Objektorienterad programmering
Föreläsning 6

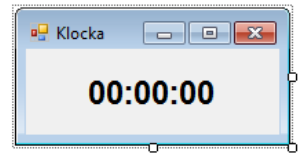
Mahmud Al Hakim
mahmud@dynamicos.se
www.webbacademy.se

Agenda

- Mer om klasser och typer
- Namnrymder
- Inkapsling
- Synlighet
- Statiska variabler
- Statiska metoder

Visa aktuellt klockslag i ett fönster

- Skapa ett nytt projekt med hjälp av mallen ”Windows Forms Application”.
- Ändra fönstrets egenskaper
  - Ändra namnet till Klocka
  - Ändra storlek till 200x100
  - Döp om filen Form1 till Klockvisare
- Skapa en etikett (Label)
  - Ändra namnet till a
  - Ändra egenskapen Text till 00:00:00
  - Ändra Font till Arial, 18, fet stil
  - Centrera etiketten
- Lägg till filen Tidpunkt.cs till projektet.
- Lägg till en Timer (finns i gruppen Components)
  - Ändra egenskapen Interval till 1000 (en sekund)
  - Ändra egenskapen Enabled till true
  - Dubbelklicka på Timer-ikonen för att skapa en automatisk Tick-metod.



Uppdatera klassen Klockvisare

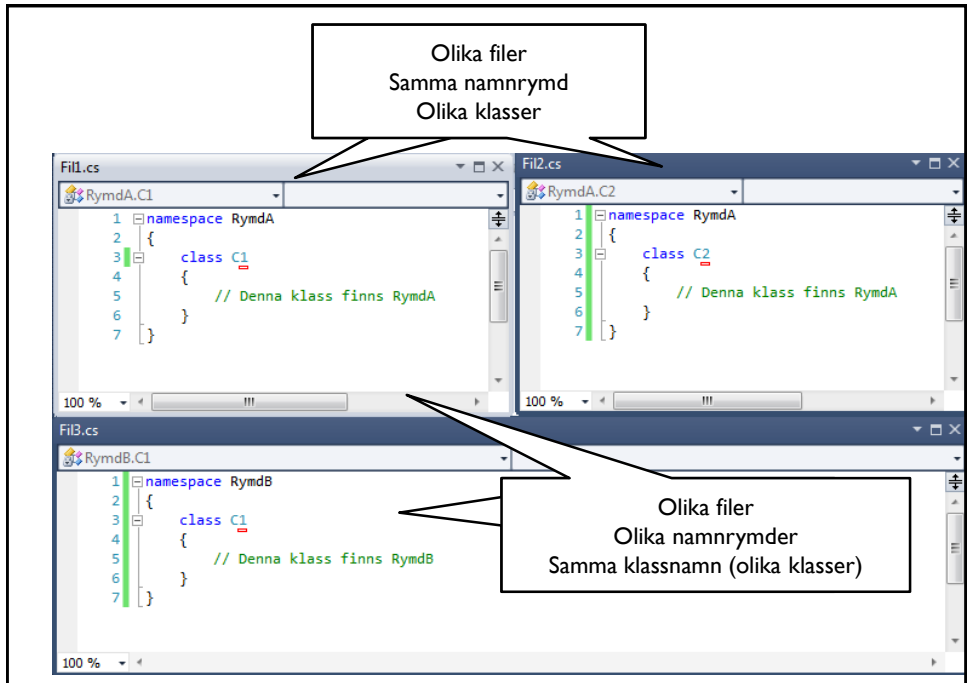
```
namespace VisaKlocka
{
    public partial class Klockvisare : Form
    {
        Tidpunkt tp = new Tidpunkt();
        public Klockvisare()
        {
            InitializeComponent();
            DateTime dt = DateTime.Now;
            tp.SetTime(dt.Hour, dt.Minute, dt.Second);
            a.Text = tp.ToString();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            tp.Tick();
            a.Text = tp.ToString();

        }
    }
}
```


Tips

```
public override string ToString() // ger tt:mm:ss
{
    string tid="";

    if(tim < 10) tid += "0";
    tid += tim + ":";

    if(min < 10) tid += "0";
    tid += min + ":";

    if(sek < 10) tid += "0";
    tid += sek;

    return tid;
}
```


Namnrymder (namespace)

- Risken att två olika klasser givits samma namn i olika programtextfiler är stor.
- För att råda bot på detta kan man använda namnrymder.
- Man kan då kapsla in en grupp av typdeklarationer i en viss namnrymd.
- Deklarationerna kan då inte kollidera med deklarationer som gjorts utanför den aktuella namnrymden.



Sub-namnrymder

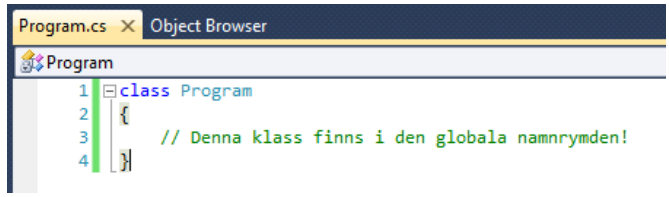
- En namnrymd kan innehålla andra namnrymder (sub-namnrymder).
- En namnrymd som innehåller en sub-namnrymd kallas super-namnrymd.
- Man kan alltså bygga upp en hierarkisk struktur.
- Klassbibliotek skapas på detta sätt.
- Standardklasserna i C# är organiserade i namnrymder enligt denna modell.
- T.ex. Supernamnrymden System som innehåller Sub-namnrymden Windows som i sin tur innehåller sub-namnrymden Forms.

Användning av namnrymder

- **Explicit namngivning**
  - Ge det fullständiga namnet vid användning:
  - Superrymdnamn.Subrymdnamn.namn (klass, struct m.m.)
  - T.ex. System.Windows.Forms.MessageBox…
- **Using-direktiv**
  - Ett using-direktiv importerar de typdeklarationer som finns i den angivna namnrymden
  - using namnrymd;
  - T.ex. using System.Windows.Forms;

Global namespace (namnlös namnrymd)

- Det är tillåtet att lägga typdeklarationer ytterst i programmet (top-level) så att de inte är inkapslade i någon namespace-deklaration.
- Dessa deklarationer hamnar då i den s.k. globala namnrymden.



Inkapsling och synlighet

- En av huvudidéerna när det gäller objektorienterad programmering att programmen skall delas in i klart avgränsade delar.
- Exempelvis, instansvariabler. Dessa måste **kapslas in (gömmas)** i en klass och får inte vara synliga och tillgängliga utanför klassen.
- För att ange synligheten för en medlem (t.ex. instansvariabel eller metod) i en klass använder man sig av de reserverade orden: private, public, internal och protected.
- De vanligaste är private och public.
- Om man inte anger något antas att man skrivit private.

Synlighet för medlemmar i klasser

- Det som deklareras som **private** är bara synligt inne i klassen själv.
- Det som deklareras som **public** kan användas överallt.
- Ordet **internal** används för att ange att en medlem är synlig i klasser som ingår i samma assemblyfil som den klass som medlemmen tillhör.
- Ordet **protected** används i samband med arv (mer om detta senare).

Synlighet – Exempel 1

```
1  class Synlighet
2  {
3      int i1;         // blir private automatiskt
4      private int i2; // Osynlig utanför klassen
5      public int i3;  // Synlig utanför klassen
6  }
7  class Test
8  {
9      static void Main(string[] args)
10     {
11         Synlighet s = new Synlighet();
12         s.i1 = 3; // FEL
13         s.i2 = 5; // FEL
14         s.i3 = 7; // OK
15     }
16
17 }
```

```
int Synlighet.i2

Error:
  'Synlighet.i2' is inaccessible due to its protection level
```


Synlighet – Exempel 2

```
1  class Synlighet
2  {
3      void metod1(){          // Blir private automatiskt
4          System.Console.WriteLine("Metod 1");
5      }
6      private void metod2(){  // Osynlig utanför klassen
7          System.Console.WriteLine("Metod 2");
8      }
9      public void metod3(){   // Synlig utanför klassen
10         System.Console.WriteLine("Metod 3");
11     }
12 }
13 class Test
14 {
15     static void Main(string[] args)
16     {
17         Synlighet s = new Synlighet();
18         s.metod1(); // FEL
19         s.metod2(); // FEL
20         s.metod3(); // OK
21         System.Console.ReadKey();
22     }
23
24 }
```

```
void Synlighet.metod1()

Error:
  'Synlighet.metod1()' is inaccessible due to its protection level
```

Statiska medlemmar

- Ibland har man medlemmar som är *gemensamma* för alla de objekt som tillhör en viss klass.
- I C# kallas sådana medlemmar **statiska medlemmar**.
- Man använder sig av det reserverade ordet **static** när man deklarerar statiska medlemmar.

Statiska variabler

- En statisk variabel är en variabel som är deklarerad inne i en klass och som bara finns **på ett enda ställe**.
- Alltså, **finns bara i en upplaga**, gemensam för alla objekt som tillhör klassen.
- Om man i någon instansmetod förändrar en statisk variabel, så kommer ändringen att påverka alla objekt av den aktuella klassen, efterson alla delar på den statiska variabeln.
- Exempel
  En variabel som håller reda på hur många instanser av en viss klass som finns för närvarande!

Statiska variabler och metoder

```
1  using System;
2  class StatiskaVariabler
3  {
4      static int antal = 0;              // En statisk variabel
5
6      public StatiskaVariabler()         // Default konstruktor
7      {   antal++;    }
8
9      public static int GetAntalObjekt()  // En statisk metod
10     {   return antal;   }
11
12 }
13 class Test
14 {
15     static void Main(string[] args)
16     {
17         StatiskaVariabler sv1 = new StatiskaVariabler();
18         Console.WriteLine(StatiskaVariabler.GetAntalObjekt());
19         StatiskaVariabler sv2 = new StatiskaVariabler();
20         Console.WriteLine(StatiskaVariabler.GetAntalObjekt());
21         Console.ReadKey();
22     }
23 }
```


Statiska egenskaper

```
1   using System;
2
3   class Konto
4   {
5       // Statiska variabler
6       static double räntesats;
7
8       // Statiska egenskaper
9       public static double Räntesats
10      {
11          get {   return räntesats;   }
12          set {   räntesats = value;  }
13      }
14  }
15
16  class KontoTest
17  {
18      static void Main(string[] args)
19      {
20          Konto.Räntesats = 4.5;
21          Console.WriteLine(Konto.Räntesats);
22          Console.ReadKey();
23      }
24  }
```