3.7 Hantering av slumptal i C#

En nackdel av programmet **GuessDo** är att det hemliga talet är hårdkodat som **17**. Det skulle innebära en väsentlig förbättring av *Gissa tal* om programmet kunde generera ett slumptal mellan 1 och 20 som hemligt tal varje gång man körde det. Därför öppnar vi här en liten parentes om slumptal av typ **int** och deras hantering.

Generellt kan man med datorn som en deterministisk maskin som datorn är, inte producera äkta slumptal utan endast *simulera* dvs på något sätt *beräkna* s.k. *pseudo*slumptal enligt en viss algortim. Överallt vi pratar om slumptal menar vi egentligen pseudoslumptal. I C# kan man simulera slumptal på olika sätt, bl.a. med klassen **Random** och dess metod **Next()** som returnerar slumptal av typ **int** mellan **1** och **int.MaxValue**, om den anropas utan parameter. En annan variant av **Next()** returnerar slumptal mellan sina parametrar, närmare bestämt:

```
a <= r.Next(a, b) < b
```

där **r** är ett objekt klassen **Random**. För att skräddarsy metoden **Next(a, b)** till att få slumptal mellan **1** och **20** måste vi anropa **r.Next(1, 21)**. Följande program testar båda varianter av **Next()**:

```
// DoRand.cs
// Skriver ut 5 slumptal mellan 1 och int.MaxValue samt
//            20         mellan 1 och 20
// Anropar två varianter av Random-metoden Next() en gång
// med ingen parameter, en gång med två paramtrar
using System;
class DoRand
{
  static void Main()
  {
    int i = 1, j = 1;
    Random r = new Random(); // Objekt av klassen Random
    Console.WriteLine("Slumptal mellan 1 & int.MaxValue:\n");
    do                       // do-loop
       Console.WriteLine("\t" + r.Next());
    while (i++ < 5);         // i testas först, ökar sedan

    Console.WriteLine("\nSlumptal mellan 1 och 20:\n\t");
    do                       // do-loop
       Console.Write(r.Next(1, 21) + "\t");
    while (j++ < 20);        // j testas först, ökar sedan

    Console.WriteLine('\n');
  }
}
```

En körning av **DoRand** ger följande resultat:

```
Slumptal mellan 1 & int.MaxValue:

        1460841191
        225482400
        1438321568
        1700127070
        1513406452

Slumptal mellan 1 och 20:

7     20    2     12    12    14    3     16    3     15
2     15    12    9     1     10    14    15    1      2
```

För det första ser man att vi får endast heltal vilket beror på att båda metoderna **`Next()`** och **`Next(a, b)`** returnerar **`int`**. Vill man ha decimalslumptal finns det en annan metod i klassen **`Random`** som heter **`NextDouble()`**. För det andra har vi fått i intervallet [1, 20] även randvärdena **`1`** och **`20`**. Hade vi anropat **`r.Next(1, 20)`** hade vi fått slumptal mellan **`1`** och **`19`** eftersom den andra parametern *inte* ingår i slumptalsgenereringen. Så, anropet **`r.Next(1, 21)`** ger slumptal mellan **`1`** och **`20`**.

När det gäller de båda varianterna av metoden **`Next()`** ger den ena utan parameter de stora slumptalen i utskriften ovan mellan **`1`** och **`int.MaxValue`** och den andra med två parametrar de små slumptalen mellan **`1`** och **`20`**. Två olika **`do`**-satser i **`Do-Rand`** tar hand om slumptalen i dessa två olika intervall. I den första **`do`**-satsen anropas **`Next()`** utan parameter, i den andra med två parametrar. Vi har här att göra med ett koncept i programmering som kallas *överlagring av metoder*. Innebörden är att det är *två olika metoder* med *samma namn*, men olika parameterlistor. I anropet avgörs vilken av dem det gäller därför att parameterlistan avslöjar identiteten – både för oss och kompilatorn. C#-biblioteket är fullt med överlagrade metoder. De flesta bibliotheksklasserna har t.o.m. flera överlagrade metoder dvs *flera* olika metoder med samma namn.

Array av slumptal

Eftersom vi i fortsättningen kommer att jobba med flera program som använder slumptal lagrade i en array vill vi skriva en metod som kan användas av alla dessa program. Vi har valt formen av en **`void`**-metod för att generera ett antal slumpvärden och tilldela dem till elementen i en array:

```
// RandArray.cs
// Ny metod Rand() slumpar fram en array av heltal mellan
// a och b, lagrar dem i arrayen no och skriver ut dem
// Anropar biblioteksmetoden Next() i en loop för att få
// ETT slumptal i varje varv

using System;
```

```
class RandArray
{
   public static void Rand(Random r, int[] no, int a, int b)
   {
      Console.Write("\n\t" + no.Length + " heltal mellan " +
                  a + " och " + b + " slumpas fram:\n\n\t");
      for (int i=0; i < no.Length; i++)
      {
          no[i] = r.Next(a, b);
          Console.Write(no[i] + " ");
          if ((i % 16 == 0) && (i != 0))
             Console.Write("\n\t");
      }
      Console.WriteLine("\n\n");
   }
}
```

För förståelse av biblioteksmetoden **`Next()`** hänvisas till hantering av slumptal. Det nya i koden ovan är att slumptalen lagras i en array som kommer att användas av fler program vilket demonstrerar inte bara modularisering utan även återanvändning av kod. Filen ovan innehåller inte ett fullständigt program utan endast en klass med **`void`**-metoden **`Rand()`** som har fyra parametrar varav den ena är en array av **`int`**, kallad **`no`** som lagrar slumptalen. Arrayen deklareras i parameterlistan och tilldelas i kroppen mellan **`a`** och **`b`** via satsen:

```
no[i] = r.Next(a, b);
```

som i en **`for`**-sats anropar den biblioteksmetoden **`Next()`** som i sin tur i varje varv av loopen slumpar fram *ett* slumptal mellan **`a`** och **`b`**. Vi har använt denna metod tidigare i andra program. **`for`**-satsen som anropar metoden skriver ut slumptalen.