

# C# Advanced Classes

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)



.NET framework

Assembly

OO modelling

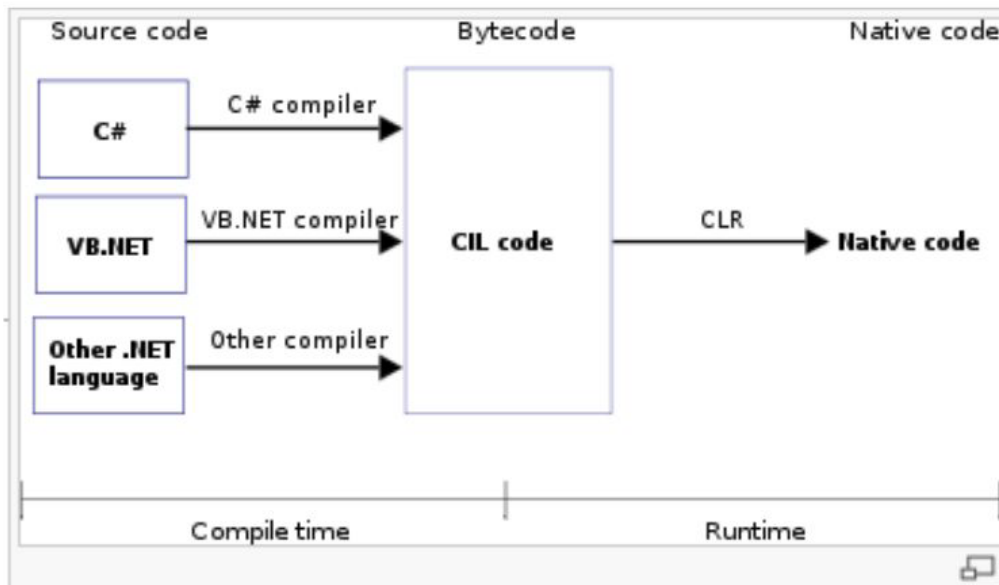
Werken met classes

Properties

ClassLibrary (ClassLib)

# .NET Framework: CLR

- CLR (Common Language Runtime)
  - C#, VB.NET, F#,...  $\Rightarrow$  CIL code
  - CIL (Common Intermediate Language):  
Soort objectgeoriënteerde assembleertaal die mensen nog kunnen lezen.
  - Bij uitvoeren van programma:  
JIT (Just In Time) compiler zet CIL om naar machinetaal (native code).



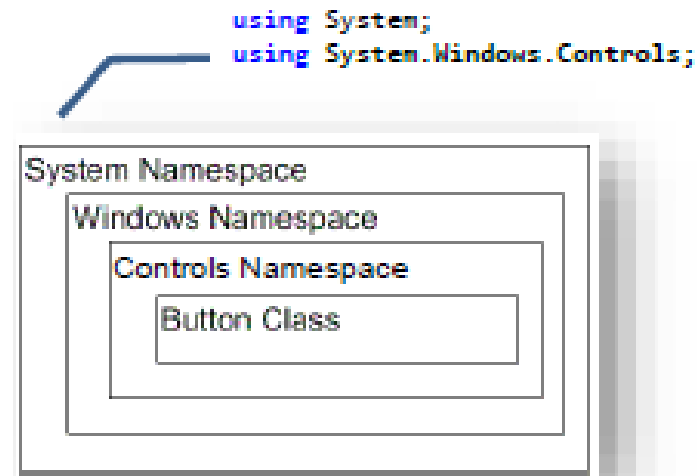
## Waarom CLR als kleine virtuele machine?

- Optimalisaties voor elk systeem
- Geheugenbeheer
- Beveiliging
- Exceptions afhandelen



# Class library

- .NET Framework bevat set van standaard class libraries
  - Base Class Library (BCL)
    - Subset van hele class library
    - Basis van CLR
  - Framework Class Library (FCL)
    - Superset van BCL classes
    - Komt met het .NET Framework



# Assembly

Assembly bevat:

- CIL (gecompileerde programmacode)
- Ondersteunende bestanden
  - Afbeeldingen
  - Tekst
  - ...
- Metagegevens (info over code zelf)
  - Methodes
  - Types
  - Classes
  - ...
- Manifest (info)
  - Versie
  - Bestanden

# Assembly

- Assembly informatie opvragen via Reflection

[https://nl.wikipedia.org/wiki/Reflectie\\_\(informatica\)](https://nl.wikipedia.org/wiki/Reflectie_(informatica))

- Zet bovenaan je bestand

```
using System.Reflection;  
using System.Diagnostics;
```

- Code

```
Assembly assembly = Assembly.GetExecutingAssembly();  
FileVersionInfo fvi = FileVersionInfo.GetVersionInfo(assembly.Location);  
string version = fvi.ProductVersion;  
string productName = fvi.ProductName;  
string companyname = fvi.CompanyName;  
string copyright = fvi.LegalCopyright;  
string description = fvi.Comments;
```

# Object Oriented (OO) modelling

- Voorwaarden
  - Probleem moet OO modelleerbaar zijn
  - Programmeertaal moet OO ondersteunen (C#, Java, C++, ...)
- Basisidee
  - Probleem opdelen in **objecten**
    - Object is een instantie van een **class**

```
Student stud = new Student();
```

      - Eerst een class aanmaken voordat je er een object van kan maken
  - Objecten communiceren door elkaar berichten (messages) te sturen.
    - via **methods**
    - via **properties**
  - Objecten kunnen een toestand hebben
    - opgeslagen in lokale variabelen

# Zelf een eigen class maken

- In VS: rechts klik op project => Add => Class...

```
namespace MijnProgramma
{
    public class Student
    {
        // inhoud van class
    }
}
```

- Class is beschrijving van hoe object
  - eruit ziet (data)
  - zich gedraagt en hoe het communiceert met andere objecten (methods + parameters)



# Membervariabelen

- We zetten hier nu gegevens in (membervariabelen):
  - `private`  $\Rightarrow$  enkel toegankelijk binnen class, niet erbuiten (= encapsulation, inkapseling, data hiding)
  - `public`  $\Rightarrow$  ook toegankelijk buiten class

```
namespace MijnProgramma
{
    public class Student
    {
        private string voornaam; // enkel toegankelijk binnen class
        private string achternaam;
    }
}
```

- Instantie van class maken (object)

```
Student stud = new Student();
```

# Properties en methods

- Objecten communiceren met elkaar via properties en methods

```
namespace MijnProgramma
{
    public class Student
    {
        private string voornaam; // enkel toegankelijk binnen class (private)
        private string achternaam;
        public string Achternaam // ook toegankelijk buiten class (public)
        {
            get { return achternaam; } //getter geeft waarde terug (read), weglaten= write-only
            set { achternaam = value; } //setter past waarde aan (write), weglaten= read-only
        }
        public string ToonVolledigeNaam() // ook toegankelijk buiten class (public)
        {
            return $"{voornaam} {achternaam}";
        }
    }
}
```

# Object aanmaken en gebruiken

- Instantie van class maken (object)

```
Student stud = new Student();
```

- Properties gebruiken:

```
string achternaam = stud.Achternaam;
```

- Methods gebruiken (kan enkel met public methods):

```
string volledigeNaam = stud.ToonVolledigeNaam();
```

- Dit kan niet, want deze variabelen zijn private en niet public

```
string voornaam = stud.voornaam();  
string achternaam = stud.achternaam();
```

# Constructors

- Constructor

- Roep je op bij aanmaken van object
- Constructor retourneert nooit iets!!!  $\Rightarrow$  dient enkel om object te maken
- Constructor heeft altijd dezelfde naam als de class!!!

```
Student stud = new Student(); //default constructor
```

- Default constructor

- Is standaard geïmplementeerd in elke class
- Maar kunnen we zelf eigen code ook in zetten

```
public class Student
{
    public Student() // kan je aanmaken via snippet: ctor
    {
        // eigen code
    }
}
```



# Constructors

- Constructor met parameters

- Als je parameters wil doorgeven aan constructor.
- Wil je default constructor nog gebruiken? ⇒ zelf implementeren!!!

**Gebruik:**

```
Student stud = new Student("Jan", "Das");  
Student stud2 = new Student();
```

```
public class Student  
{  
    private string voornaam;  
    private string naam;  
    public Student(string vn, string n) //Net zoals je parameters doorgeeft aan een method  
    {  
        voornaam = vn;  
        naam = n;  
    }  
    public Student() //Default constructor zelf schrijven!  
    {  
        voornaam = "Tom";  
        naam = "Quareme";  
    }  
}
```

# Properties

```
public class Student
{
    private string achternaam; // (snippet: propfull)
    public string Achternaam // ook toegankelijk buiten class (public)
    {
        get { return achternaam; } // getter geeft waarde terug (read), kan je niet weglaten!
        set { achternaam = value; } // setter past waarde aan (write), weglaten = read-only
    }
    // Auto-implemented property (snippet: prop)
    // Enkel een public property, geen private member variabele die erbij hoort
    public string Voornaam { get; set; }
    // Expression-bodied property (sinds C# 6.0)
    // Kortere manier om read-only properties te schrijven (waar enkel get en return in komen)
    public string Gegevens => $"{Voornaam} {Achternaam.ToUpper()}"
}
```

- Je kan get of set ook telkens weglaten
  - Enkel get  $\Rightarrow$  read-only property
  - Enkel set  $\Rightarrow$  write-only property

# Properties

- Auto-implemented properties met private set

```
public class Student
{
    // Auto-implemented property (snippet: prop)
    // Enkel een public property, geen private member variabele die erbij hoort
    // De "private set" zorgt ervoor dat je enkel binnen de class kan aanpassen,
    // maar niet erbuiten!
    public string Voornaam { get; private set; }
    public Student()
    {
        Voornaam = "John"; // Binnen de class kan je wel aanpassen, maar daarbuiten niet!
    }
}
```

- Wanneer je set wil kunnen doen in de klasse zelf, maar niet erbuiten  
Dit gaat NIET:

```
Student stud = new Student();
stud.Voornaam = "Pete"; // geeft een fout...
```

# ClassLib(rary)






Een ClassLib laat toe de klasse(n) te herbruiken in meerdere projecten

- Stap 1: Maak een nieuw project BewerkingsLib (gebruik de template Class Library)
- Stap 2: Wijzig de namespace en class name

```
namespace ClassLibraryBewerking
{
    public class Bewerking
    {
        public float Som(float x, float y)
        {
            return x + y;
        }
        public float Min(float x, float y)
        {
            return x - y;
        }
        public float Maal(float x, float y)
        {
            return x * y;
        }
    }
}
```

## Create a new project

Recent project templates

	WPF App (.NET Framework)	C#
	Class Library (.NET Framework)	C#
	Class library	C#
	WPF Application	C#
	Console Application	C#



# ClassLib(rary)

- Stap 3: Wijzig de solution configuration van Debug naar Release
- Stap 4: Build de solution
- Resultaat: Je vindt nu een DLL terug onder de map bin/release

## ClassLib gebruiken

- Stap 1: Maak een nieuw WPF (of console) project
- Stap 2: Voeg de reference toe naar je zelf gemaakte ClassLib
- Stap 3: Voeg een using statement toe naar **ClassLibraryBewerking**
- Stap 4: Maak gebruik van de class

```
Bewerking bw = new Bewerking();  
TxtResultaat.Text = $"{bw.Som(float.Parse(TxtGetal1.Text), float.Parse(TxtGetal2.Text))}";
```