koden som läser in tio heltal används metoden `ReadInts`. Metoden anropas två gånger – en gång för den första serien och en gång för den andra. Varje inläst heltal läggs till det `Statistics`-objekt som referensvariabeln `statistics` refererar till med hjälp av metoden `Add`. Metoden `TryParse` använder en `out`-parameter. Vad det är och hur den fungerar hittar du i kurslitteraturen, kapitel 4, under underrubriken *”Output Parameters (out)”*.

I samband med den avslutande utskriften i metoden Main används ”if”-operatorn (*conditional operator*). Du kan läsa om den i kurslitteraturen, kapitel 3, under underrubriken *”Conditional Operator (?)”*.

## Mål

Efter att ha gjort uppgiften ska du:

- Inse att en klass som inte explicit (uttryckligen) ärver från en annan klass automatiskt ärver från klassen `Object`, och därför bl.a. får tillgång till metoden `ToString`.
- Kunna implementera en klass som innehåller fält, konstruktorer, egenskaper och över skuggade metoder.

## Tips

Läs om:

- Klasser i kurslitteraturen, kapitel 5.
- Roten till alla klasser i kurslitteraturen, kapitel 6, under rubriken ”*All Classes Derive fromSystem.Object*”.
- Överskuggning hittar du i kurslitteraturen, kapitel 6, under rubriken ”*Overriding the Base Class*”. Läs mer om överskuggning i artikeln ”*Versioning with the Override and New Keywords (C# Programming Guide)*” http://msdn.microsoft.com/en-us/library/6fawty39.aspx som du hittar i online-hjälpen.

# Lösning

Statistics.cs

StatisticsOnIntegers.Statistics

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace StatisticsOnIntegers
8  {
9      class Statistics
10     {
11         private int _count;
12         private int _sum;
13         private int _max;
14         private int _min;
15
16         public Statistics()
17         {
18             _count = 0;
19             _sum = 0;
20             _max = int.MinValue;
21             _min = int.MaxValue;
22         }
23
24         public float Average
25         {
26             get
27             {
28                 return (float)_sum / _count;
29             }
30         }
31
32         public int Count
33         {
34             get { return _count; }
35         }
36
37         public int Max
38         {
39             get { return _max; }
40         }
41
42         public int Min
43         {
44             get { return _min; }
45         }
46
47         public int Sum
48         {
49             get { return _sum; }
50         }
```

*Lösningen forsätter på nästa sida!*

*Fortsättning på lösningen från föregående sida!*

```
51
52          public void Add(int value)
53          {
54              ++_count;
55              _sum += value;
56
57              if (value > _max)
58              {
59                  _max = value;
60              }
61
62              if (value < _min)
63              {
64                  _min = value;
65              }
66          }
67
68          public override string ToString()
69          {
70              StringBuilder sb = new StringBuilder();
71              sb.AppendFormat("Summan av talen är: {0}\n", _sum);
72              sb.AppendFormat("Det största talet är: {0}\n", _max);
73              sb.AppendFormat("Det lägsta talet är: {0}\n", _min);
74              sb.AppendFormat("Medelvärdet är: {0}", Average);
75
76              return sb.ToString();
77          }
78      }
79  }
```

**Figur 4. Implementationen av klassen Statistics.**

Konstruktorn initierar objektets fält till lämpliga startvärden. `int.MinValue` och `int.MaxValue` används för att logiken i metoden `Add` ska bli enklare då högsta och minsta värdet ska bestämmas.

Då egenskapen `Average` returnerar medelvärdet måste den en av operanderana typas om till ett flyttal av typen float. Varför? Därför annars sker en heltalsdivision och eventuella decimaler skulle i så fall tunkeras (klippas bort).

I metoden `Add` händer ”allt”. Här räknas antalet heltal upp, summan uppdateras och det undersöks om det nya heltalet är ett nytt högsta eller minsta heltal.

Metoden `ToString` skapar en sträng med den information som ska presneteras om objektet. Använder här klassen `StringBuilder` som är mycket effektivare att använda (ger kod som är snabbare att exekvera) än att konkatinera (slå samman) strängar. I kurslitteraturen på sidorna 319-321 hittar du information om klassen `StringBuilder`. Det som returneras av metoden är hur som helst en sträng som beskriver objektet som anropade den. När anropas den då? Satsen `Console.WriteLine(statics1)` ser faktiskt till att metoden `ToString` anropas. Det sker implicit (automatiskt). För tydlighets skull kan man så klart skriva `Console.WriteLine(statics1.ToString())` men det behövs ju faktiskt inte.