

C# Advanced Inheritance

Koen Bloemen



**DE HOGESCHOOL
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, www.pxl.be

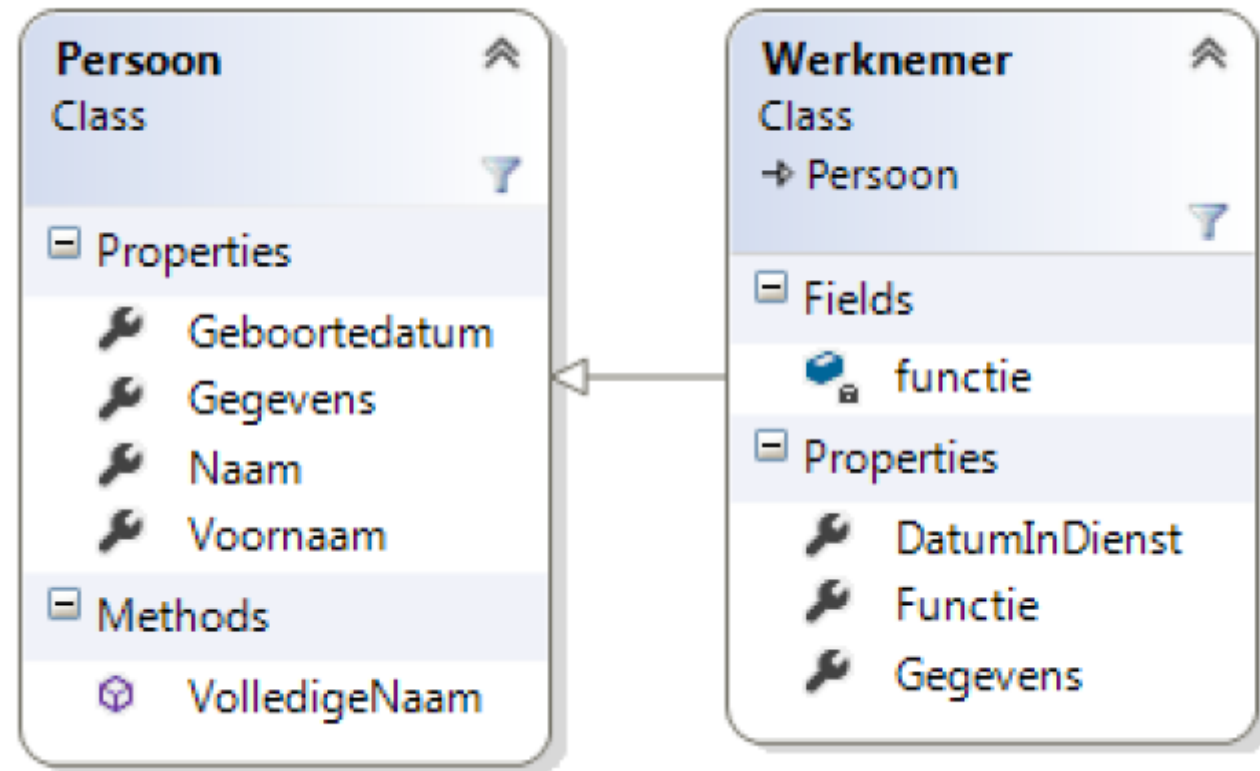




Inheritance
Sleutelwoord base
Abstract class

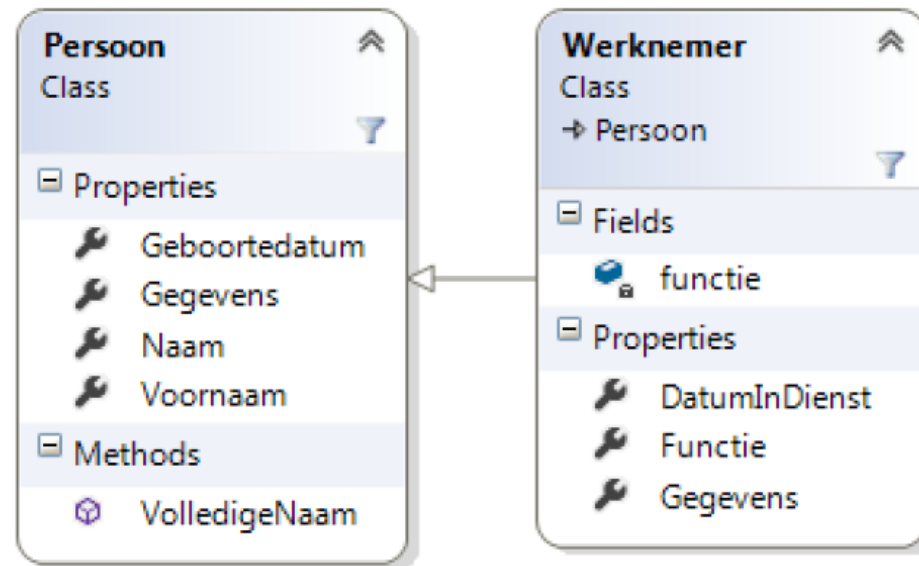
Inheritance

- Nieuwe classes maken op basis van bestaande **base class**
- Nieuwe class kan van de bestaande base class volgende overerven:
 - Membervariabelen
 - Properties
 - Events
 - Methods



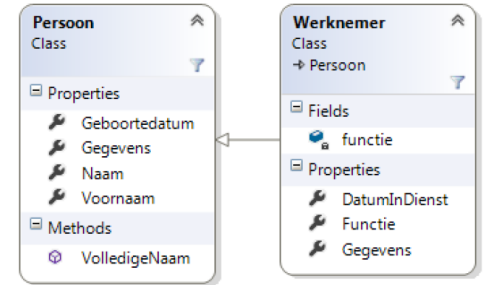
Inheritance

- Methods en properties van base class kunnen ook overschreven worden in de subclass
 - Maak de method/property in de base class virtual. Zet er een standaard implementatie in.
 - Maak de method/property in de subclass override. Zet er een implementatie in die de implementatie van de base class overschrijft.



Inheritance

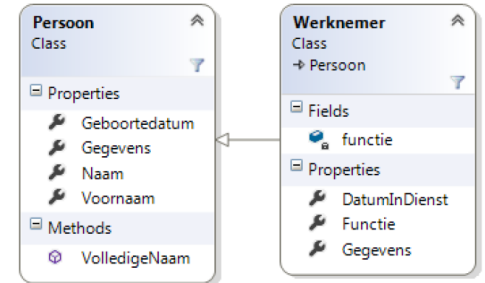
- Base class Persoon (Persoon.cs)
 - Onze basisklasse waarvan we andere classes kunnen afleiden.



```
public class Persoon
{
    public string Voornaam {get; set;}
    public string Naam {get; set;}
    public DateTime Geboortedatum {get; set;}
    // Virtual method/property staat in de base class
    // Je kan deze overschrijven (override) in de afgeleide class en
    // een eigen implementatie daarvoor voorzien.
    public virtual string VolledigeNaam() => $"{Voornaam} {Naam}";
    public virtual string Gegevens =>
        $"{Voornaam} {Naam}\r\n{Geboortedatum.ToLongDateString()}";
}
```

Inheritance

- Afgeleide class Werknemer (Werknemer.cs):
 - Schrijf achter class Medewerker dan : Persoon
 - De dubbele punt betekent: erft over van
 - Een class kan enkel overerven van 1 base class



```

public class Werknemer : Persoon
{
    public DateTime DatumInDienst { get; set; }
    public string Functie { get; set; } = "Lector"; // automatisch initialiseren
    // Override (overschrijf) de implementatie van Gegevens van de base class
    Persoon.
    // Een Medewerker object heeft nu zijn eigen implementatie van Gegevens.
    // base.Gegevens: roept de property Gegevens aan van de base class
    // Dus base.Gegevens is de implementatie die in de class Persoon staat
    // Hierachter vullen we dan aan met de Functie.
    public override string Gegevens => $"{base.Gegevens} - {Functie}";
}
  
```

Sleutelwoord base

- Gebruik sleutelwoord base als je binnen een afgeleide klasse de implementatie van een eigenschap of methode van de basisklasse wil gebruiken

- Base class is Persoon

```
public virtual string Gegevens => $"{Voornaam}{Naam}\r\n{Geboortedatum.ToLongDateString()}";
```

- Afgeleide class is Werknemer

```
public override string Gegevens => $"{base.Gegevens} - {Functie}";
```

- Opmerking:
 - Stel dat Werknemer geen override deed bij een method, dan erfde Werknemer de implementatie van die method mee over
⇒ dan was de implementatie van deze method dezelfde als die van Persoon

Inheritance

Gebruik van base class Persoon (hier in een Console toepassing Program.cs)

```
using System;
using System.Text;
namespace OverervingTest
{
    class Program
    {
        static void Main(string[] args)
        {
            Persoon persoon = new Persoon();
            persoon.Naam = "Doe";
            persoon.Voornaam = "John";
            persoon.Geboortedatum = DateTime.Parse("12/01/1992");
            Console.WriteLine(persoon.VolledigeNaam());
            Console.WriteLine(persoon.Gegevens);
            Console.ReadLine(); // console open laten tot je op een toets drukt
        }
    }
}
```


Inheritance

Gebruik van afgeleide class Werknemer (hier in een Console toepassing Program.cs)

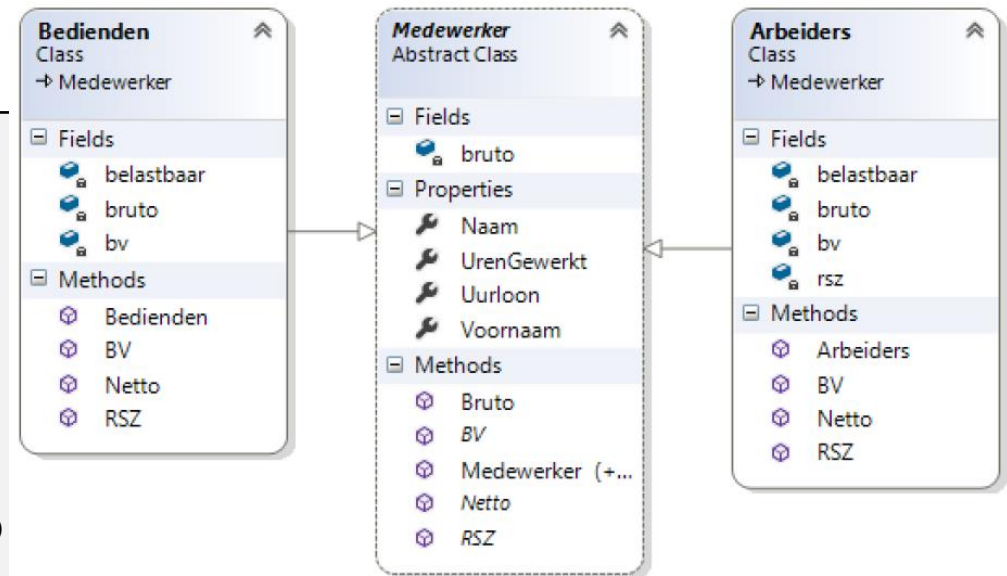
```
using System;
using System.Text;
namespace OverervingTest
{
    class Program
    {
        static void Main(string[] args)
        {
            Werknemer werknemer = new Werknemer();
            werknemer.Naam = "De Bolle";
            werknemer.Voornaam = "Octaaf";
            werknemer.Functie = "Kruidenier";
            werknemer.Geboortedatum = DateTime.Parse("18/05/1950");
            werknemer.DatumInDienst = DateTime.Parse("28/12/1980");
            Console.WriteLine(werknemer.VolledigeNaam());
            Console.WriteLine(werknemer.Gegevens);
            Console.ReadLine(); // console open laten tot je op een toets drukt
        }
    }
}
```

Abstract class

- Een abstracte klasse is een base class
- Afgeleide klassen moeten de basis overerven
- Definiëren het concept
- Dynamisch polymorfisme i.p.v. statisch polymorfisme (methods)
- Instance maken van abstracte klasse gaat niet!

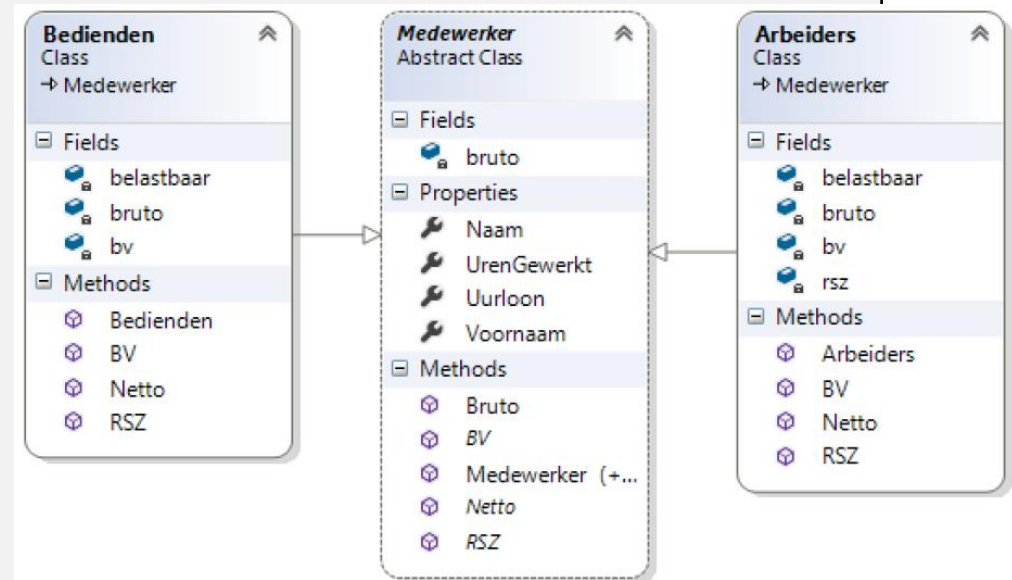
Abstract class

```
public abstract class Medewerker
{
    private double bruto;
    // De klasse Werknemer is een abstracte klasse.
    public string Voornaam { get; set; }
    public string Naam { get; set; }
    public double Uurloon { get; set; }
    public double UrenGewerkt { get; set; }
    public double Bruto() => bruto;
    // Constructor.
    // Bedienden
    public Medewerker(string firstName, string familyName, double salary)
    {
        Voornaam = firstName;
        Naam = familyName;
        bruto = salary;
    }
    // Arbeiders
    public Medewerker(string firstName, string familyName, double hourlyWage, double hours)
    {
        Voornaam = firstName;
        Naam = familyName;
        bruto = hourlyWage * hours;
    }
    // Abstracte methode.
    public abstract double RSZ();
    public abstract double BV();
    public abstract double Netto();
}
```



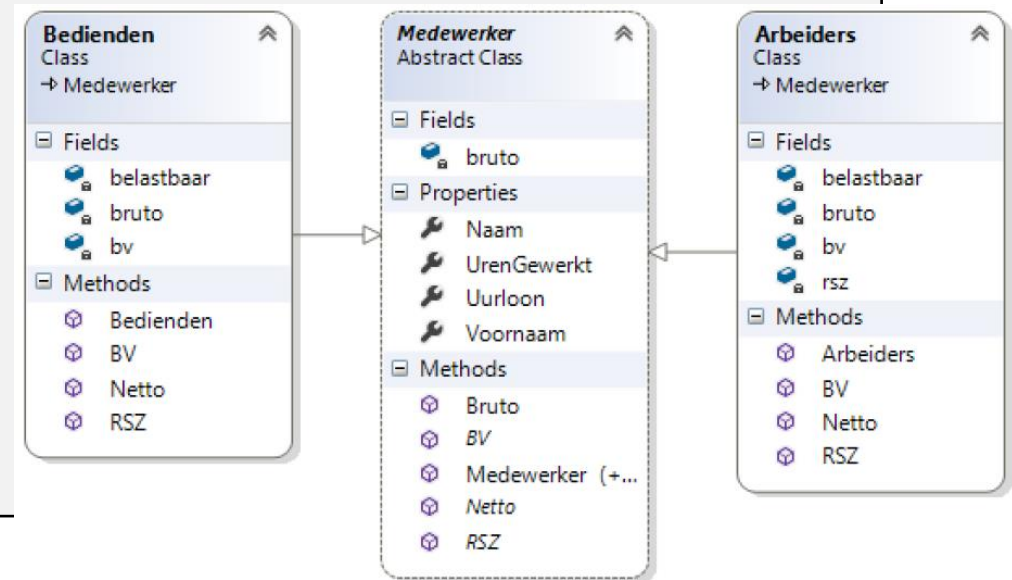
Abstract class

```
public class Arbeiders : Medewerker
{
    private double rsz, belastbaar, bruto, bv;
    public Arbeiders(string firstName, string familyName, double hourlyWage, double hours) :
        base(firstName,familyName,hourlyWage,hours)
    {
        bruto = Bruto();
    }
    public override double BV()
    {
        belastbaar = bruto - RSZ();
        if (belastbaar <= 50000)
        {
            bv= belastbaar * 0.45f;
        }
        else
        {
            bv = (bruto - 50000) * 0.5f + (50000*0.45f);
        }
        return bv;
    }
    public override double Netto() => belastbaar - bv;
    public override double RSZ()
    {
        rsz = bruto * 1.08 * 0.1307;
        return rsz;
    }
}
```



Abstract class

```
public class Bedienden: Medewerker
{
    private double belastbaar, bruto, bv;
    //Roept klasse Werknemer op.
    public Bedienden(string firstName, string familyName, double salary) : base(firstName, familyName,salary)
    {
        bruto = Bruto();
    }
    // De functies moeten overschreven worden.
    public override double RSZ() => bruto * 0.1307f;
    public override double BV()
    {
        belastbaar = bruto - RSZ();
        // Belastingsschijven If-structuur
        if (belastbaar <= 45000)
        {
            bv = belastbaar * 0.45f;
        }
        else
        {
            bv = (bruto - 50000) * 0.5f + (50000 * 0.45f);
        }
        return bv;
    }
    public override double Netto() => belastbaar - bv;
}
```



Abstract class

```
Bedienden med = new Bedienden(txtVoornaamBediende.Text, txtNaamBediende.Text,  
    double.Parse(txtBruto.Text));
```

```
Arbeiders med = new Arbeiders(txtVoornaamArbeider.Text, txtNaamArbeider.Text,  
    double.Parse(txtUurloon.Text), double.Parse(txtAantalUren.Text));
```

