UPPSALA
UNIVERSITET

. NET-programmering

Klasser

UPPSALA
UNIVERSITET

Klass och Strukt

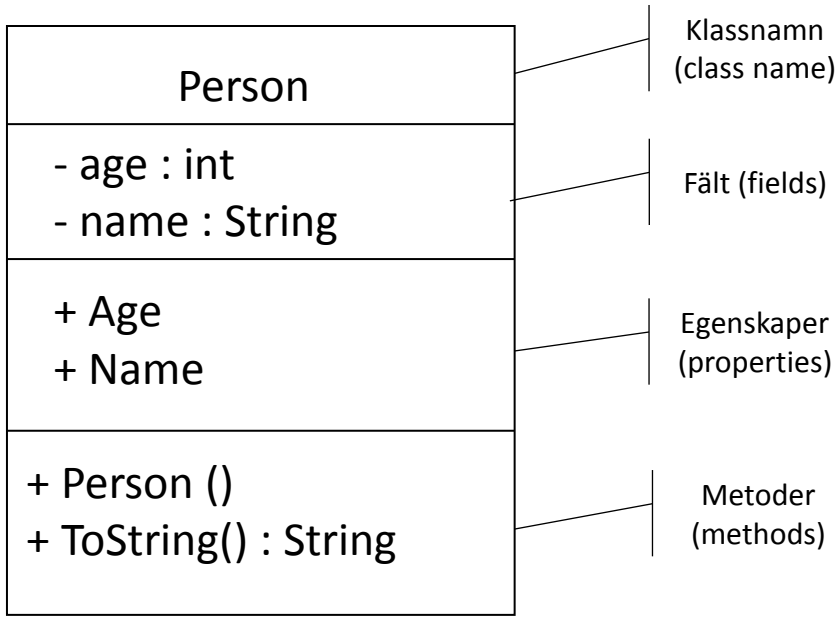
- Fält, konstanter
- Metoder
- Konstruktorer, destruktorer
- Egenskapare
- Indexerare
- Händelser
- Överlagrade operatorer
- Nästlade typer

Klasser

- Nytt objekt skapas med hjälp av new ()
- Standardkonstruktor
- Flera konstruktorer kan överlagras
- Objekt som inte används längre tas bort av Garbage Collectorn

UPPSALA UNIVERSITET

Klass i C#




```
class Person
{
   private int age;
   private String name;

   public Person()
   {
   }

   public String ToString()
   {
       return ”Namn: ” + Name +
              ”, Ålder: ” + Age;
   }

   public int Age
   {
      get{return age;}
      set {age = value;}
   }

   public String Name
   {
      get{return name;}
      set {name= value;}
   }
}
```

Synlighet

- **public**
  - synlig där deklarerande namnutrymme är känt
- **private**
  - synlig i deklarerande klass
- **internal**
  - synlig i deklarerande assembly
- **protected**
  - synlig i deklarerande klass och dess underklasser
- **protected internal**
  - synlig i deklarerande assembly samt i underklasser till deklarerande klass som ligger i andra assemblies

UPPSALA UNIVERSITET

Exempel

```
class A
{
    private int x;
}


class B
{
   private int y;

   public void F(B b)
   {
      y = 2;
      b.y = 4;
      A a = new A();
      a.x = 6;
   }
}
```

OK

Inte tillåtet då denna variabel är privat i en annan klass

Java – Accessors / Mutators

```
public class Person
{
    private int mAge;

    //accessor eller "getter"
    public int getAge()
    {
        return mAge;
    }

    // mutator eller "setter"
    public void setAge(int age)
    {
        mAge = age;
    }
}
```

UPPSALA
UNIVERSITET

C# - Properties

```
class Person
{
    private int age;

    public int Age
    {
        get
        {
            return age;
        }
        set
        {
            age = value;
        }
    }
}
```

UPPSALA
UNIVERSITET

C# - Properties

```
Person p = new Person();
p.Age = 42;

System.Console.WriteLine(p.Age);
```

UPPSALA UNIVERSITET

C# - Properties

Följande är identiskt med förra exemplet:

```
class Person
{
  public int Age { get; set; }
}
```

Men varför krångla med properties när en publik medlemsvariabel hade gjort samma sak?

UPPSALA
UNIVERSITET

C# - Properties

```
class Timer
{
    private int _seconds;

    public int Hours
    {
        get
        {
            return _seconds/3600;
        }
        set
        {
            _seconds = value < 24 ? value * 3600 : _seconds;
        }
    }
}
```

UPPSALA
UNIVERSITET

C# - Object initializer

```
class Cat
{
   public int Age { get; set; }
   public string Name { get; set; }
}

-

Cat cat =
   new Cat(){Age = 10, Name = "Fluffy"};
```

UPPSALA UNIVERSITET

Fält och konstanter

- Kan vara variabler eller konstanter
- För variabler: Initiering vid deklaration går att göra i klass
  I strukt görs initieringen i en konstruktor
- Konstanter måste initieras vid deklarationen (även i struktar)

```
const int MAX_USERS = 10;
```

- Readonly-fält är en konstant som initieras vid deklaration eller i konstruktor. Därefter kan det bara läsas. Initiering under runtime ger möjlighet att sätta värdet dynamiskt.

```
readonly string adminEmail;
```

UPPSALA UNIVERSITET

Statiska fält och konstanter

- Statiska fält och konstanter tillhör klassen, inte enskilda objekt

```
class Rectangle
{
    static Color defaultColor;      // En gång per klass
    static readonly int scale;      // En gång per klass
    int x, y, width, height;        // En gång per objekt
}
```

- Åtkomst från andra klasser genom klassnamnet

```
int rectangleScale = Rectangle.scale;
```

- Endast fält kan vara statiska, inte konstanter

UPPSALA
UNIVERSITET

Arv

```
class Banana : Fruit
{
   public Banana(int size) : base(size)
   {
      //...
   }
}
```