

C# Advanced Classes

Koen Bloemen
Sander De Puydt
Erdem Yarici



**DE HOGESCHOOL
MET HET NETWERK**

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



Classes

Introductie

Wat is een Klasse

- Variabelen
- Methode

Wat is een object

- Constructor

Properties

Associaties

Class Designer

Klassendiagram

ClassLib(rary)



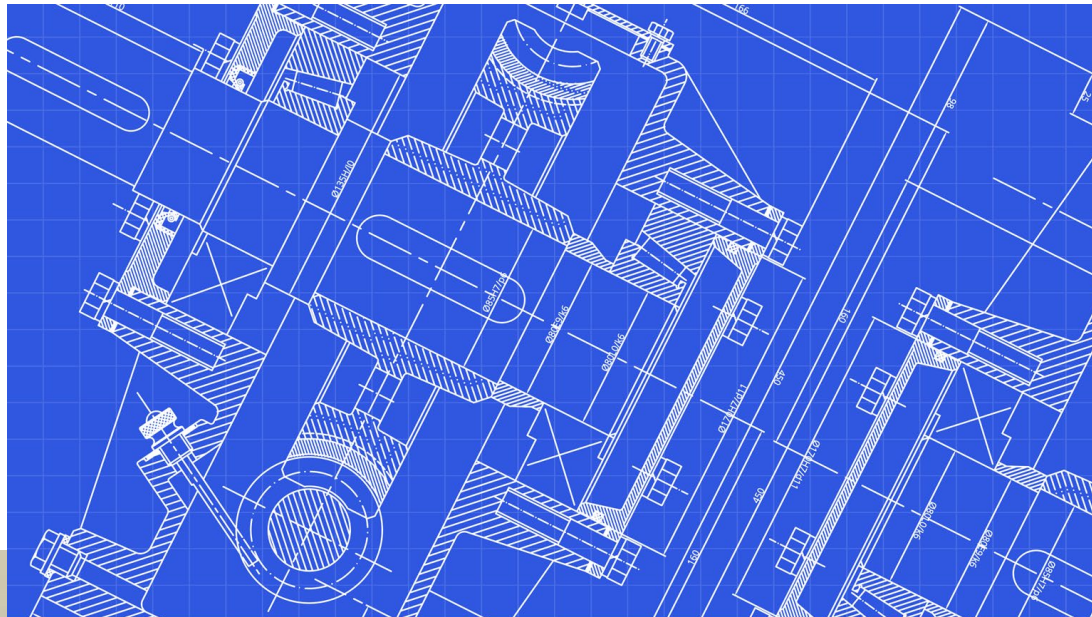
Introductie

- Indien we programma's complexer en groter willen maken, dan gaan we klassen nodig hebben om structuur te brengen in onze logica. Denk aan hoe we meerdere windows kunnen gebruiken om functionaliteiten op te delen onder verschillende schermen.

Klassen doen echter meer dan dat...

Wat is een Klasse

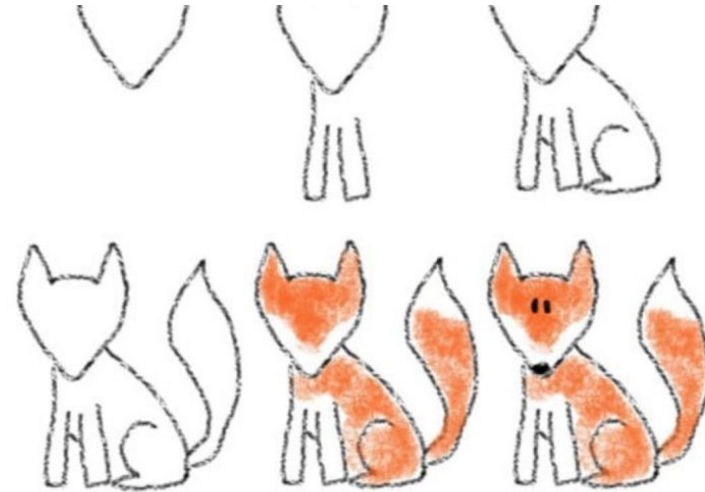
- Een klasse is een **blueprint** waarin we een logica en data in beschrijven.
- Het kan evenzeer uitlijnen hoe een nieuw type data er uit ziet. (DateTime, TimeSpan)



Wat is een Klasse

Opdracht 1: Teken een dier in paint (of beschrijf deze in notepad)

- Welke eigenschappen heeft je dier?
 - Kleur?
 - Aantal poten?



Wat is een Klasse

- Elk dier is een object.
- Alle dieren hebben gemeenschappelijke kenmerken.



Wat is een Klasse

- Dier (object heeft variabelen)

variabelen:

- aantal poten
- soort
- heeft vacht
- legt ei
- grootte
- ...

Variabelen/Eigenschappen zijn zelfstandige
naamwoorden.

= kenmerken

Wat is een Klasse

- Dier (object heeft methodes)

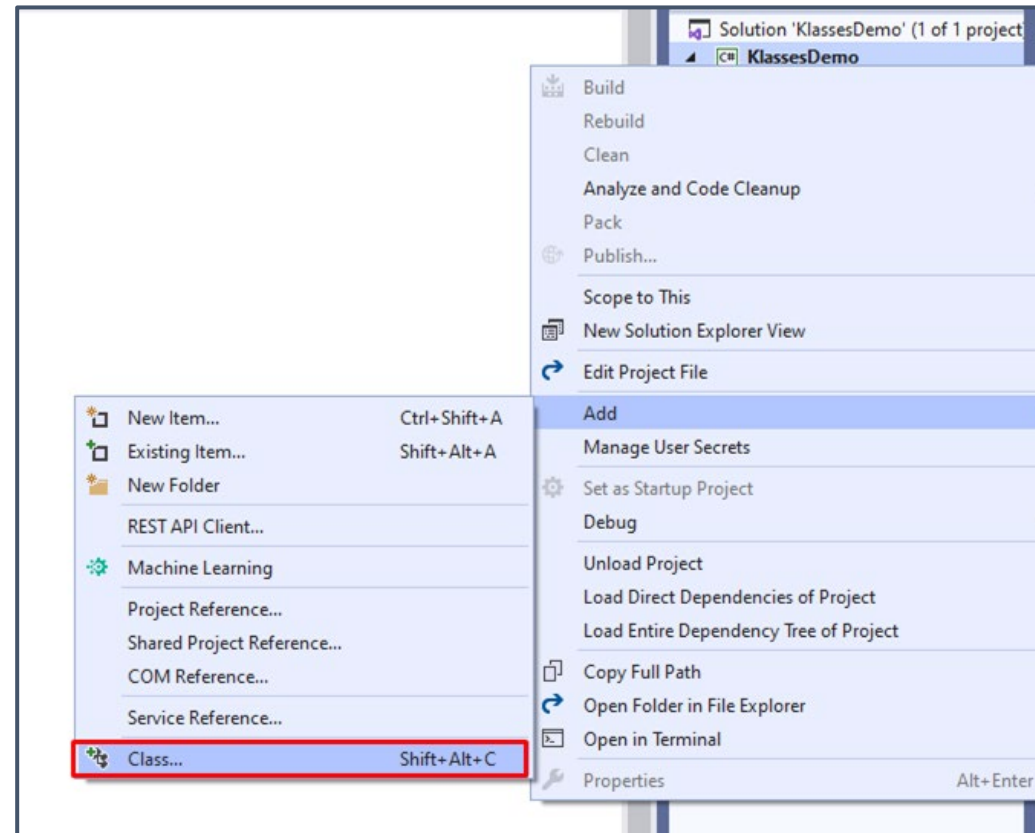
methodes:

- loop
- maak geluid
- eet
- slaap
- ...

Methodes zijn werkwoorden.
= gedrag

Wat is een Klasse

- Een nieuwe klasse toevoegen aan een project.



Wat is een Klasse

- Een klasse wordt als volgt **gedeclareerd** en wordt altijd binnen een namespace geplaatst.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace KlassenDemo
{
    class NieuweKlasse
    {
    }
}
```

Wat is een Klasse - Variabele

- Binnen een klasse kan je waardes definiëren deze hebben een **access modifier** om te bepalen wie deze kan manipuleren (zoals, private en public)

```
namespace KlassenDemo
{
    class NieuweKlasse
    {
        private DateTime momentVanAanmaak;
        public string Beschrijving;
    }
}
```

Wat is een Klasse - Methode

- Een klasse kan ook gedrag definiëren. Dit zijn methodes die je kan uitvoeren op een instantie (of een object van) de klasse.

```
class NieuweKlasse
{
    private string naam;

    private void ZegHallo()
    {
        Console.WriteLine($"Hallo ik ben {naam}");
    }
}
```


Wat is een Klasse - Methode

- Methodes kunnen zowel private als public zijn.
- Wanneer een methode of variabele private is, dan is deze niet bruikbaar door andere classes.

```
class NieuweKlasse
{
    public int BerekenTotaal()
    {
        return 9001;
    }
}
```

Wat is een object

- Eens je de blueprint hebt geschreven, dan kan je er een **afdruk** van maken.
- Deze afdruk wordt een **instantie** van de klasse of een **object** van de klasse genoemd.
- Een object wordt aangemaakt met behulp van een constructor.

Wat is een object - Constructor

- Een **constructor** heeft altijd de naam van de klasse.
- Een constructor heeft geen return type in de signatuur. Het return type is altijd de klasse zelf.

```
class NieuweKlasse
{
    public NieuweKlasse()
    {
    }
}
```

Wat is een object - Constructor



- In de constructor kan je alle waarden van het object initialiseren. Hier beschrijf je alles wat moet gebeuren bij het aanmaken van het object van de klasse.

```
class NieuweKlasse
{
    private int leeftijd;

    public NieuweKlasse()
    {
        leeftijd = 0;
    }
}
```


Wat is een object - Constructor

- Een constructor kan ook argumenten ontvangen, net zoals een functie.

```
class NieuweKlasse
{
    private int leeftijd;

    public NieuweKlasse(int jaren)
    {
        leeftijd = jaren;
    }
}
```

Wat is een object - Constructor



- Een constructor kan je oproepen in een andere klasse.

```
class Program
{
    static void Main(string[] args)
    {
        NieuweKlasse klasse = new NieuweKlasse();
        // klasse.ZegHallo(); deze methode is private en kan
        // niet opgeroepen worden
        Console.WriteLine(klasse.BerekenTotaal());
    }
}
```

Wat is een object - Constructor

- Constructor met parameters **en** default constructor
 - Als je zowel parameters wil doorgeven aan constructor
 - en als je default constructor nog wil gebruiken ⇒ zelf implementeren!!!

```
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 = "Jorg";
        naam = "Bakay";
    }
}
```

Gebruik:

```
Student stud = new Student("Talha", "Van de Parre");
Student stud2 = new Student();
```

Hoe wordt data gecommuniceerd

- Wanneer je informatie uit een klasse wil halen, dan kan je dit rechtstreeks doen met **public variabelen**. (maar dat is niet de juiste aanpak!)

```
class Program
{
    static void Main(string[] args)
    {
        NieuweKlasse klasse = new NieuweKlasse();
        klasse.teller = 0;
    }
}
```

```
public class NieuweKlasse
{
    public int teller;
}
```


Properties

- In plaats van public variabelen gebruik je beter **properties (eigenschappen)**.

```
public class NieuweKlasse
{
    // Met snippet "propfull"
    private int teller;

    public int Teller
    {
        get { return teller; }
        set { teller = value; }
    }
}
```

Properties

- Property: de korte versie

```
public class NieuweKlasse
{
    // Met snippet "prop"
    public int Teller { get; set; }
}
```

Properties

- Een property gebruiken (aanpassen) in een andere klasse.

```
class Program
{
    static void Main(string[] args)
    {
        NieuweKlasse klasse = new NieuweKlasse();
        klasse.Teller = 2;
    }
}
```

Properties

- Met properties kan je **extra logica** toevoegen aan hoe een waarde wordt opgevraagd of gewijzigd.

```
public class NieuweKlasse
{
    private int getTeller;

    public int GetTeller
    {
        get { return getTeller++; }
        // de waarde wordt altijd verhoogd
        // wanneer de variabelen wordt opgevraagd

        // set { getTeller = value; }
        // Er is geen set aanwezig, dus de waarde kan
        // niet in een andere klasse worden aangepast,
        // maar wel worde opgevraagd = readonly
    }
}
```


Associaties

- Wat is het voordeel van OO-programmeren?
 - > hergebruik van bestaande code
 - Als een klasse goed gedefinieerd is kan ze in een andere context hergebruikt worden.
 - Dit doen we door relaties te leggen tussen klassen (en objecten).
 - Eén object gaat zo gebruik kunnen maken van de functionaliteit van een ander object.

Associaties

- Voorbeeld

Een **Persoon** heeft een **Adres**

Een **Persoon** heeft een **GeboorteDatum**

Een **Boek** heeft een **Auteur**

Associaties zijn een “HAS A”-relatie

Associaties

- Voorbeeld met een Auto

```
public class Auto
{
    private string nummerPlaat;

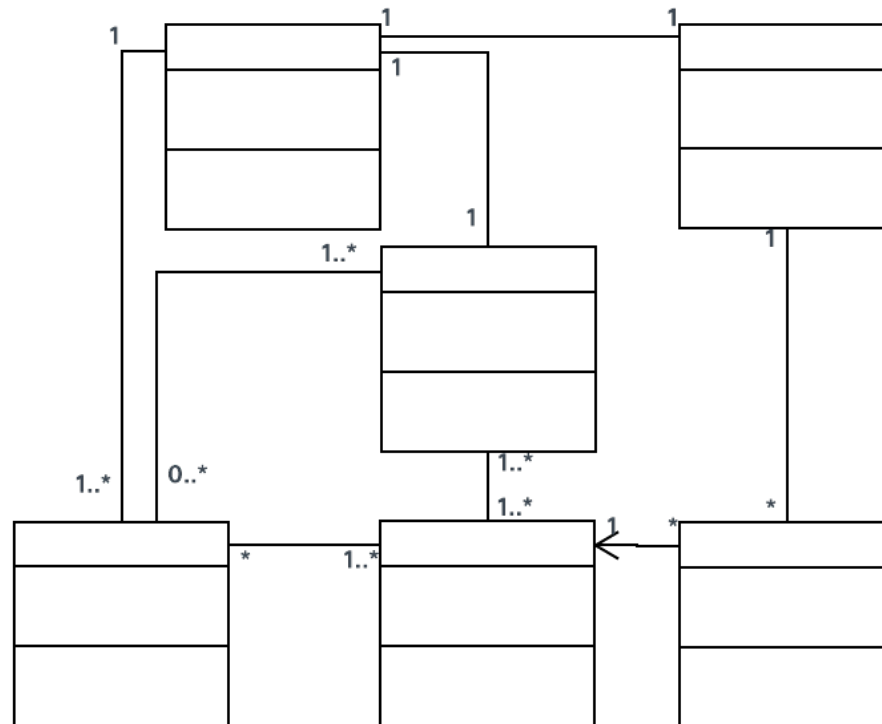
    public Auto(string nummerPlaat)
    {
        this.nummerPlaat = nummerPlaat;
    }
}
```

```
public class Persoon
{
    private Auto mijnWagen;

    public Persoon(Auto mijnWagen)
    {
        this.mijnWagen = mijnWagen;
    }
}
```

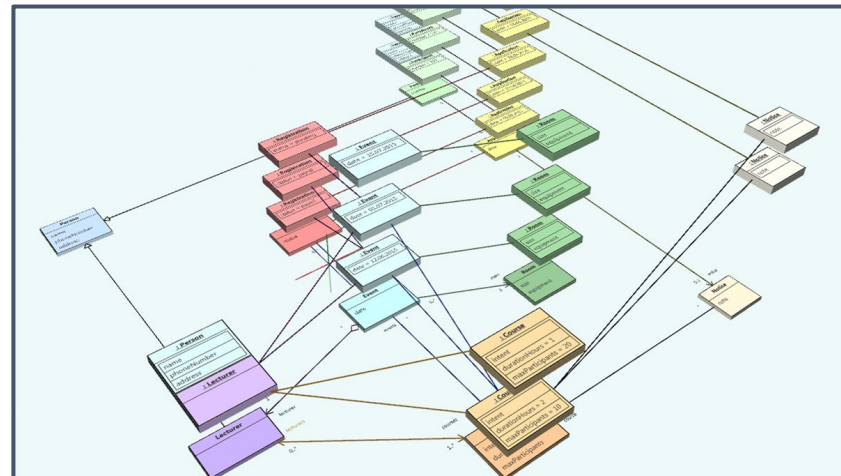
Associaties

- Relaties visualiseren tussen verschillende klassen doe je via een Class Diagram



Class Designer

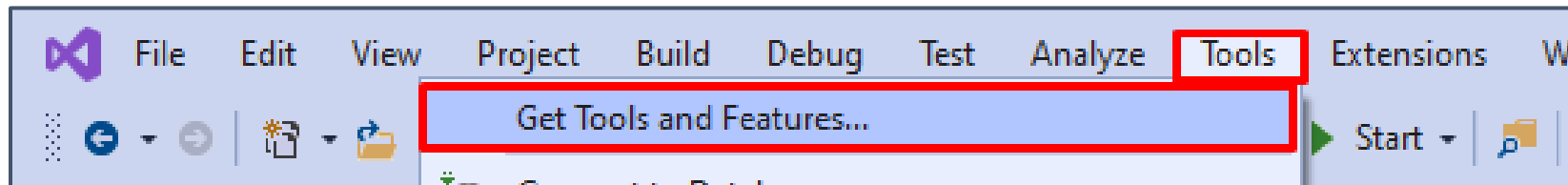
- Wanneer je een overzicht wil maken van een klasse, dan gebruik je de Class Designer.
- De Class Designer maakt Klasse diagrammen.
- Klasse diagrammen worden gebruikt om de inhoud van een klasse te visualiseren.



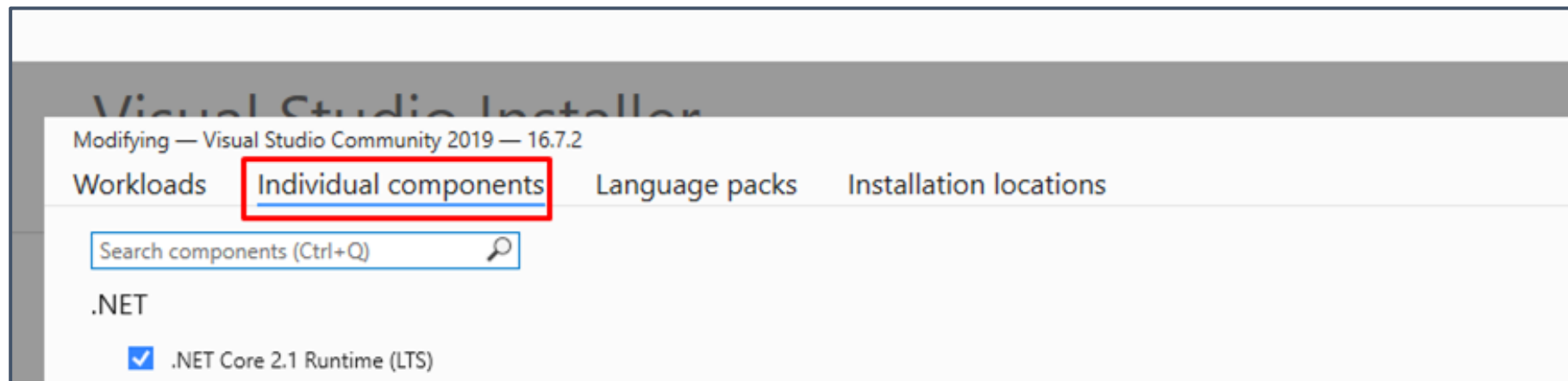
Class Designer

Hoe installeer je de class designer?

- Tools > Get Tools and Features



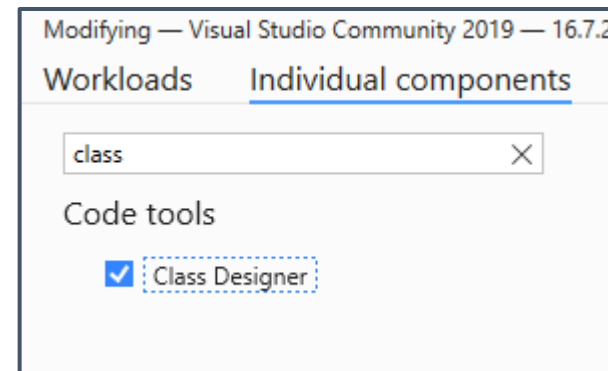
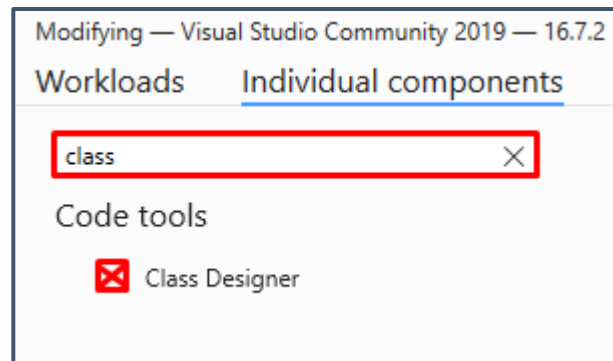
- Individual Components



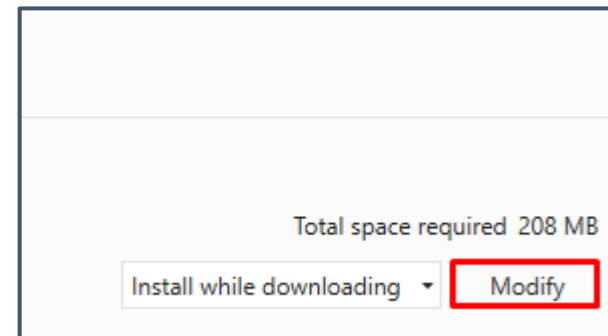
Class Designer

Hoe installeer je de class designer?

- Vink de Class Designer aan



- Klik op modify



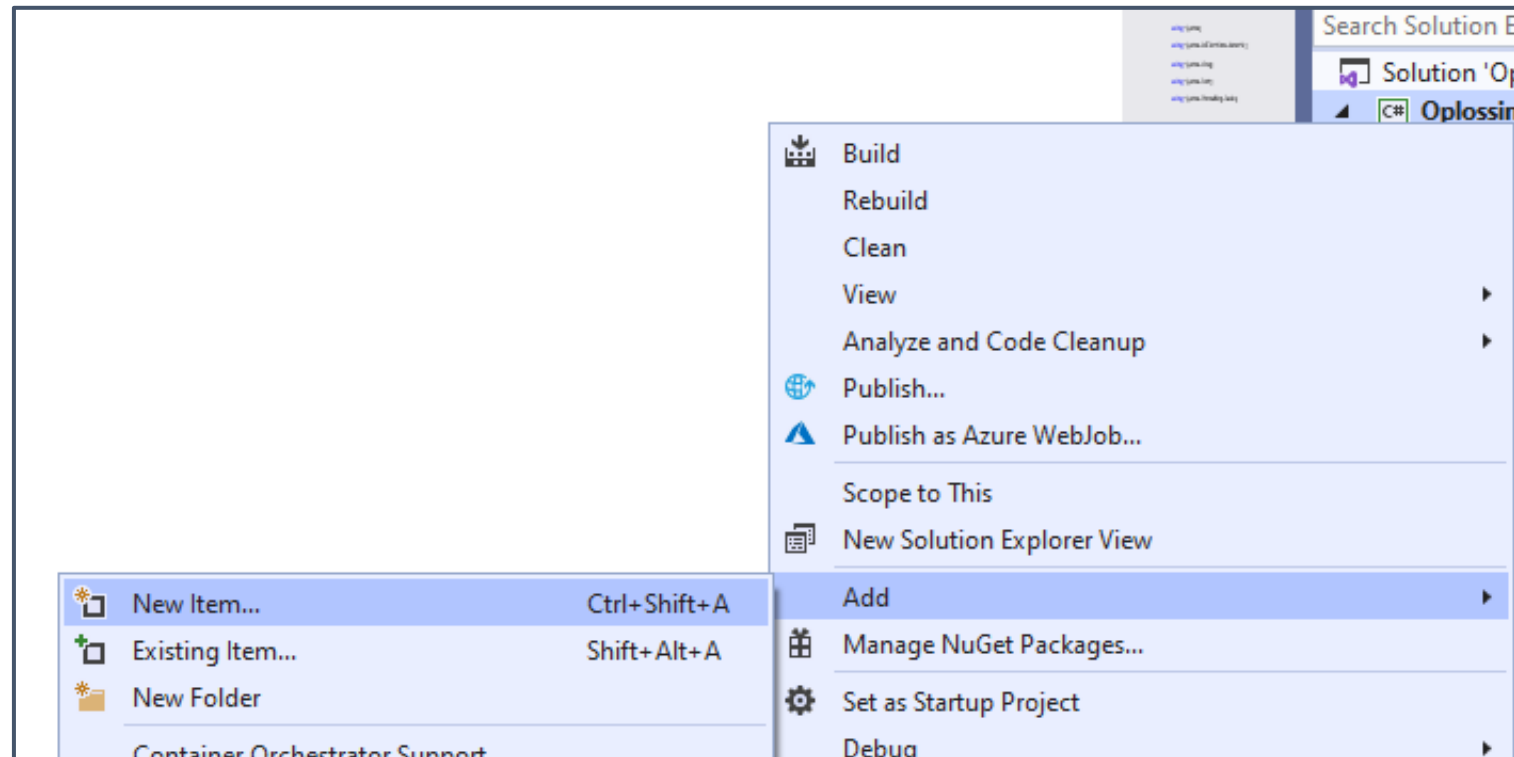
Klassendiagram

- In een klassendiagram heb je een overzicht van welke variabelen er in een klasse bestaan en welke methodes er zijn.
- Het geeft een overzicht van de eigenschappen en relaties van het systeem.

Klassendiagram

Hoe maken we een klassendiagram?

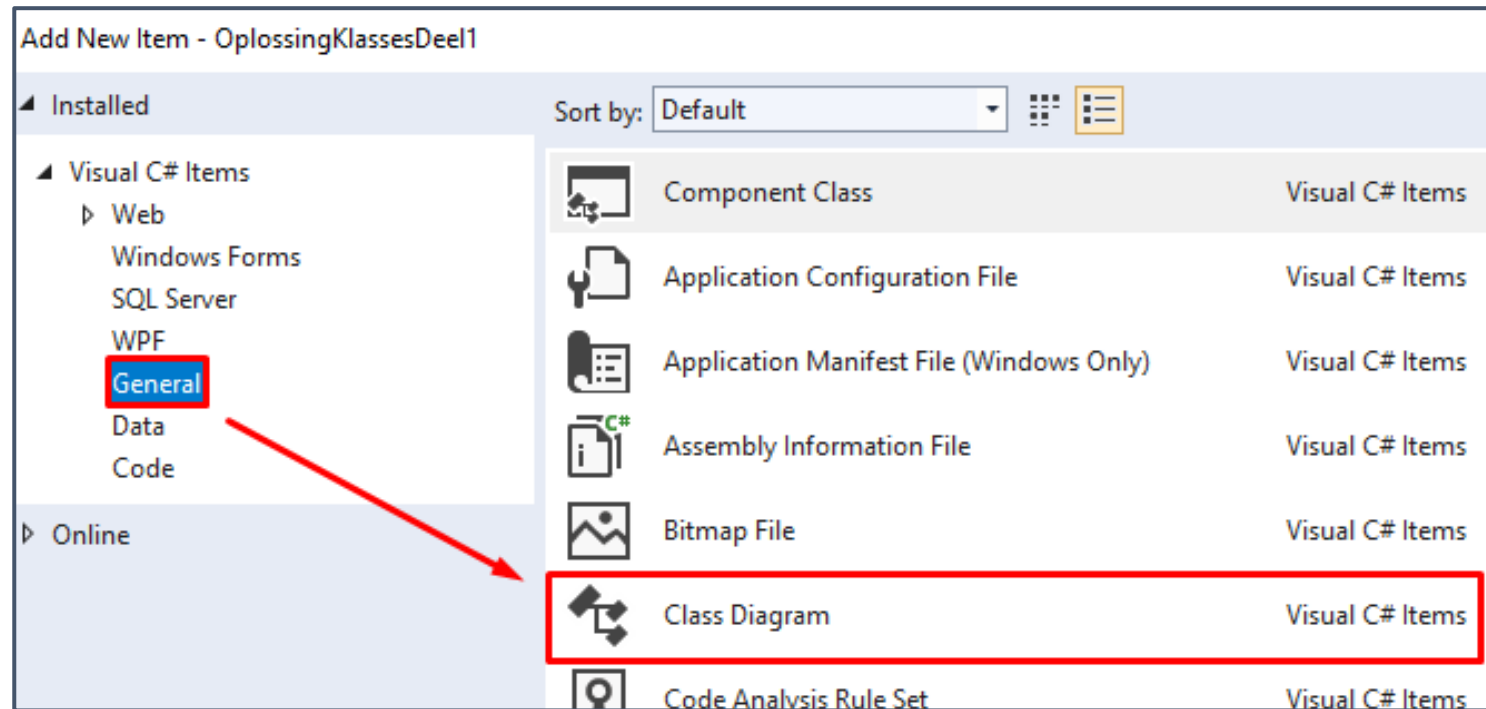
- In de Solution Explorer > Add > New Item...



Klassendiagram

Hoe maken we een klassendiagram?

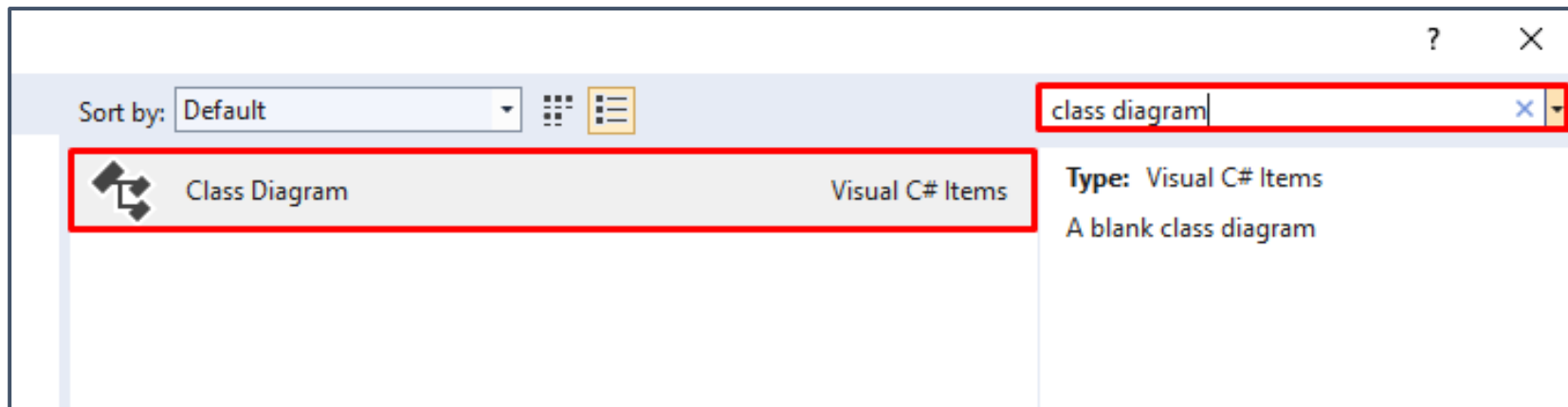
- In General > Class Diagram



Klassendiagram

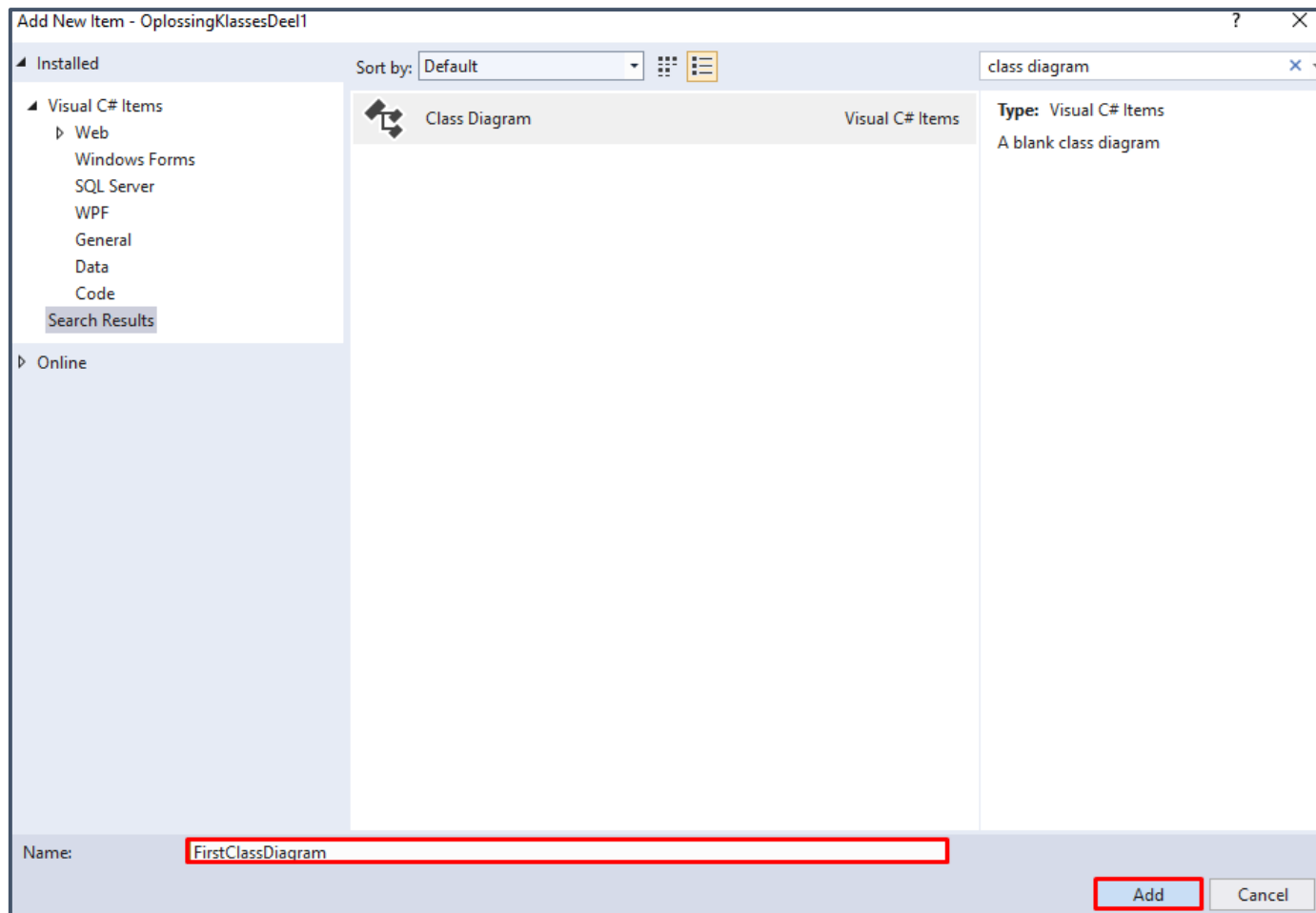
Hoe maken we een klassendiagram?

- Zoeken op “Class Diagram”



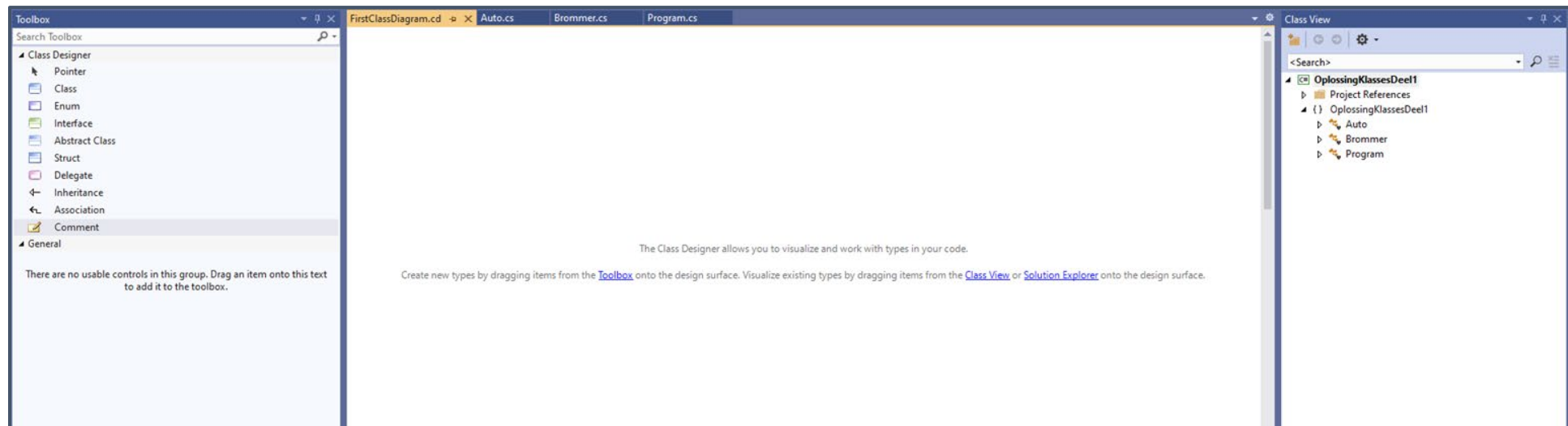
Klassendiagram

- Geef een naam en Add



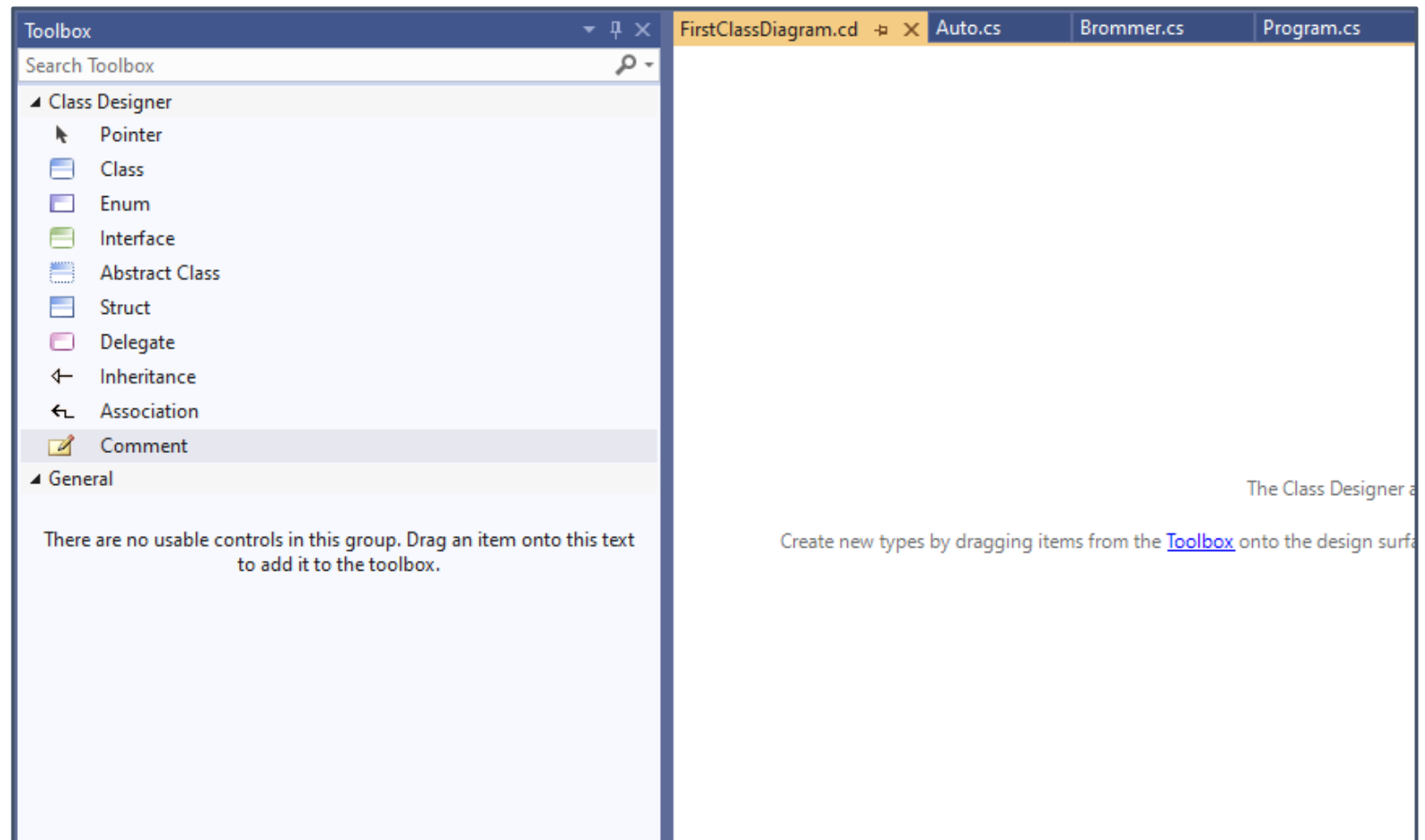
Klassendiagram

- De Class Diagram moet nog ingevuld worden:
 - Toolbox
 - Bestaande klassen



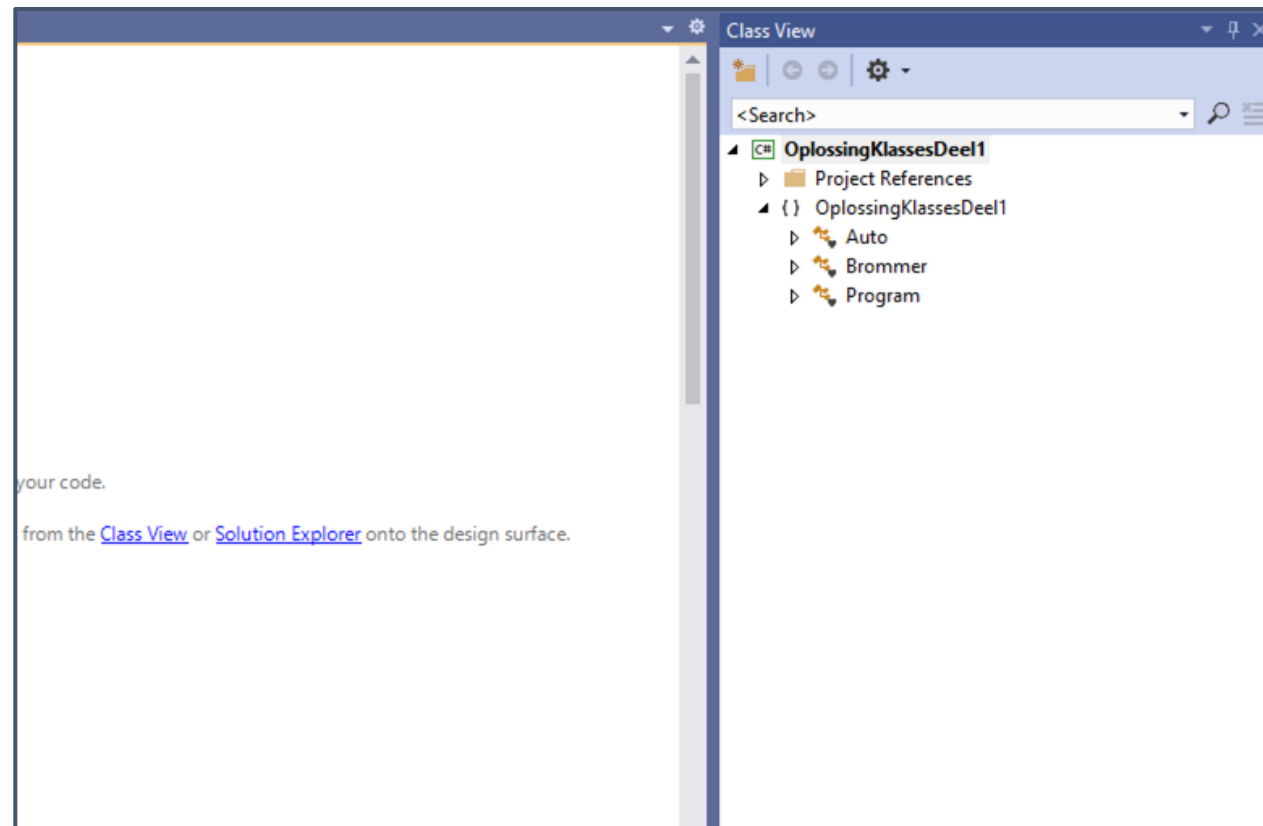
Klassendiagram

- Werk met de toolbox om de diagram te bouwen.



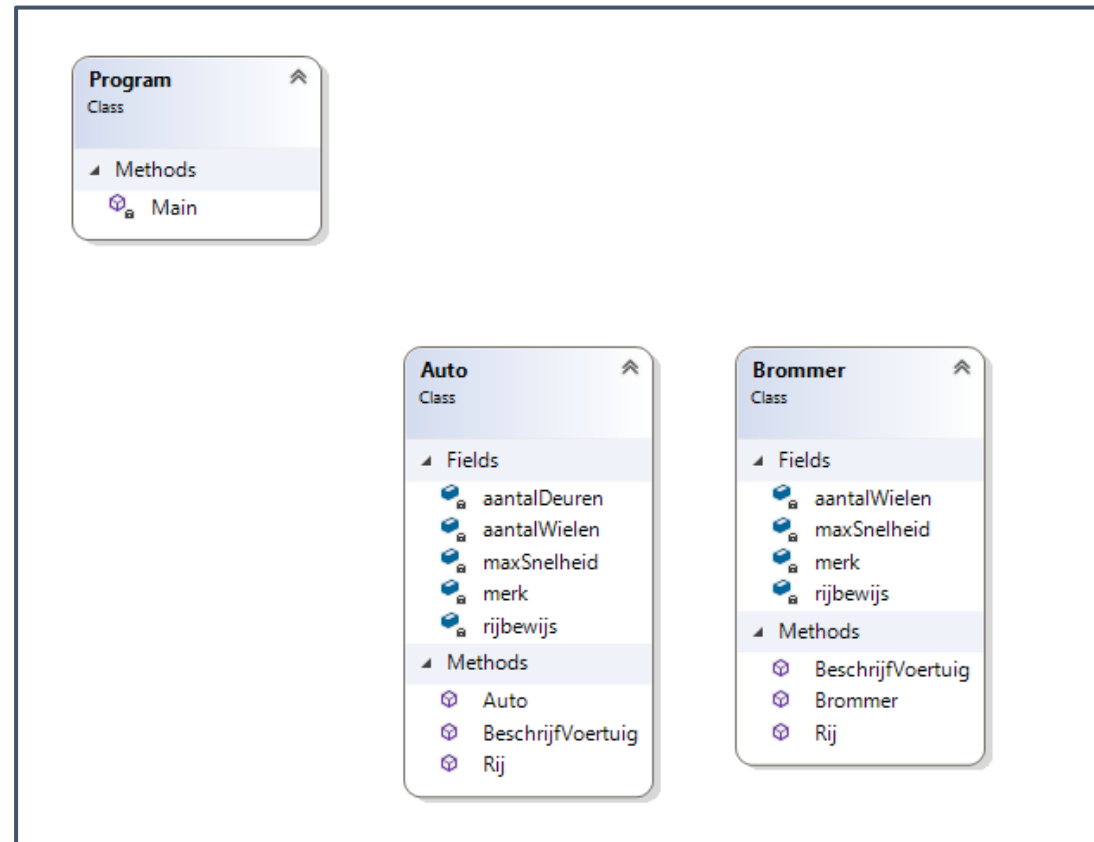
Klassendiagram

- “View” bestaande klassen uit de Class View of Solution Explorer.



Klassendiagram

- In Solution Explorer: RMK Project > View > View Class Diagram



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(TxtGeta11.Text), float.Parse(TxtGeta12.Text))}";
```