Objektorienterad programmering
Föreläsning 5

Mahmud Al Hakim
mahmud@dynamicos.se
www.webbacademy.se

Agenda

- UML Övning
- Mer om metoder
- Standardklassen String
- Konstruktorer
- Överlagrade metoder
- Standardklassen Random
- Struct-typer

Rita ett klassdiagram som beskriver följande klass

```
class Person
{
    // Instansvariabler
    int ålder;
    double vikt; // kg
    double längd; // m

    // Metoder
    public void SättÅlder(int age)
    {
        ålder = age;
    }
    public void SättVikt(double weight)
    {
        vikt = weight;
    }
    public void SättLängd(double lenght)
    {
        längd = lenght;
    }
    public void FyllerÅr()
    {
        ålder = ålder+1;
    }
}
```


Metoder (instansmetoder)

- Metoder beskriver vad man kan göra med objekt.
- En **metoddeklaration** består av två delar, ett huvud och en kropp.
- I huvudet talar man om hur metoden ska användas.
- Kroppen omges av klamrar och beskriver vad som ska ske när metoden anropas.
- Metoder kan returnera ett värde med hjälp av en **return-sats**.
- För att använda en metod måste man **anropa metoden.** Detta görs med hjälp av **punktoperatorn.**

  referensnamn**.**metodnamn(argument)

Exempel – Klassen Tidpunkt

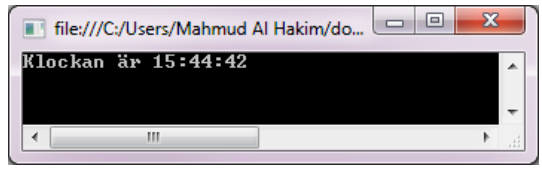
```
1   using System;
2
3   class Tidpunkt
4   {
5       // instansvariabler
6       int tim, min, sek;
7
8       // metoder
9       public void SetTime(int t, int m, int s)
10      { tim = t; min = m; sek = s; }
11
12      public void Tick() // stegar fram tiden en sekund
13      {
14          if (++sek == 60)    { sek = 0; ++min; }
15          if (min   == 60)    { min = 0; ++tim; }
16          if (tim   == 24)    { tim = 0; }
17      }
18
19      public override string ToString() // ger t:m:s
20      {
21          string tid = tim + ":" + min + ":" + sek;
22          return tid;
23      }
24  }
```


Klassen TidDemo

```
1   using System;
2
3   class TidDemo
4       {
5           static void Main(string[] args)
6           {
7               Tidpunkt tp = new Tidpunkt();
8
9               tp.SetTime(12, 00, 00);
10              Console.WriteLine(tp);
11              tp.Tick();
12              Console.WriteLine(tp);
13
14              Console.ReadKey();
15          }
16      }
```

Visa aktuellt klockslag i kommandofönstret

```
1   using System;
2
3   class TidDemo
4       {
5           static void Main(string[] args)
6           {
7               DateTime dt = DateTime.Now;
8               Tidpunkt tp = new Tidpunkt();
9               tp.SetTime(dt.Hour, dt.Minute, dt.Second);
10              Console.WriteLine("Klockan är " + tp);
11
12              Console.ReadKey();
13          }
14      }
```



Standardklassen String

- Det finns många egenskaper och metoder i klassen String. Några viktiga är:
- Egenskapen **Length**: ger antalet tecken.
- Metoden **ToUpper**: ger en kopia av strängen där alla bokstäver ersatts med stora.
- Metoden **ToLower**: ger en kopia av strängen där alla bokstäver ersatts med små.
- Metoden **Substring**: ger en deltext .
- Metoden **Replace(s1,s2)**: byter ut s1 mot s2

String – Några exempel

```
String s1 = "Välkommen till C#";
Console.WriteLine("Original sträng: " + s1);
Console.WriteLine("Antal tecken: " + s1.Length);
Console.WriteLine("Visa stora bokstäver: " + s1.ToUpper() );
Console.WriteLine("Visa små bokstäver: " + s1.ToLower() );
Console.WriteLine("Visa från tecken nr 10: " + s1.Substring(10) );
Console.WriteLine("Visa tecken 0-10: " + s1.Substring(0,10) );
Console.WriteLine("Vi byter ut ett ord: " +
    s1.Replace("C#","Visual Studio") );
```


Konstruktorer – initiering av objekt

- En konstruktor är en speciell initieringsmetod som **anropas automatiskt** varje gång man skapar ett objekt av den aktuella klassen.
- Har samma namn som klassen.
- Får inte ha returtyp.
- Får finnas flera med olika parametrar eller olika typer av parametrar.
- En parameterlös konstruktor kallas **defaultkonstruktor**.
- En konstruktor kan anropa en annan konstruktor med hjälp av en konstruktorinitierare (**:this)**

Konstruktorer – Exempel Klassen Tidpunkt

```
// Defaultkonstruktor
public Tidpunkt() { }

// En konstruktor med 3 parameterar
public Tidpunkt(int t, int m, int s){
    SetTime(t,m,s);
}

// Konstruktor anropar en annan konstruktor med hjälp av en konstruktorinitierare
public Tidpunkt(int t, int m) : this(t, m, 0) { }
```


Överlagrade metoder

- Man får ha hur många metoder som helst med samma namn men metoderna måste ha olika parametrar eller olika typer av parametrar.
- Metoder som har samma namn kallas **överlagrade metoder.**
- När man anropar en överlagrad metod avgör kompilatorn vilken av de överlagrade metoderna man menar genom att studera vilka argument man angivit (smart, eller hur!)

Överlagrade metoder - Exempel

```
public void Tick(int antalSek)
{    while (antalSek > 0)
     {
          Tick();
          antalSek = antalSek -1;
     }
}
```


Standardklassen Random

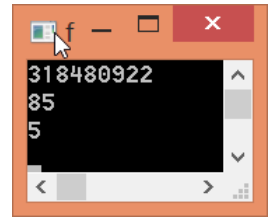
- I C# finns en standardklass som heter Random.
- Om man skapar ett objekt av denna klass får man en slumpgenerator. T.ex.

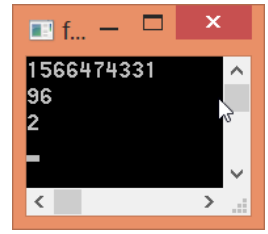
  Random r = new Random();
- Metoden Next() ger ett slumptal av typen int i intervallet 0 till 2147483647
- Metoden Next(b) ger ett slumptal av typen int i intervallet 0 till b-1
- Metoden Next(a,b) ger ett slumptal av typen int i intervallet a till b-1

Random - Exempel

```
static void Main(string[] args)
{
    Random r = new Random();
    Console.WriteLine( r.Next() );
    Console.WriteLine( r.Next(100) );
    Console.WriteLine( r.Next(1, 7) );

    Console.ReadKey();
}
```





Structer (Struct-typer)

- För att konstruera små, enkla klasser använder man i C# **structer**.
- Structer kan liksom klasser innehålla variabler, konstruktorer och metoder.
- När man deklarerar en variabel av struct-typ får man en **databehållare**, inte någon referens. (structer är värdetyper)
- Det finns en parameterlös defaultkonstuktor. Man får inte skriva någon egen sådan!
- Operatorn new skapar inte något nytt objekt. Den anropar bara en konstruktor.
- I en struct ges instansvariablerna ofta egenskapen public så att de blir synliga utanför structen.

Struct - Exempel

```
struct Cirkel
{
    public double x, y; // Mittpunktens koordinater
    public double radie;
    // Konstruktor
    public Cirkel(int xx, int yy, int rr) {
        x = xx; y = yy; radie = rr;
    }
    // Metoder
    public double Area() {
        return Math.PI * radie * radie;
    }

}
```


Struct – Exempel forts.

```
static void Main(string[] args)
{
    Cirkel c = new Cirkel();
    c.radie = 10;
    Console.WriteLine("Area: " + c.Area() );
    Console.ReadKey();
}
```