C# Advanced
Classes deel 2

Koen Bloemen
Sander De Puydt
Erdem Yaraci

DE HOGESCHOOL MET HET NETWERK

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

Classes deel 2

- Partial
- Constructor Chaining
- Static methods
- Access modifiers
- Objecten vernietigen
- Class Library (herhaling)
- Samenvatting

Partial

- Sommige klasses bestaan gedeeltelijk uit gegenereerde code, zoals de WPF Windows. Deze klasses bestaan uit meerdere bestanden en noemen **partial** klasses.
  - Partial klasses worden gebruikt om menselijk geschreven code te onderscheiden van gegenereerde code.
- Partial klasses kunnen ook gebruikt worden om helpfuncties in te definiëren.

Partial

- Het partial keyword komt na de access modifier.

```
namespace Klasses3
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
                ...
```

Partial

- Voorbeeld van opsplitsen van helpfuncties:

```
public partial class Verhuizing
{
    public DateTime Dag { get; set; }

    private double prijs;
    public double Prijs
    {
        get { return prijs; }
    }

    public List<string> Meubels { get; set; }
    public int AantalVrachtwagens { get; set; }
}
```

Partial

- Voorbeeld van opsplitsen van helpfuncties:

```
public partial class Verhuizing
{
    private double prijsPerMeubelStuk = 20;
    private double prijsPerVrachtWagen = 300;

    public void BerekenPrijs()
    {
        this.prijs = Meubels.Count * 20 +
        AantalVrachtwagens * prijsPerVrachtWagen;
    }
}
```

Constructor Chaining

- Wanneer er verschillende manieren zijn om een object aan te maken van een klasse (= meerdere constructors), dan kan het zijn dat de code in constructors duplicaat is.

  Je kan dit echter vermijden door **constructor chaining**.

Constructor Chaining

- Bijvoorbeeld:

```
public class Boek
{
    private string auteur;
    private string titel;
    private string uitgeverij;
    private int editie;
    private int aantalPaginas;

    ...
}
```

Constructor Chaining

- Constructors met duplicate code:

```
public class Boek {
    ...
    public Boek(string auteur, string titel)
    {
        this.auteur = auteur;
        this.titel = titel;
    }

    public Boek(string auteur, string titel, string uitgeverij,
                int editie) {
        this.auteur = auteur;
        this.titel = titel;
        this.uitgeverij = uitgeverij;
        this.editie = editie;
    }
}
```

Constructor Chaining

- Oplossing: Constructor Chaining

```
public class Boek
{
    ...
    public Boek(string auteur, string titel)
    {
        this.auteur = auteur;
        this.titel = titel;
    }

    public Boek(string auteur, string titel, string uitgeverij,
                int editie) : this(auteur, titel)
    {
        this.uitgeverij = uitgeverij;
        this.editie = editie;
    }
}
```

Constructor Chaining

- Constructor Chaining voert eerst de ge-chainde constructor uit en werkt op hetzelfde object.
- Kan ook gebruikt worden om default constructor te initiëren met waardes.

```
public Boek() : this("JK Rowling",
                     "Harry Potter en de gevangenen van Azkaban")
{ }

public Boek(string auteur, string titel)
{
    this.auteur = auteur;
    this.titel = titel;
}
```

Static methods

- Method die op klasseniveau wordt gebruikt en niet op een object
- Kan NIET gebruik maken van membervariabelen of properties, want die horen bij een object. We kunnen enkel lokale variabelen of static variabelen gebruiken hierin.
- Voorbeelden: `Math.Sqrt()`, `Convert.ToString()`, `Console.WriteLine()`, `String.Format()`
- Zelf een static method maken

```
public class Student
{
   public static Student MaakNieuweStudent()
   {
      return new Student();
   }
}
```

- Gebruik

```
Student stud = Student.MaakNieuweStudent();
```

Access modifiers

De access modifier bepaalt de toegankelijkheid (van properties, methods, ...)

- public
  - binnen de assembly en de assembly die ernaar verwijst
- private
  - binnen dezelfde class of struct
- protected
  - binnen de klasse of afgeleide klasse (zie volgend hoofdstuk over inheritance)
- internal
  - enkel binnen de assembly
- protected internal
  - binnen dezelfde assembly of binnen een afgeleide klasse

Objecten vernietigen

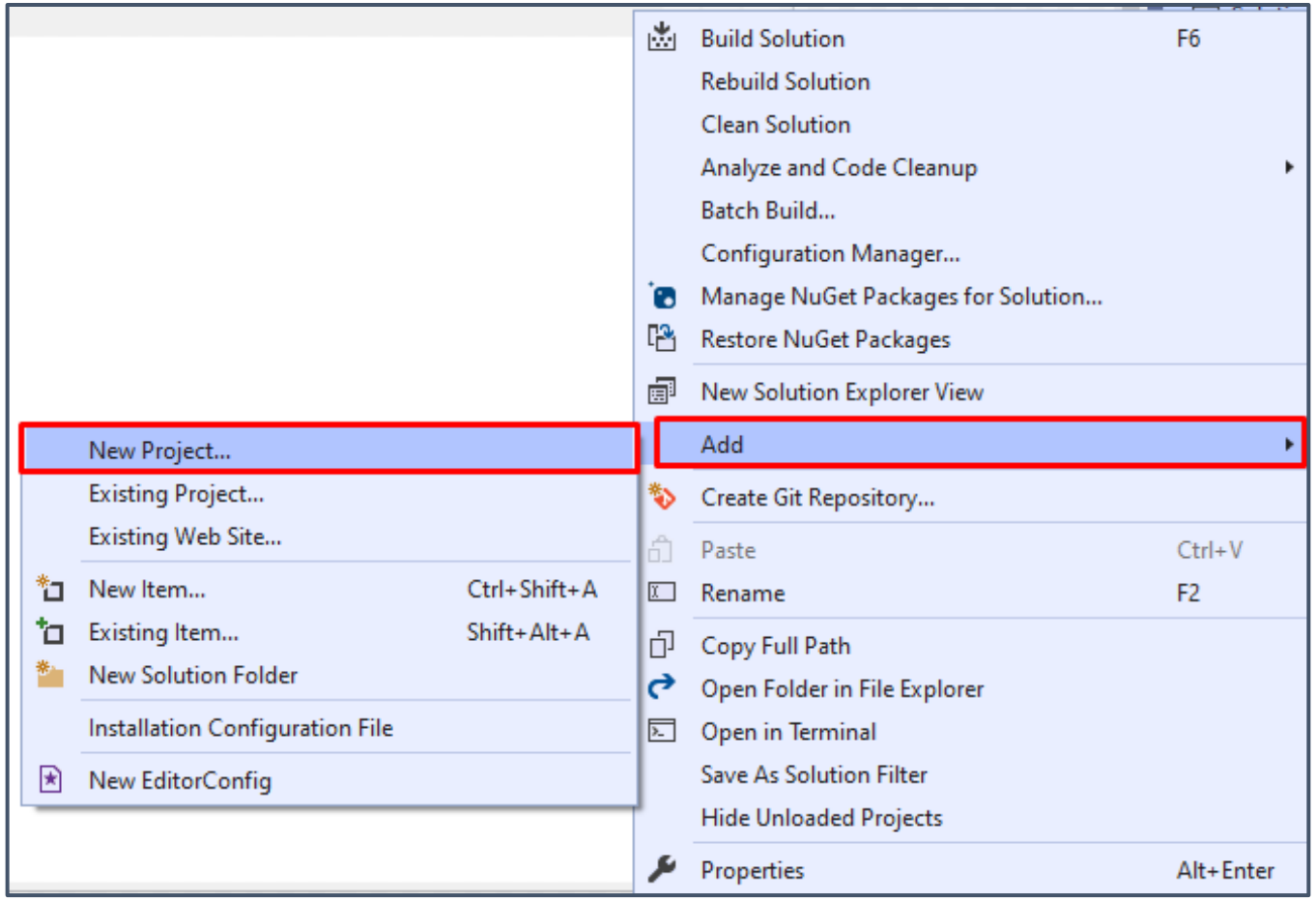
- Wanneer programma stopt
- Als je het object gelijk stelt aan null
  - stud = null;
  - er is dan geen referentie meer naar het bestaand object
  - garbage collector ruimt object op
- Wanneer een object uit scope gaat
  - stel dat het object aangemaakt wordt binnen een method
  - method afgelopen => object bestaat niet meer

Class Library (herhaling)

- Een class library is een project waarin we methodes en klasses definiëren die we kunnen oproepen in andere project**en**.
  - Het helpt het project net te houden en code op een logische manier op te delen.
  - Het helpt om code te hergebruiken binnen verschillende projecten.

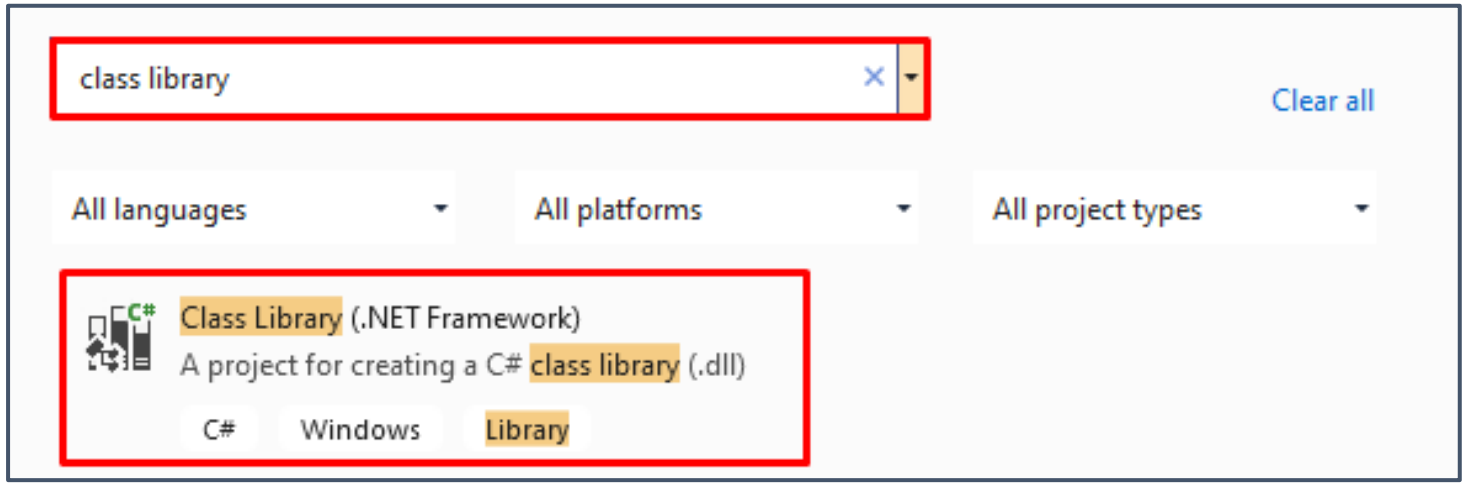
Class Library

- Aanmaken van een class library: RMK Solution



Class Library

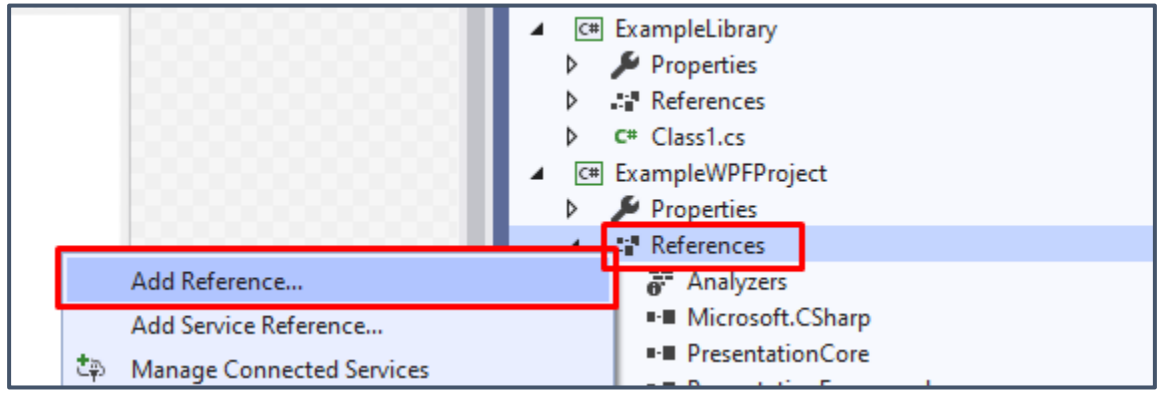
- Aanmaken van een class library:



  - Geef een naam aan het project

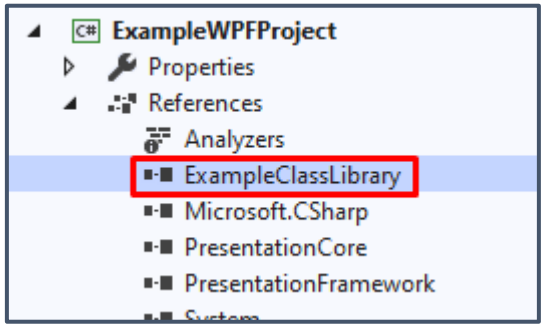
Class Library

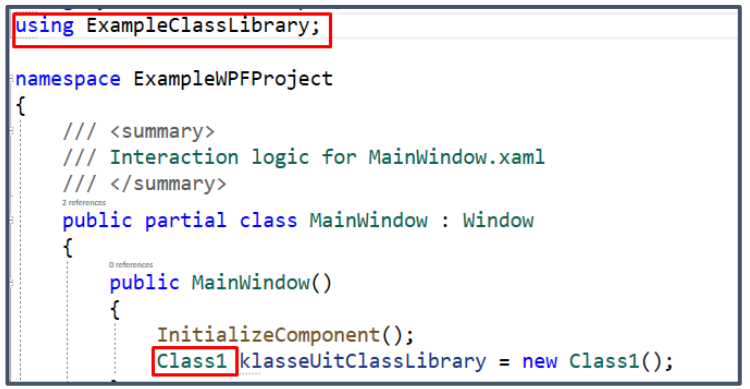
- Eens het project is aangemaakt moeten we het nog gebruiken in een ander project door middel van een reference.
  - RMK op references van project dat de class library wil gebruiken.
  - Add Reference



Class Library

- Vervolgens vind je je eigen projecten terug onder “Projects”.
  - Vink je project aan
- Vervolgens kan je de klasses uit je class library oproepen in een klasse met een using statement:
  - using ExampleClassLibrary;





Samenvatting

- De designer van WPF gebruikt partial klasses
- Vermijd duplicate code in constructoren met constructor chaining.
- Gebruik class libraries om code te structureren en makkelijk te hergebruiken.