Objektorienterad programmering
Föreläsning 11

Mahmud Al Hakim
mahmud@webacademy.se
www.webacademy.se

Agenda

- Multipla relationer
- Flerdimensionella fält
- Nationella inställningar
- Redigering av utskrifter (format)
- Uppdelning av text (split)
- Felkontroll med TryParse
- Textfiler

Multipla relationer

- Fält och listor kan användas för att beskriva relationer där ett objekt har flera delkomponenter av samma slag eller **känner till** flera andra objekt, vilka alla är av samma typ.
- Man deklarerar då instansvariabler som är fält eller listor.
- Exempel
  En person kan ha ett eller flera barn.

Multipla relationer - Exempel

```
class Person
{
    string namn, adress;
    Person makeMaka; // referens till en annan person

    // barn är en lista av typen Person
    List<Person> barn = new List<Person>();

    // Metoden NyttBarn anropas när en person får ett nytt barn
    public void NyttBarn(Person b)
    {
        barn.Add(b);    // b är en referens till ett barn
    }

    // Egenskapen Barn returnerar en referens till en kopia av
    // den lista instansvariabeln barn refererar till
    public List<Person> Barn
    {
        get
        {
            return new List<Person>(barn);
        }
    }

    // Övriga metoder som tidigare (se känner-till-relationer)
```

Klassen Person – fort.

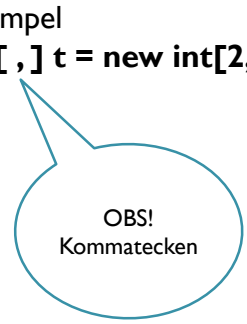
```
    public Person (string namn) // Konstuktor
    {   this.namn = namn;   }

    // Egenskaper
    public string Namn {
        get {   return namn;      }
    }
    public string Adress {
        get {   return adress;    }
        set {   adress = value;   }
    }

    // Metoder
    public void Bröllop (Person p){
        makeMaka = p;       // ordna en referens till maken
        p.makeMaka = this;  // låt maken referera till denna person
    }
    public void Skilsmässa(){
        makeMaka.makeMaka = null;
        makeMaka = null;
    }
    public Person GiftMed(){
        return makeMaka;
    }
}
```

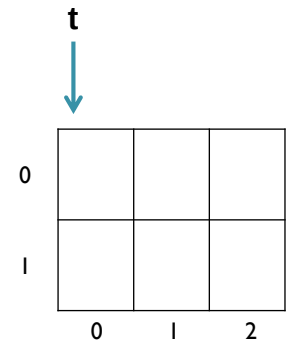

Multipla relationer - Testprogram

```
static void Main(string[] args)
{
    Person p = new Person("Anders");

    // Lägg till två barn
    p.NyttBarn(new Person("Maria") );
    p.NyttBarn(new Person("Kalle") );

    // Skapa en lista och hämta alla barn
    List<Person> b = p.Barn;

    Console.WriteLine(p.Namn + " har " + b.Count + " barn:" );
    foreach (var item in b)
    {
        Console.WriteLine(item.Namn);
    }

    Console.ReadKey();
}
```

Flerdimensionella fält

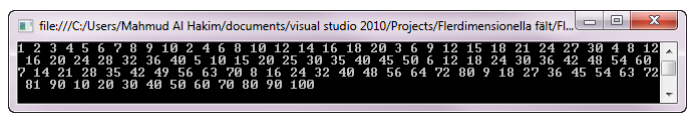
- I C# kan man använda sig av fält med flera index, s.k. *flerdimensionella fält*.
- Tvådimensionella uppställningar kallas tabeller eller **matriser**.
- Exempel
  int[,] t = new int[2,3]





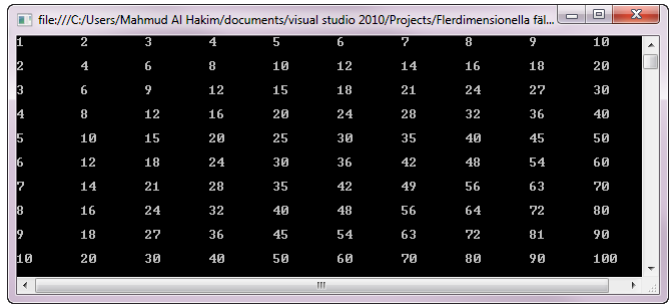
Flerdimensionella fält - Exempel

```
// En matris 10 rader x 10 kolumner
int[,] tabell = new int[10, 10];

// Metoden GetLength hämtar antal rader eller kolumner
Console.WriteLine("Antal rader: " + tabell.GetLength(0));
Console.WriteLine("Antal kolumner : " + tabell.GetLength(1));

// Med hjälp av nästlade for-satser kommer man åt alla komponenter
for (int i = 0; i < tabell.GetLength(0); i++)
    for (int j = 0; j < tabell.GetLength(1); j++)
        tabell[i, j] = (i + 1) * (j + 1);

// Foreach löper igenom komponenterna radvis
foreach (var item in tabell)
{
    Console.Write(item + " ");
}
```



Nästlade for-satser för att skriva ut

```
for (int i = 0; i < tabell.GetLength(0); i++)
{
    for (int j = 0; j < tabell.GetLength(1); j++)
    {
        Console.Write(tabell[i, j] + "\t");
    }

    Console.WriteLine("");
}
```



Nationella inställningar

- Konventioner för olika språk hanteras med hjälp av standardklassen **CultureInfo.**
- Klassen CultureInfo finns i namnrymden System.Globalization
- Ett objekt av denna klass beskiver en lokal konvention.
- De lokala konventionerna bestämmer sådant som format för datum, tider och numeriska tal.
- Det finns alltid ett default CultureInfo-objekt som från början sätts av systemet.

CultureInfo - Exempel

```
static void Main(string[] args)
{
    // Visa aktuell lokal konvention
    // Ger sv-SE om operativsystemet är svenskt
    Console.WriteLine(CultureInfo.CurrentCulture);

    // Ange amerikanska konvensioner istället
    Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");

    Console.WriteLine(CultureInfo.CurrentCulture);

    Console.ReadKey();
}
```


Redigering av utskrifter

- För att få ”snyggare” utskrifter t.ex. ange antal decimaler eller redigera som ett belopp behöver man formatera text.
- Tekniken finns för metoderna Write, WriteLine och string.Format.
- Anrop sker på följande sätt

  metodnamn(format, v0, v1, v2,…)
- v0 ,v1 etc. är de värden som skall redigeras.
- Parametern format kan innehålla vanlig text och **platsmarkörer**.

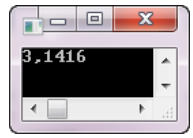
Platsmarkörer

- Platsmarkörer har följande form
  { index , bredd : formatsträng }
- Index: anger numret på det värde som skall redigeras.
- Bredd (kan utelämnas): anger det minsta antalet positioner. Utfyllnad med blanka tecken sker till denna bredd om så behövs.
- Formatsträngen har en formatspecificerare och ev. ett heltal.
- Exempel på några formatspecificerare
  - d heltal (decimal form)
  - f reella tal med heltalsdel och decimaler (flyttal)
  - c värdet redigeras som ett belopp

Redigering av utskrifter – Exempel 1

```
double pi = Math.PI;

Console.WriteLine("{0:f4}", pi);
```

index



Formatspecificerare
F = flyttal (reell tal)
4 = 4 decimaler

Redigering av utskrifter – Exempel 2

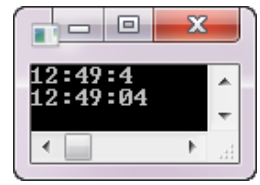
```
double pris = 1999.90;

Console.WriteLine("{0:c2}", pris);
```

index

Formatspecificerare
C = Visa som ett belopp
(beror på nationella inställningar)
2 = 2 decimaler

Redigering av utskrifter – Exempel 3

```
DateTime dt = DateTime.Now;
int tim = dt.Hour;
int min = dt.Minute;
int sek = dt.Second;

string tid = tim + ":" + min + ":" + sek;

// Visa klockan utan formatering
Console.WriteLine(tid);
```



index

```
// Formatera
tid = string.Format("{0:d2}:{1:d2}:{2:d2}", tim, min, sek);

// Visa klockan efter formatering
Console.WriteLine(tid);
```

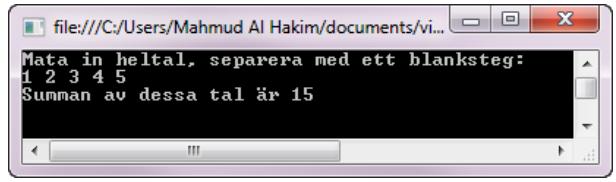
Formatspecificerare
d = heltal
2 = antal siffror
ev. utfyllnad med nollor

Uppdelning av text

- I klassen String finns en instansmetod med namnet **Split**.
- Denna delar upp en text i olika delar, s.k. ord.
- Med ord menar man en följd av ett eller flera icke-blanka tecken.
- Metoden Split ger som resultat ett fält, där varje komponent i fältet innehåller ett ord.

Split - Exempel

```
int sum = 0;
Console.WriteLine("Mata in heltal, separera med ett blanksteg: ");
string rad = Console.ReadLine();
string[] a = rad.Split();
foreach (string ord in a)
{
    if (ord.Length > 0)
        sum += int.Parse(ord);
}
Console.WriteLine("Summan av dessa tal är " + sum);
```



Felkontroll med TryParse

- När man läser indata i ett program bör man kontrollera at de är korrekta.
- Metoden **TryParse** finns för alla de enkla standardtyperna och den gör samma sak som metoden Parse.
- TryParse ger ett returvärde som man kan kontrollera.
- TryParse har två parametrar. Den första är texten som skall avkodas och den andra är den variabel det avkodade värdet skall läggas i (en out-parameter).

TryParse - Exempel

```
int heltal = 0;
double flyttal = 0.0;

Console.WriteLine("Mata in valfri tal eller text: ");
string indata = Console.ReadLine();

if (int.TryParse(indata, out heltal))
    Console.WriteLine("Du har matat in ett heltal: " + heltal);

else if (double.TryParse(indata, out flyttal))
    Console.WriteLine("Du har matat in ett flyttal: " + flyttal);

else
    Console.WriteLine("Du har matat in text: " + indata);
```

Textfiler

- För att läsa eller skriva en fil kan man i C# koppla en **ström** till filen.
- Det enklaste är att använda klasserna **StreamReader** och **StreamWriter** som finns i namnrymden **System.IO**
- När man skapar strömmen ger man filens namn som parameter till konstruktorn.

StreamWriter - Exempel

```
1  using System;
2  using System.IO;
3
4  class Program
5  {
6      static void Main(string[] args)
7      {
8          // Skapa en textfil
9          StreamWriter utFil = new StreamWriter("utdata.txt");
10
11         // Skicka 100 rader till filen
12         for (int i = 1; i <= 100; i++)
13         {
14             utFil.WriteLine(i);
15         }
16
17         // Stäng filen (viktigt)
18         utFil.Close();
19     }
20 }
```

StreamReader- Exempel

```
static void Main(string[] args)
{
    // Öppna filen utdata.txt
    StreamReader inFil = new StreamReader("utdata.txt");

    int antalRader = 0;

    // Läs in en rad i taget
    while (true)
    {
        string rad = inFil.ReadLine();

        if (rad == null) // När filen är slut avsluta while
            break;

        antalRader++;
    }

    Console.WriteLine("Antal rader: " + antalRader);
    Console.ReadKey();
}
```