



Linnéuniversitetet

Kalmar Växjö

Guide

Frekvenstabell över tärningskast med C#

Introduktionsuppgift



Författare: Mats Looek

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 Frekvenstabell över tärningskast med C# av Mats Loock, 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!

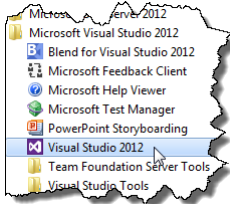
Vid all användning måste du ange källan: ”Linnéuniversitetet – Inledande programmering med C#” och en länk till <https://coursepress.lnu.se/kurs/inledande-programmering-med-csharp> och till Creative Common-licensen här ovan.



Innehåll

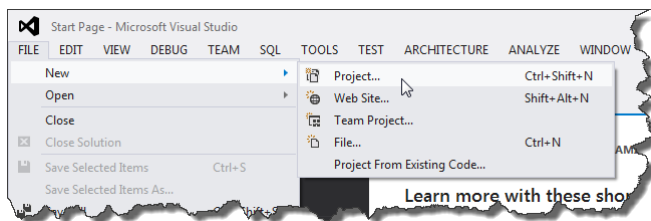
Du ska följa ”steg för steg”-instruktionen i denna introduktionsuppgift och skapa en konsolapplikation med C# och Visual Studio 2012. Applikationen ska slumpa i det slutna intervallet mellan 100 till 1000 tärningskast och presentera en frekvenstabell över förekomsten av ettor, tvåor, osv.

1. Starta Visual Studio 2012.



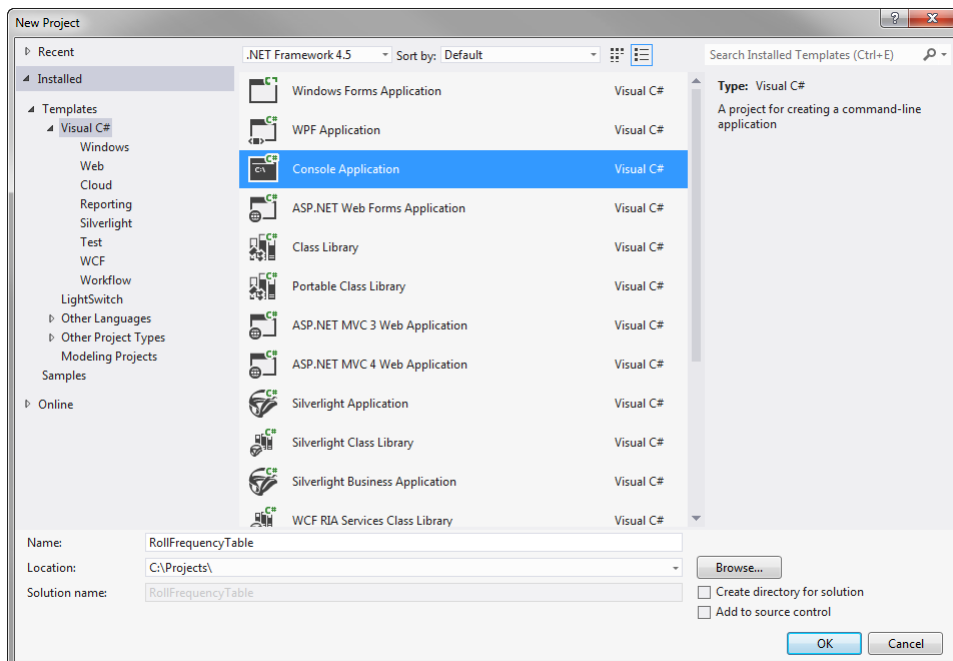
Figur 1.

2. Du ska skapa ett projekt för en konsolapplikation med C#, välj därför **File ► New ► Project....**



Figur 2.

3. Dialogrutan **New Project** visas.



Figur 3.

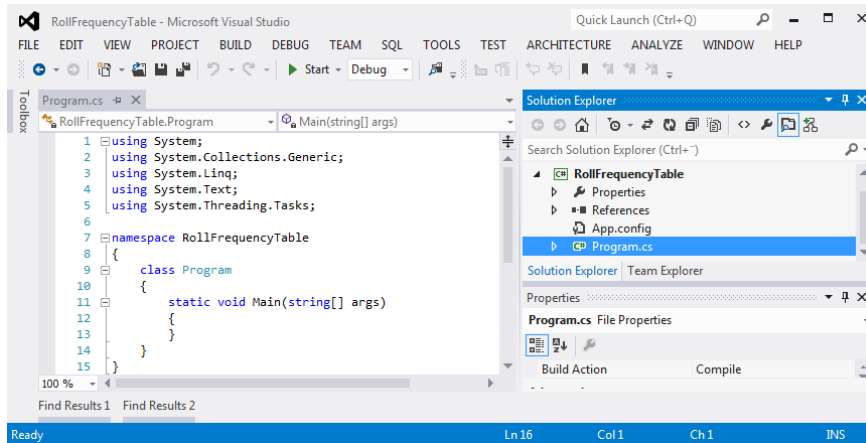
- a) Markera **Visual C#** under **Installed, Templates**.
- b) Kontrollera så att **.NET Framework 4.5** visas i den nedrullningsbara listrutan.
- c) Markera **Console Application**.
- d) Vid **Name** skriver du in projektets namn (RollFrequencyTable).
- e) Ange vid **Location** en lämplig katalog där projektet ska sparas (t.ex. C:\Projects).



- Visual Studio skapar ett nytt konsolprojekt som placeras i en egen "solution" med samma namn som projektet. Det nya projektet innehåller bland annat filen Program.cs.

Filer är organiserade i projekt och ett projekt kan innehålla flera filer som tillsammans utgör applikationen.

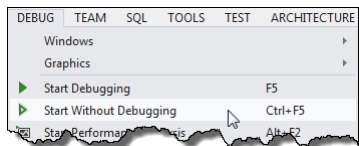
Projekt är organiserade i "solutions". En "solution" kan innehålla flera projekt. Fönstret **Solution Explorer** innehåller projekt och filer.



Figur 4.

- Den filen Program.cs innehåller klassen Program med metoden Main och därmed är minimikraven uppfyllda för att du nu ska kunna köra applikationen.

För att köra applikationen välj menykommandot **Debug ► Start Without Debugging**, eller tryck ner tangentbordskombinationen **Ctrl + F5**.

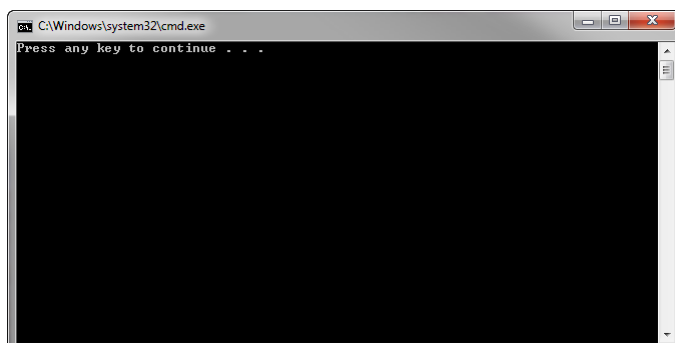


Figur 5.

Vad händer med källkoden i Program.cs?

Källkoden i Program.cs kompileras till IL-kod ("Intermediate Language") som placeras i en ny fil, Program.exe, en "assembly". Vid exekvering av applikationen ansvarar en virtuell maskin kallad CLR ("Common Language Runtime"), en central del av dotnetramverket (.NET Framework), för att IL-koden i "assemblyn" "just-in-time"-kompileras till maskinkod som sedan exekveras.

- Applikationen körs och ett tomt konsolfönster, förutom texten *Press any key to continue* som Visual Studio lagt dit, visas. Stäng konsolfönstret genom att trycka på vilken tangent som helst på tangentbordet.



Figur 6.

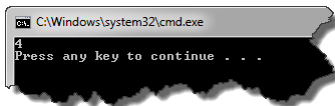


7. Applikationen ska simulera tärningskast. Ett tal mellan 1 och 6 kan representera ett tärningskast. Resultatet av ett tärningskast måste sparas i en variabel vars värde sedan presenteras i konsolfönstret.
 - a) Börja med att radera satsen på rad 5 `using System.Threading.Tasks;` för att radnummer i anvisningar och bilder ska stämma!
 - b) Deklarera en variabel, med namnet `roll`, av typen `int` och initiera den till värdet 4 (satsen på rad 12 i Figur 7).
 - c) Presentera variabelns värde (satsen på rad 13 i Figur 7).

```
1 using System;
2     using System.Collections.Generic;
3     using System.Linq;
4     using System.Text;
5
6 namespace RollFrequencyTable
7 {
8     class Program
9     {
10         static void Main(string[] args)
11         {
12             int roll = 4;
13             Console.WriteLine(roll);
14         }
15     }
16 }
```

Figur 7. Deklaration och initiering av variabeln `roll` vars värde skrivs ut.

8. Testa applikationen genom att välj menykommandot **Debug ► Start Without Debugging**, eller tryck ner tangentbordskombinationen `Ctrl + F5`. (Gjorda förändringar sparas automatiskt i samband med att ett projekt exekveras).



Figur 8. Värdet 4 skrivs ut eftersom variabeln `roll` har det värdet.

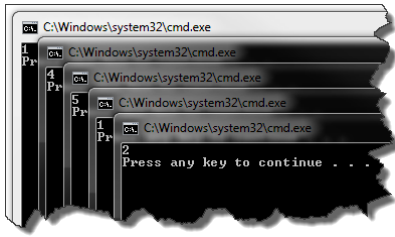
9. Resultatet av ett tärningskast slumpas inte på något vis av applikationen utan är vad som kallas "hårkodat" till värdet 4. För att slumpa ett värde i det slutna intervallet mellan 1 och 6 måste ett objekt av typen `Random` användas.
 - a) Istället för att tilldela variabeln `roll` värdet 4 ska variabeln tilldelas värdet 0 (satsen på rad 12 i Figur 9).
 - b) Skapa och initiera en referensvariabel, med namnet `die` (som betyder tärning på engelska), av typen `Random` att referera till ett objekt av typen `Random` (satsen på rad 13 i Figur 9).
 - c) Tilldela `roll` värdet som metoden `Next()` returnerar (satsen på rad 15 i Figur 9).

```
10 static void Main(string[] args)
11 {
12     int roll = 0;
13     Random die = new Random();
14
15     roll = die.Next(1, 7);
16
17     Console.WriteLine(roll);
18 }
```

Figur 9. Metodanropet `die.Next(1,7)` returnerar ett tal i det slutna intervallet mellan 1 och 6.



10. Testa applikationen flera gånger och konstatera värden mellan 1 och 6 skrivs ut i konsolfönstret.



Figur 10.

11. Applikationen kan nu simulera ett tärningskast. Kravet är att mellan 100 och 1000 tärningskast ska kunna simuleras. Hur ska applikationen göra för att simulera t.ex. 273 tärningskast?

Det kan enklast göras genom att låta en "for"-sats omsluta raderna 15 och 17 i Figur 9. Satserna på raderna 17 och 19 i Figur 11 kommer att upprepas 273 gånger.

```
10 static void Main(string[] args)
11 {
12     int roll = 0;
13     Random die = new Random();
14
15     for (int i = 0; i < 273; i++)
16     {
17         roll = die.Next(1, 7);
18
19         Console.WriteLine(roll);
20     }
21 }
```

Figur 11. En "for"-sats som används för att upprepa de omslutna satserna 273 gånger.


12. Antalet tärningskast är nu "hårdkodat" till 273. Användaren av applikationen ska kunna ange ett heltal i det slutna intervallet mellan 100 och 1000.

- Deklarera och initiera variabeln count till 0 (satsen på rad 12 i Figur 12).
- Skriv ut en ledtext (satsen på rad 16 i Figur 12).
- Läs in heltalet användaren matat in och lagra värdet i count (satsen på rad 17 i Figur 12).
- Ersätt 273 med count i "for"-satsens villkorsuttryck (uttrycket på rad 19 i Figur 12).

```
10 static void Main(string[] args)
11 {
12     int count = 0;
13     int roll = 0;
14     Random die = new Random();
15
16     Console.Write("Ange antal tärningskast [100-1000]: ");
17     count = int.Parse(Console.ReadLine());
18
19     for (int i = 0; i < count; i++)
20     {
21         roll = die.Next(1, 7);
22
23         Console.WriteLine(roll);
24     }
25 }
```

Figur 12. Variabeln count tilldelas inläst heltal av typen int.



- Hur många gånger har du hittills sparat de ändringar du gjort? Inte någon gång? HUUU! Hög tid att klicka på knappen **Save All**  som sparar alla öppna filer som har osparade ändringar. Ta för vana att spara ofta, t.ex. efter varje sats. Ctrl + S, som sparar filen som har fokus, ska sitta i ryggmärgen.
- Efter att ett antal förändringar är gjorda är det lämpligt att testa applikationen igen.



Figur 13. Resultatet av varje tärningskast skriv ut och applikationen fungerar, så långt den är skriven, som förväntat.

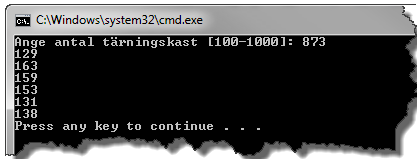
- Applikationen kan nu simulera av användaren angivet antal tärningskast. Men hur ska en frekvenstabell över tärningskasterna kunna skrivas ut? För att kunna skriva ut en frekvenstabell måste applikationen räkna antalet framslumpade ettor, tvåor, o.s.v. En tärning har sex sidor varför en variabel med plats för sex värden behövs, en så kallad array.
 - Ersätt satsen som deklarerar variabeln `roll` med en sats som skapar referensvariabeln `frequencyTable`, av typen `int[]`, och tilldelar den en referens till en array med sex element (satsen på rad 13 i Figur 14).
 - En array har 0-baserat index, d.v.s. det första elementet har index 0, varför tal i det slutna intervallet mellan 0 och 5 nu måste slumpas fram. Satsen som tilldelar `roll` ett slumpat värde ska ersättas med en sats där det framslumpade värdet leder till att motsvarande elements värde i arrayen ökas med 1 (satsen på rad 21 i Figur 14).
 - En "foreach"-loop är lämplig att använda för att skriva ut elementens värden i arrayen (satserna på rad 24-27 i Figur 14).

```
10 static void Main(string[] args)
11 {
12     int count = 0;
13     int[] frequencyTable = new int[6];
14     Random die = new Random();
15
16     Console.Write("Ange antal tärningskast [100-1000]: ");
17     count = int.Parse(Console.ReadLine());
18
19     for (int i = 0; i < count; i++)
20     {
21         frequencyTable[die.Next(0, 6)]++;
22     }
23
24     foreach (int value in frequencyTable)
25     {
26         Console.WriteLine(value);
27     }
28 }
```

Figur 14.



16. Testa applikationen och konstatera att något som kan vara en frekvenstabell skrivs ut i konsolfönstret.

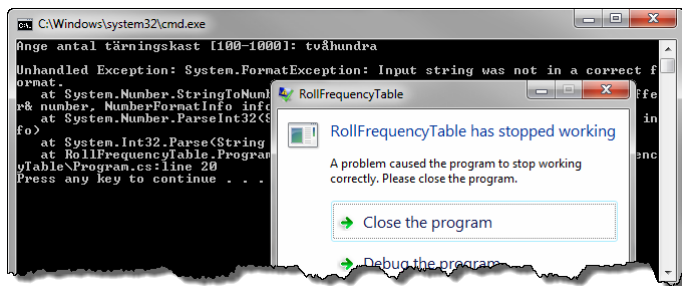


Figur 15.

Applikationen uppfyller nu de löst ställda kraven och en frekvenstabell över ett antal tärningskast skrivs ut. Men hur är det med kvaliteten på användargränssnittet?

- Användaren uppmanas att mata in ett heltal i det slutna intervallet mellan 100 och 1000. Men vad händer om användaren matar in något som inte kan tolkas som ett heltal? Vad händer om användaren matar in t.ex. 13? Ska det gå?
- Presentationen av frekvenstabellen har en del övrigt att önska. Vad betyder 129? Vad betyder 163?

Allvarligaste problemet är första punkten ovan eftersom om användaren matar in något som inte kan tolkas som ett heltal så kraschar applikationen.



Figur 16. Applikationen kraschar då användaren matar in strängen tvåhundra istället för heltalet 200.

17. Eventuella fel i samband med inläsning av antalet tärningskast som ska göras måste hanteras.

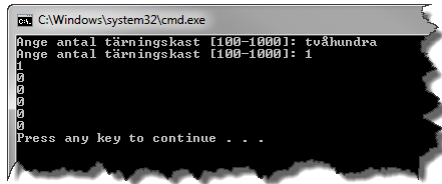
Satsen som sköter inläsningen måste ersättas av "do-while"-satsen, som får omsluta satsen som skriver ut ledtexten, på raderna 16-19 i Figur 17.

```
10 static void Main(string[] args)
11 {
12     int count = 0;
13     int[] frequencyTable = new int[6];
14     Random die = new Random();
15
16     do
17     {
18         Console.WriteLine("Ange antal tärningskast [100-1000]: ");
19     } while (!int.TryParse(Console.ReadLine(), out count));
20
21     for (int i = 0; i < count; i++)
22     {
23         frequencyTable[die.Next(0, 6)]++;
24     }
25
26     foreach (int value in frequencyTable)
27     {
28         Console.WriteLine(value);
29     }
30 }
```

Figur 17.



18. Testa applikationen och konstatera att applikationen inte längre kraschar. Det går fortfarande att mata in ett tal som inte är i det slutna intervallet mellan 1 och 100!



Figur 18. Resultat då ett värde mindre än 100 matats in.

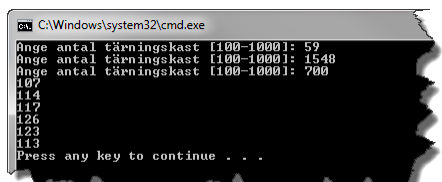
19. I samband med inläsningen måste validering ske av värdet så att det ligger i det slutna intervallet mellan 100 och 1000.

Villkorsuttrycket i ”do-while”-satsen måste kompletteras så att det undersöker om det inlästa värdet är större eller lika med 100 och om det är mindre eller lika med 1000 (*uttrycket på raderna 19-21 i Figur 19*).

```
10 static void Main(string[] args)
11 {
12     int count = 0;
13     int[] frequencyTable = new int[6];
14     Random die = new Random();
15
16     do
17     {
18         Console.WriteLine("Ange antal tärningskast [100-1000]: ");
19     } while (!(int.TryParse(Console.ReadLine(), out count) &&
20             count >= 100 &&
21             count <= 1000));
22
23     for (int i = 0; i < count; i++)
24     {
25         frequencyTable[die.Next(0, 6)]++;
26     }
27
28     foreach (int value in frequencyTable)
29     {
30         Console.WriteLine(value);
31     }
32 }
33 }
```

Figur 19.

20. Vid test av applikationen ska det nu vara omöjligt att mata in något annat än ett heltal i det slutna intervallet mellan 100 och 1000.



Figur 20.

Punkt 16 i) är nu åtgärdad. Återstår att åtgärda gör punkt 16 ii), d.v.s. att låta applikationen presentera frekvenstabellen på ett något bättre sätt så att t.ex. tärningssidan framgår:

Ettor: 107
Tvåor: 114
...
Sexor: 113



21. Applikationen måste känna till vad tärningssidorna kallas. Lämpligt är då att låta en array innehålla de strängar som beskriver tärningens sidor. Då frekvenstabellen sedan presenteras kan respektive sträng skrivas ut tillsammans med resultatet.¹
- Lägg till en sats som skapar referensvariabeln `facets`, av typen `string[]`, och initiera den med en array innehållande de sex strängar som var och en beskriver tärningens sidor (satsen på rad 14 i Figur 21).
 - Ersätt "foreach"-satsen med en "for"-sats som skriver ut varje tärningssida med tillhörande värde i frekvenstabellen (satserna på rad 29-32 i Figur 21).

```
10 static void Main(string[] args)
11 {
12     int count = 0;
13     int[] frequencyTable = new int[6];
14     string[] facets = { "Ettor", "Tvåor", "Treor", "Fyror", "Femmor", "Sexor" };
15     Random die = new Random();
16
17     do
18     {
19         Console.WriteLine("Ange antal tärningskast [100-1000]: ");
20     } while (!(int.TryParse(Console.ReadLine(), out count) &&
21             count >= 100 &&
22             count <= 1000));
23
24     for (int i = 0; i < count; i++)
25     {
26         frequencyTable[die.Next(0, 6)]++;
27     }
28
29     for (int i = 0; i < facets.Length; i++)
30     {
31         Console.WriteLine("{0}: {1}", facets[i], frequencyTable[i]);
32     }
33 }
```

Figur 21.

22. Testa applikationen på nytt. Nu ska det tydligare framgå hur många ettor, tvåor, o.s.v. som har slumpats fram. Lägg märke till att tabellen är något svår att läsa då en-, tio- och hundratal inte står under varandra.

```
C:\Windows\system32\cmd.exe
Ange antal tärningskast [100-1000]: 600
Ettor: 102
Tvåor: 118
Treor: 99
Fyror: 99
Femmor: 92
Sexor: 90
Press any key to continue . . .
```

Figur 22. Frekvenstabellen är dåligt formaterad vilket gör den svår att läsa.

23. Värden (strängar, heltal, etc.) som skrivs ut med hjälp av en formatspecifierare kan justeras på olika sätt. Den förändrade satsen på rad 31 i Figur 23 ser till att tärningssidan vänsterjusteras med en fältbredd på sex tecken, och resultatet högerjusteras med en fältbredd på fyra tecken.

```
28
29     for (int i = 0; i < facets.Length; i++)
30     {
31         Console.WriteLine("{0,-6}: {1,4}", facets[i], frequencyTable[i]);
32     }
33 }
```

Figur 23.

¹ Det är inte optimalt med en lösning som kräver synkronisering av två olika arrayer. En bättre lösning är att använda en associativ array ("dictionary"), men den konstruktionen får vänta till den avslutande delen av kursen.



24. En test av applikationen visar att den nu beträffande användargränssnittet är acceptabel, men har kanske en hel del övrigt att önska, bl.a. saknas felmeddelande vid felaktig inmatning.

```
C:\Windows\system32\cmd.exe
Ange antal tärningskast [100-1000]: sexhundra
Ange antal tärningskast [100-1000]: 1200
Ange antal tärningskast [100-1000]: 36
Ange antal tärningskast [100-1000]: 600
Ettor : 111
Tvåor : 96
Treor : 98
Fyror : 92
Femmor : 89
Sexor : 107
Press any key to continue . . .
```

Figur 24. Frekvenstabellen bättre formaterad vilket underlättar läsningen av den.

Applikationen uppfyller nu det löst ställda kraven bättre beträffande användargränssnittet. Men hur är det med kvaliteten beträffande koden?

- i. All kod är skriven i en och samma metod. Ofta vinner även mindre applikationer på att omstruktureras i flera metoder. Det blir då lättare att förstå och därmed underhålla koden.
 - ii. Avsaknaden av kommentarer är total vilket försvårar förståelse av koden.
25. Placera koden som har hand om inläsningen av antal tärningskast som ska simuleras i en separat metod med namnet `ReadNumberOfRolls()` enligt satserna på raderna 30-42 i Figur 25. Metoden anropas enligt satsen på rad 17 i Figur 25 och värdet metoden returnerar lagras i den lokala variabeln `count`.

```
8 class Program
9 {
10     static void Main(string[] args)
11     {
12         int count = 0;
13         int[] frequencyTable = new int[6];
14         string[] facets = { "Ettor", "Tvåor", "Treor", "Fyror", "Femmor", "Sexor" };
15         Random die = new Random();
16
17         count = ReadNumberOfRolls();
18
19         for (int i = 0; i < count; i++)
20         {
21             frequencyTable[die.Next(0, 6)]++;
22         }
23
24         for (int i = 0; i < facets.Length; i++)
25         {
26             Console.WriteLine("{0,-6}: {1,4}", facets[i], frequencyTable[i]);
27         }
28     }
29
30     private static int ReadNumberOfRolls()
31     {
32         int count = 0;
33
34         do
35         {
36             Console.Write("Ange antal tärningskast [100-1000]: ");
37             while (!(int.TryParse(Console.ReadLine(), out count) &&
38                 count >= 100 &&
39                 count <= 1000));
40
41             return count;
42         }
43     }
44 }
```

Figur 25.



26. Även skapandet av frekvenstabellen och presentationen av den kan omstruktureras i två separata metoder enligt Figur 26.

Trots att koden fortfarande inte innehåller några kommentarer är den lättare att förstå då metodernas namn har valts med omsorg och beskriver väl vad respektive metod har till uppgift.

```
8 class Program
9 {
10     static void Main(string[] args)
11     {
12         int count = 0;
13         int[] frequencyTable = null;
14
15         count = ReadNumberOfRolls();
16
17         frequencyTable = RollDie(count);
18
19         ViewFrequencyTable(frequencyTable);
20     }
21
22     private static int ReadNumberOfRolls()
23     {
24         int count = 0;
25
26         do
27         {
28             Console.WriteLine("Ange antal tärningskast [100-1000]: ");
29         } while (!(int.TryParse(Console.ReadLine(), out count) &&
30             count >= 100 &&
31             count <= 1000));
32
33         return count;
34     }
35
36     private static int[] RollDie(int count)
37     {
38         int[] frequencyTable = new int[6];
39         Random die = new Random();
40
41         for (int i = 0; i < count; i++)
42         {
43             frequencyTable[die.Next(0, 6)]++;
44         }
45
46         return frequencyTable;
47     }
48
49     private static void ViewFrequencyTable(int[] frequencyTable)
50     {
51         string[] facets = { "Ettor", "Tvåor", "Treor", "Fyror", "Femmor", "Sexor" };
52
53         for (int i = 0; i < facets.Length; i++)
54         {
55             Console.WriteLine("{0,-6}: {1,4}", facets[i], frequencyTable[i]);
56         }
57     }
58
59 }
```

Figur 26.

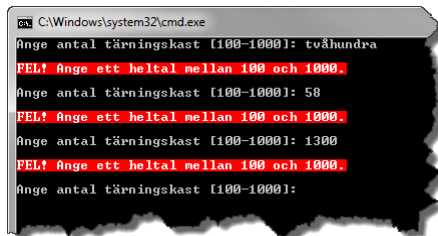


27. Då användaren råkar mata in något som inte kan tolkas som ett heltal i det slutna intervallet mellan 100 och 1000 visas inget felmeddelande. Metoden `ReadNumberOfRolls()` måste skrivas om så att så sker (*satserna på raderna 39-48 och 50-58 i Figur 27*).

```
35 private static int ReadNumberOfRolls()
36 {
37     int count = 0;
38
39     while (true)
40     {
41         try
42         {
43             Console.Write("Ange antal tärningskast [100-1000]: ");
44             count = int.Parse(Console.ReadLine());
45             if (count < 100 || count > 1000)
46             {
47                 throw new ApplicationException();
48             }
49             return count;
50         }
51         catch
52         {
53             Console.BackgroundColor = ConsoleColor.Red;
54             Console.ForegroundColor = ConsoleColor.White;
55             Console.WriteLine("\nFEL! Ange ett heltal mellan 100 och 1000.\n");
56             Console.ResetColor();
57         }
58     }
59 }
```

Figur 27.

28. Testa applikationen genom att mata in några felaktiga värden.



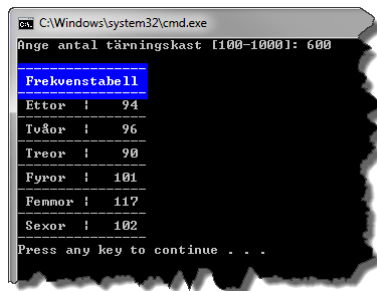
Figur 28. Felmeddelande visas då användaren matar in något som inte kan tolkas som ett heltal i det slutna intervallet mellan 100 och 1000.

29. Presentationen av frekvenstabellen behöver åtgärdas så den liknar en tabell (*satserna på raderna 92-97 och 100-101 i Figur 29*).

```
88 private static void ViewFrequencyTable(int[] frequencyTable)
89 {
90     string[] facets = { "Ettor", "Tvåor", "Treor", "Fyror", "Femmor", "Sexor" };
91
92     Console.BackgroundColor = ConsoleColor.Blue;
93     Console.ForegroundColor = ConsoleColor.White;
94     Console.WriteLine("\n-----");
95     Console.WriteLine(" Frekvenstabell ");
96     Console.WriteLine("-----");
97     Console.ResetColor();
98     for (int i = 0; i < facets.Length; i++)
99     {
100         Console.WriteLine(" {0,-7} | {1,4}", facets[i], frequencyTable[i]);
101         Console.WriteLine("-----");
102     }
103 }
```

Figur 29.

30. Testa applikationen och verifiera att frekvenstabellen nu påminner mer om en tabell än tidigare.



```
C:\Windows\system32\cmd.exe
Ange antal tärningskast [100-1000]: 600

Frekvenstabell
Ettor | 94
Tvåor | 96
Treor | 90
Fyror | 101
Femnor | 117
Sexor | 102
Press any key to continue . . .
```

Figur 30. Frekvenstabellen presenterad på mer tilltalande sätt.

31. Kod ska alltid dokumenteras. Figur 31 visar hur källkoden blir enklare att förstå när den kompletterats med kommentarer.

```
1 using System;
2
3 namespace RollFrequencyTable
4 {
5     /// <summary>
6     /// Applikationen simulerar tärningskast och presenterar utfallet i
7     /// form av en frekvenstabell.
8     /// </summary>
9     class Program
10    {
11        /// <summary>
12        /// Startpunkt för applikationen.
13        /// </summary>
14        /// <param name="args">Argument som kan skickas till applikationen (används
15        /// inte).</param>
16        static void Main(string[] args)
17        {
18            // Deklarerar lokala variabler.
19            int count = 0;
20            int[] frequencyTable = null;
21
22            // Läser in hur många tärningskast som ska simuleras då
23            // en frekvenstabell över tärningskast skapas och presenteras.
24            count = ReadNumberOfRolls();
25
26            frequencyTable = RollDie(count);
27
28            ViewFrequencyTable(frequencyTable);
29        }
30
31        /// <summary>
32        /// Efterfrågar, läser in och returnerar antalet tärningskast som ska simuleras.
33        /// </summary>
34        /// <returns>Antal tärningskast.</returns>
35        private static int ReadNumberOfRolls()
36        {
37            // Deklarerar lokal variabel.
38            int count = 0;
39
40            // Läser in och returnerar ett heltal mellan 100 och 1000.
41            while (true)
42            {
43                try
44                {
45                    Console.Write("Ange antal tärningskast [100-1000]: ");
46                    count = int.Parse(Console.ReadLine());
47                    if (count < 100 || count > 1000)
48                    {
49                        throw new ApplicationException();
50                    }
51                }
52                catch { }
53            }
54        }
55    }
56 }
```



```
26         frequencyTable = RollDie(count);
27
28         ViewFrequencyTable(frequencyTable);
29     }
30
31     /// <summary>
32     /// Efterfrågar, läser in och returnerar antalet tärningskast som ska simuleras.
33     /// </summary>
34     /// <returns>Antal tärningskast.</returns>
35     private static int ReadNumberOfRolls()
36     {
37         // Deklarerar lokal variabel.
38         int count = 0;
39
40         // Läser in och returnerar ett heltal mellan 100 och 1000.
41         while (true)
42         {
43             try
44             {
45                 Console.WriteLine("Ange antal tärningskast [100-1000]: ");
46                 count = int.Parse(Console.ReadLine());
47                 if (count < 100 || count > 1000)
48                 {
49                     throw new ApplicationException();
50                 }
51                 return count;
52             }
53             catch
54             {
55                 Console.BackgroundColor = ConsoleColor.Red;
56                 Console.ForegroundColor = ConsoleColor.White;
57                 Console.WriteLine("\nFEL! Ange ett heltal mellan 100 och 1000.\n");
58                 Console.ResetColor();
59             }
60         }
61     }
62
63     /// <summary>
64     /// Simulerar tärningskast, skapar och returnerar frekvenstabell över utfallet.
65     /// </summary>
66     /// <param name="count">Antal tärningskast att simulera.</param>
67     /// <returns>Array innehållande frekvenstabell över simulerade
68     /// tärningskast.</returns>
69     private static int[] RollDie(int count)
70     {
71         // Deklarerar lokala variabler.
72         int[] frequencyTable = new int[6];
73         Random die = new Random();
74
75         // Slumpar tärningskast och uppdaterar frekvenstabellen som därefter
76         // returneras.
77         for (int i = 0; i < count; i++)
78         {
79             frequencyTable[die.Next(0, 6)]++;
80         }
81
82         return frequencyTable;
83     }
84
85     /// <summary>
86     /// Presenterar en frekvenstabell över tärningskast.
87     /// </summary>
88     /// <param name="frequencyTable">Referens till frekvenstabell i form av en array
89     /// innehållande utfallet av tärningskast.</param>
90     private static void ViewFrequencyTable(int[] frequencyTable)
91     {
92         // Deklarerar lokal variabel.
```



```
80     }
81
82     return frequencyTable;
83 }
84
85 /// <summary>
86 /// Presenterar en frekvenstabell över tärningskast.
87 /// </summary>
88 /// <param name="frequencyTable">Referens till frekvenstabell i form av en array
89 /// innehållande utfallet av tärningskast.</param>
90 private static void ViewFrequencyTable(int[] frequencyTable)
91 {
92     // Deklarerar lokal variabel.
93     string[] facets = { "Ettor", "Tvåor", "Treor", "Fyror", "Femmor", "Sexor" };
94
95     // Går igenom tärningssida för tärningssida och skriver ut tärningssidans
96     // namn samt antalet gånger sidan "kommit upp" vid ett tärningskast.
97     Console.BackgroundColor = ConsoleColor.Blue;
98     Console.ForegroundColor = ConsoleColor.White;
99     Console.WriteLine("\n-----");
100    Console.WriteLine(" Frekvenstabell ");
101    Console.WriteLine("-----");
102    Console.ResetColor();
103    for (int i = 0; i < facets.Length; i++)
104    {
105        Console.WriteLine(" {0,-7} | {1,4}", facets[i], frequencyTable[i]);
106        Console.WriteLine("-----");
107    }
108 }
109
110 }
111 }
```

Figur 31.