

The background features abstract green geometric shapes. On the left, a solid green trapezoid points upwards. On the right, a complex arrangement of overlapping translucent green triangles and polygons creates a layered, crystalline effect. The central text is positioned between these two main graphic elements.

Object Oriented programming/modelling

The beginning

Even voorstellen...

- ▶ Erik Ellinger
- ▶ 11 jaar Hogeschool Alkmaar
- ▶ 18 jaar Bedrijfsleven
 - ▶ CMG (nu CGI)
 - ▶ SNS Reaal (nu Athora)
- ▶ 7 Hogeschool InHollland
 - ▶ BIM
 - ▶ Programmeren
 - ▶ Databases
 - ▶ Data Integratie



Object georiënteerd programmeren/modelleren

- ▶ OO Programmeren
 - ▶ C#
 - ▶ Practicum toets
- ▶ OO Modelleren
 - ▶ Unified Modelling Language (UML)
 - ▶ Object diagram
 - ▶ State transition diagram
 - ▶ Sequence diagram
 - ▶ Theorietentamen
- ▶ Vakken zijn tijdens de lessen niet strikt gescheiden



Werkwijze

- ▶ Docent: Uitleg met voorbeeldslides
- ▶ Studenten: voorbeelden nabouwen en testen
 - ▶ Voor de snelle werkers: slides op Moodle
- ▶ Studenten: Opdrachten uit boek maken
 - ▶ Alleen bespreken bij problemen of onduidelijkheden
- ▶ Opdrachten op de virtual machine maken
- ▶ Tentamen op de virtual machine
 - ▶ Zelfgemaakte opdrachten mag je erbij houden!



Geschiedenis

- ▶ C
 - ▶ 1978
 - ▶ Opvolger van B
 - ▶ Procedurele taal - alle code is onderdeel van een functie
- ▶ C++
 - ▶ 1983
 - ▶ Uitbreiding op C
 - ▶ Object georiënteerde taal
- ▶ C#
 - ▶ 2001
 - ▶ Uitbreiding op C++, Java, Pascal
 - ▶ Garbage collection
 - ▶ Geen pointers
 - ▶ Veel uitbreidingen

Stack

- ▶ Eerste voorbeeld programma
- ▶ Vergelijk met een stapel dienbladen
 - ▶ Push - leg er een blad op
 - ▶ Pop - haal er een blad vanaf
- ▶ Last in first out
 - ▶ LIFO
- ▶ Eerst het belang van een stack
- ▶ Daarna bouwen we een klassiek procedureel programma
- ▶ Vervolgens de OO versie



Belang van de stack

- ▶ Het mechanisme achter function calls
- ▶ Voorbeeld: stupid program
- ▶ Stack
 - ▶ Code
 - ▶ Globale en statische data
 - ▶ Parameters en variabelen

```
static void Main(string[] args)
{
    int Getal0=0;

    void Een()
    {
        int Getal1= 1;
        Twee();
    }

    void Twee()
    {
        int Getal2 = 2;
    }

    Een();
}
```

Heap:
Dynamisch variabelen
Stack:
parameters en variabelen
Globale en statische data
Machine code
Machine code

Belang van de stack

- ▶ OS is geladen
- ▶ Programmacode is geladen
- ▶ Main roept Een() aan
- ▶ Getal0 gaat op de stack



```
static void Main(string[] args)
{
    int Getal0=0;

    void Een()
    {
        int Getal1= 1;
        Twee();
    }
    void Twee()
    {
        int Getal2 = 2;
    }
    Een();
}
```

0
Programma code
OS code

Belang van de stack

- ▶ Een() vult Getal1
- ▶ Een() roept Twee() aan
- ▶ Getal1 gaat op de stack



```
static void Main(string[] args)
{
    int Getal0=0;

    void Een()
    {
        int Getal1= 1;
        Twee();
    }
    void Twee()
    {
        int Getal2 = 2;
    }
    Een();
}
```

1
0
Programma code
OS code

Belang van de stack

- ▶ Twee() vult Getal2
- ▶ Getal2 gaat op de stack



```
static void Main(string[] args)
{
    int Getal0=0;

    void Een()
    {
        int Getal1= 1;
        Twee();
    }
    void Twee()
    {
        int Getal2 = 2;
    }
    Een();
}
```

2
1
0
Programma code
OS code

Belang van de stack

- ▶ Twee() is klaar
- ▶ Getal2 gaat van de stack



```
static void Main(string[] args)
{
    int Getal0=0;

    void Een()
    {
        int Getal1= 1;
        Twee();
    }
    void Twee()
    {
        int Getal2 = 2;
    }
    Een();
}
```

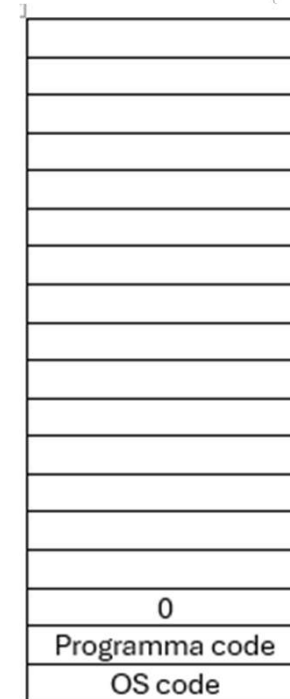
1
0
Programma code
OS code

Belang van de stack

- ▶ Een() is klaar
- ▶ Getal1 gaat van de stack
- ▶ Terug naar Main()

```
static void Main(string[] args)
{
    int Getal0=0;

    void Een()
    {
        int Getal1= 1;
        Twee();
    }
    void Twee()
    {
        int Getal2 = 2;
    }
    Een();
}
```

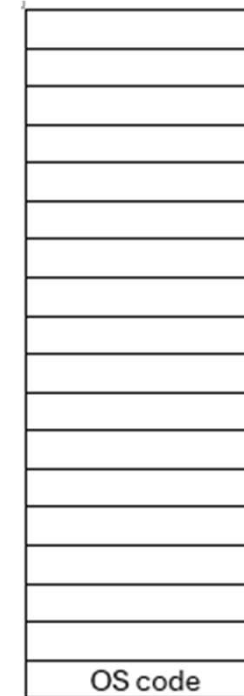


Belang van de stack

- ▶ Main() is klaar
- ▶ Getal0 gaat van de stack
- ▶ Programmacode van de stack
- ▶ Terug naar Operating system

```
static void Main(string[] args)
{
    int Getal0=0;

    void Een()
    {
        int Getal1= 1;
        Twee();
    }
    void Twee()
    {
        int Getal2 = 2;
    }
    Een();
}
```



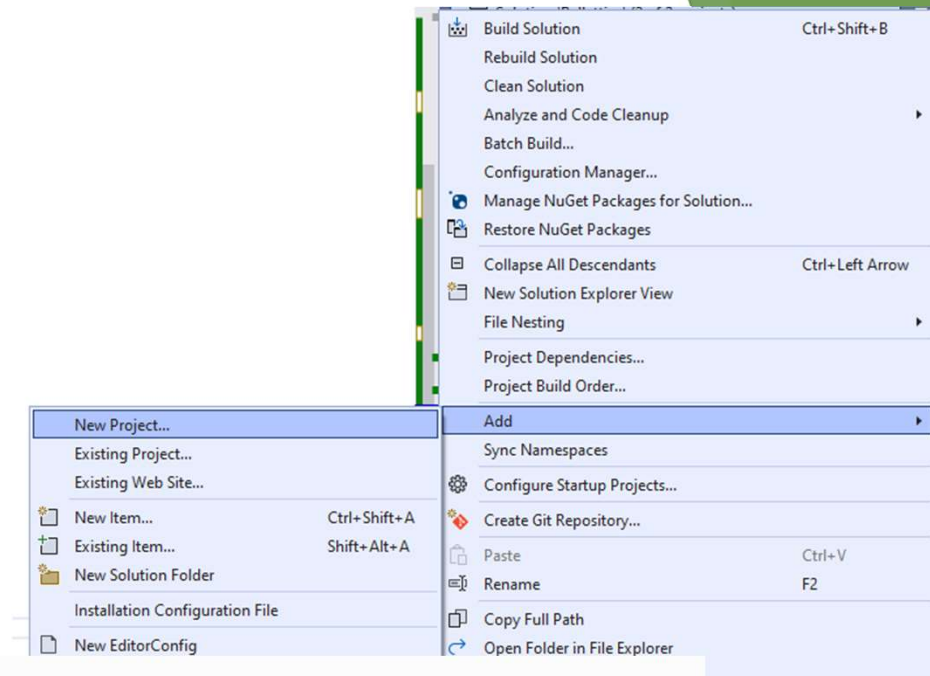
Bouw van de stack

- ▶ Voorbeeld: stack met getallen (integers)
- ▶ Getallen worden opgeslagen in een array
- ▶ Functies:
 - ▶ Push(Getal)
 - ▶ Zet een getal op de stack
 - ▶ Pop()
 - ▶ Haalt een getal van de stack
 - ▶ Print()
 - ▶ Drukt de getallen in de stack af



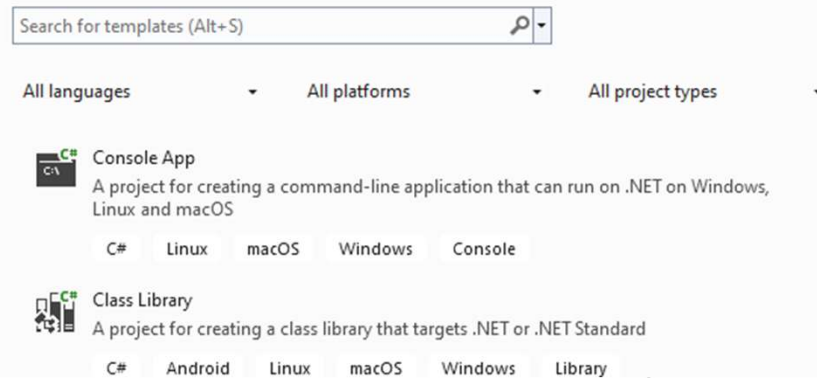
Nieuw project maken

- Rechtermuis klik op de solution



Add a new project

Recent project templates



Nieuw project maken

Configure your new project

Console App C# Linux macOS Windows Console

Project name

Stack

Location

C:\Users\Erik.Ellinger\OneDrive - Hogeschool Inholland\Documents\VSProjects\LessenT12\Ballettje: ▾

...

Additional information

Console App C# Linux macOS Windows Console

Framework ⓘ

.NET 8.0 (Long Term Support) ▾

☒ Do not use top-level statements ⓘ

☐ Enable native AOT publish ⓘ

Stap 1: declaraties

- ▶ Meteen achter Main()
- ▶ Let op: naam met kleine letters beginnen
- ▶ Camel casing

```
static void Main(string[] args)
{
    const int maxAantalElementen = 10;
    int aantal = 0;
    int[] stack = new int[maxAantalElementen];
}
```

Stap 2: functies

- ▶ Push()
 - ▶ Tussen de { } van Main()
- ▶ Naam met hoofdletters beginnen
- ▶ Controleer of er nog plaats is op de stack
- ▶ Vul de stack op positie *Aantal*
- ▶ Arrays beginnen op positie 0, dus aantal wijst naar de eerste lege positie
- ▶ Hoog *Aantal* met 1 op voor de volgende positie

```
const int maxAantalElementen = 10;
int aantal = 0;
int[] stack = new int[maxAantalElementen];

void Push(int getal)
{
    if (aantal < maxAantalElementen)
    {
        stack[aantal] = getal;
        aantal++;
    }
    else
    {
        Console.WriteLine("Maximum aantal is bereikt");
    }
}
```

Stap 2: functies

- ▶ Pop()
 - ▶ Tussen de { } van Main()
 - ▶ Na Push()
- ▶ Naam met hoofdletter beginnen
- ▶ Controleer of er elementen in de stack zitten
- ▶ Arrays beginnen op positie 0, dus aantal wijst naar de eerste lege positie
- ▶ Verminder Aantal met 1
- ▶ Return de waarde op die positie

```
int Pop()
{
    if (aantal > 0)
    {
        aantal--;
        return stack[aantal];
    }
    else
    {
        Console.WriteLine("Stack is leeg");
        return 0;
    }
}
```

Stap 2: functies

- ▶ Print()
 - ▶ Tussen de { } van Main()
 - ▶ Na Pop()
- ▶ Druk alle waarden af van de teller en van stack[0] tot en met stack[Aantal-1]

```
void Print()
{
    for (int i = 0; i < aantal; i++)
    { Console.WriteLine(i + ":" + stack[i]); }
}
```

Stap 3: Hoofdprogramma

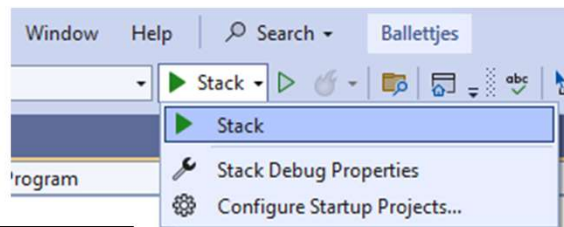
- ▶ Functie Main zoals in C
- ▶ Tussen de { } van Main()
 - ▶ Na print()
- ▶ Lees Commando's:
 - ▶ Push
 - ▶ Pop
 - ▶ Print
 - ▶ Stop
- ▶ Int.Parse() zet een string om naar getal

```
String Commando = "";
while (Commando != "Stop")
{
    Console.Write("Geef commando:");
    Commando = Console.ReadLine();
    if (Commando == "Push")
    {
        Console.Write("Geef getal:");
        Push(int.Parse(Console.ReadLine()));
    }
    else if (Commando == "Pop")
    {
        Console.WriteLine(Pop());
    }
    else if (Commando == "Print")
        Print();
}
```

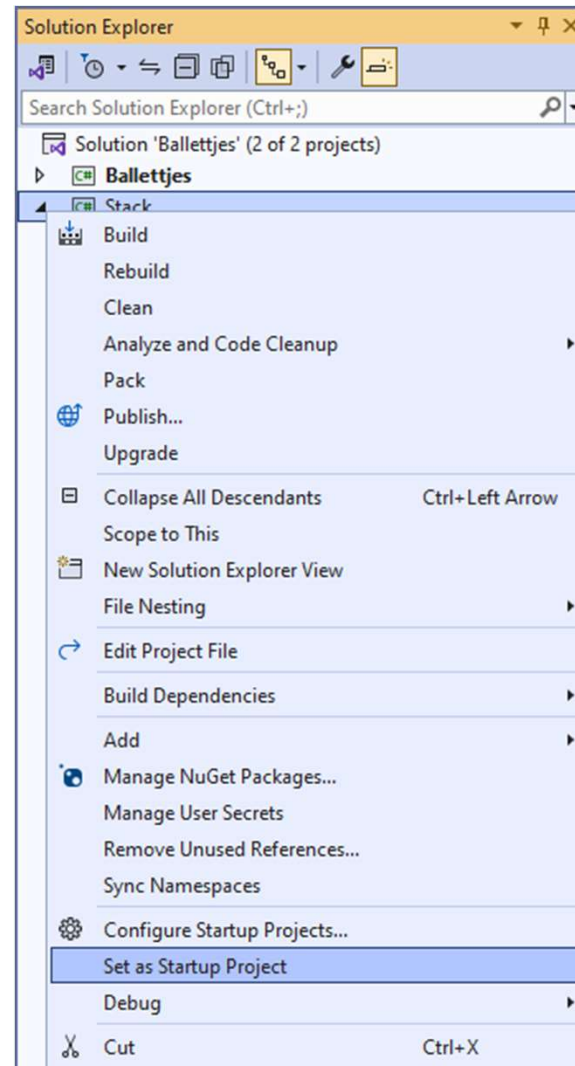
Start

- ▶ Maak van je project het startup project
 - ▶ Project is vet gedrukt
- ▶ Start

- ▶ F5
- ▶ Via menu



```
Geef commando:Push
Geef getal:5
Geef commando:Push
Geef getal:6
Geef commando:Push
Geef getal:7
Geef commando:Print
0:5
1:6
2:7
Geef commando:Pop
7
Geef commando:|
```

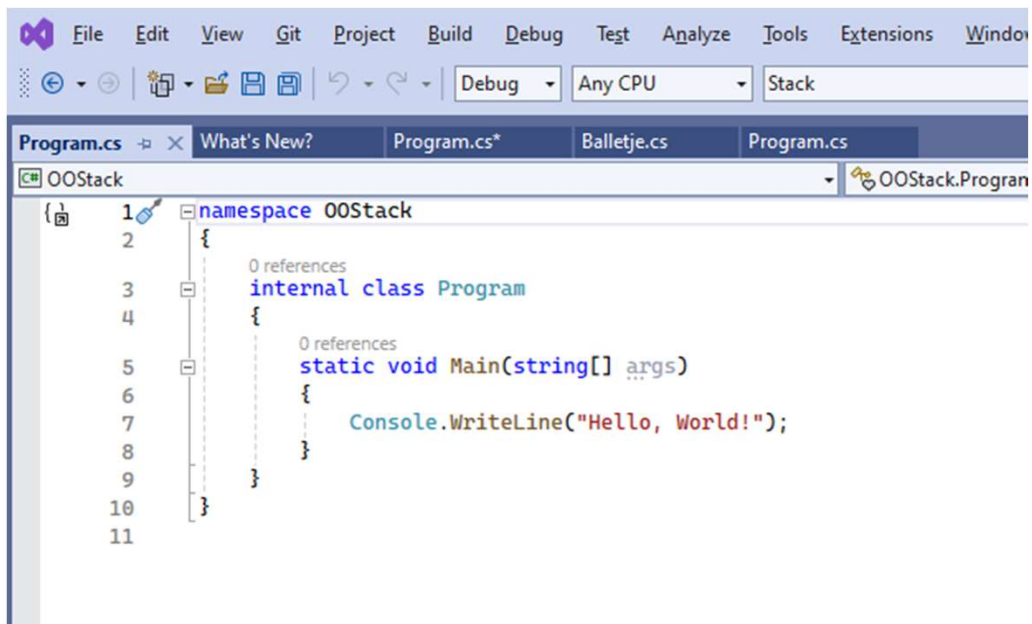


De OO variant

- ▶ In OO programma's worden functies en variabelen die bij elkaar horen ondergebracht in een Class
- ▶ Een class is eigenlijk een complex datatype
- ▶ Vervolgens declareer je een variabele van deze class
- ▶ Onze class wordt OOSTack
 - ▶ Variabelen
 - ▶ Functies Pop, Push, Pront

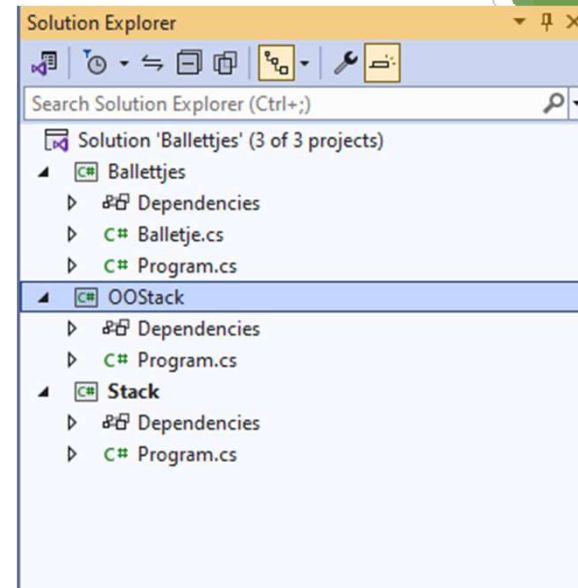


Nieuw project: OOSTack



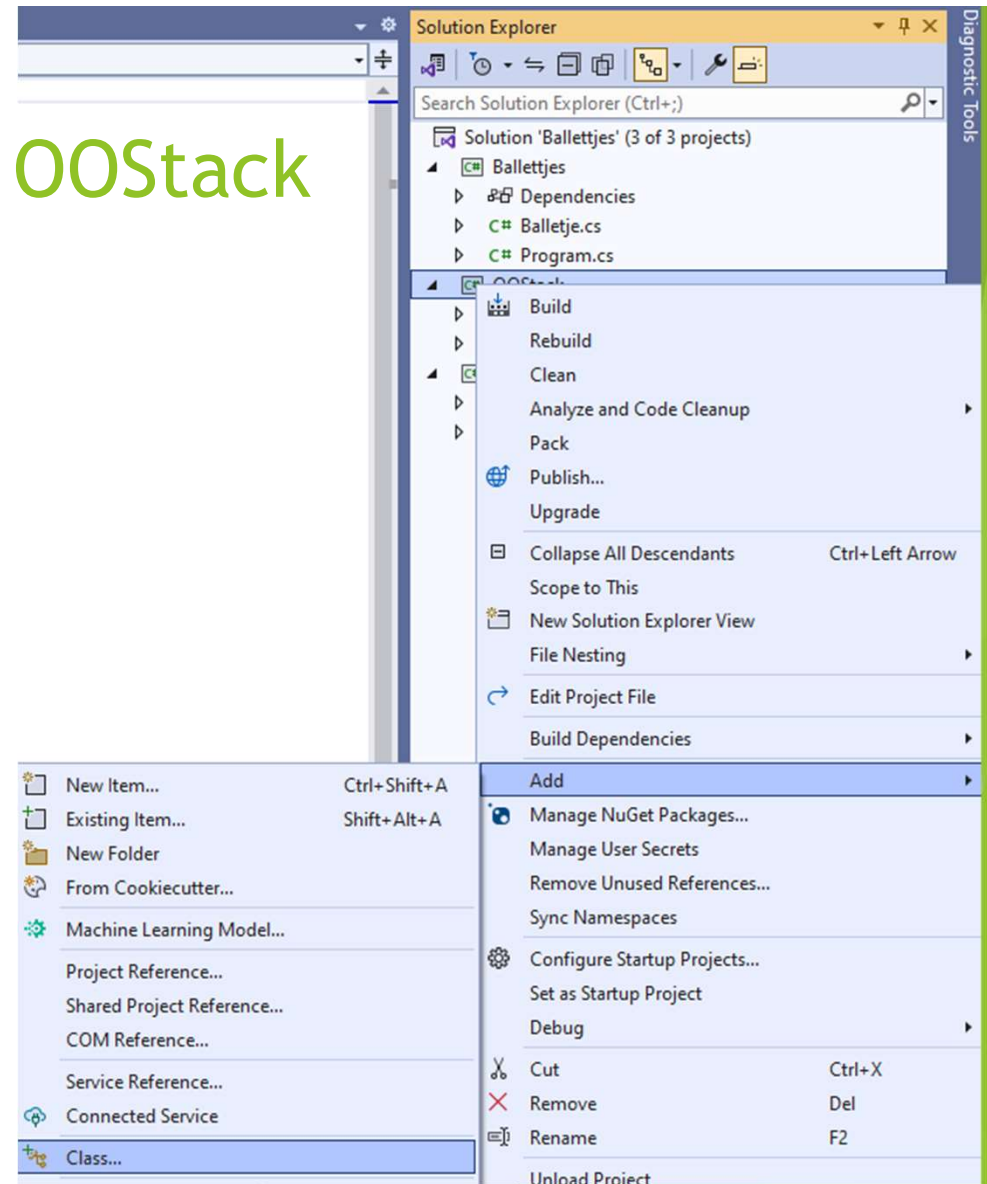
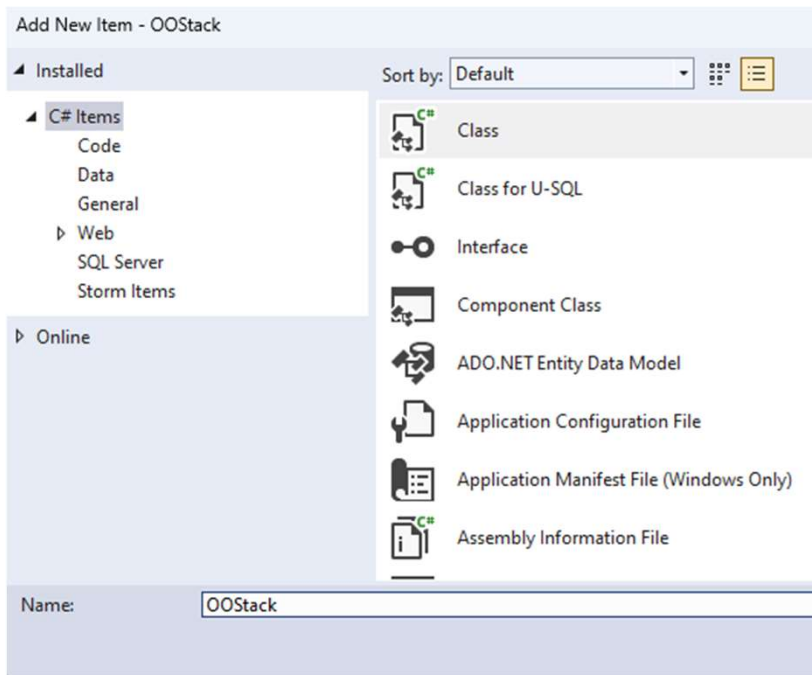
The screenshot shows the Visual Studio IDE with the 'Program.cs' file open in the 'OOSTack' project. The code defines a namespace 'OOSTack' containing an internal class 'Program' with a static 'Main' method that prints 'Hello, World!' to the console.

```
1 namespace OOSTack
2 {
3     0 references
4     internal class Program
5     {
6         0 references
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Hello, World!");
10        }
11    }
```



Een nieuwe Class maken: OOSTack

- Maak een nieuwe class
 - OOSTack



Class OOSTack

- ▶ Kopieer de declaraties en functies
 - ▶ Plaats ze in de Class
- ▶ Zet het woord **public** voor de functies
 - ▶ Public: buiten de class zichtbaar
 - ▶ Private alleen in de class zichtbaar
 - ▶ Private is default

```
internal class OOSTack
{
    const int maxAantalElementen = 10;
    int aantal = 0;
    int[] stack = new int[maxAantalElementen];
    1 reference
    public int Aantal
    {
        get { return aantal; }
    }
    1 reference
    public void Push(int getal)
    {
        if (aantal < maxAantalElementen)
        {
            stack[aantal] = getal;
            aantal++;
        }
        else
        {
            Console.WriteLine("Maximum aantal is bereikt");
        }
    }

    1 reference
    public int Pop()
    {
        if (aantal > 0)
        {
            aantal--;
            return stack[aantal];
        }
        else
        {
            Console.WriteLine("Stack is leeg");
            return 0;
        }
    }

    1 reference
    public void Print()
    {
        for (int i = 0; i < aantal; i++)
        { Console.WriteLine(i + ":" + stack[i]); }
    }
}
```

Hoofdprogramma

- ▶ Kopieer het hoofdprogramma
- ▶ Plaats het in de Main functie
- ▶ Declareer de stack
 - ▶ Een Class is eigenlijk een datatype
- ▶ Wijzig de functie aanroepen
- ▶ Test!

Declareer

Aanroep

Aanroep

Aanroep

```
static void Main(string[] args)
{
    OOStack MyStack = new OOStack();

    String Commando = "";
    while (Commando != "Stop")
    {
        Console.WriteLine("Geef commando:");
        Commando = Console.ReadLine();
        if (Commando == "Push")
        {
            Console.WriteLine("Geef getal:");
            MyStack.Push(int.Parse(Console.ReadLine()));
        }
        else if (Commando == "Pop")
        {
            Console.WriteLine(MyStack.Pop());
        }
        else if (Commando == "Print")
        {
            MyStack.Print();
        }
    }
}
```

OO termen

- ▶ Een Class is eigenlijk een datatype : *OOSTack*
 - ▶ Een struct uitgebreid met functies
- ▶ Objecten zijn variabelen van het datatype class: *OOSTack MyStack*
 - ▶ Worden met het keyword **new** aangemaakt : *MyStack = new OOSTack*
- ▶ Functies in een Class heten methods : *Push, Pop, ..*
- ▶ Variabelen in een class heten members: *aantal*
- ▶ Methods en members kunnen *public* zijn of *private*
 - ▶ Public begint met hoofdletter: *Print*
 - ▶ Private begint met een kleine letter: *stack*
 - ▶ Default is private

Properties

- ▶ Je kunt de toegang tot membervariabelen regelen via Properties
- ▶ Get: opvragen
- ▶ Set: waarde geven
- ▶ Extra code is mogelijk
- ▶ Let op hoofd- en kleine letters
 - ▶ Private variabelen kleine eerste letter
 - ▶ Properties beginnen met een hoofdletter

Geen set mogelijk

```
const int maxAantalElementen = 10;
int aantal = 0;
int[] stack = new int[maxAantalElementen];
0 references
public int Aantal
{
    get { return aantal; }
    set { aantal = value; }
}
```

```
const int maxAantalElementen = 10;
int aantal = 0;
int[] stack = new int[maxAantalElementen];
0 references
public int Aantal
{
    get { return aantal; }
}
```

Properties

- ▶ Het aantal elementen kan nu opgevraagd worden
- ▶ Aantal kan niet gewijzigd worden opdat er geen Set is
- ▶ 9.3 in boek

Aantal opvragen

```
while (Commando != "Stop")
{
    Console.Write("Geef commando:");
    Commando = Console.ReadLine();
    if (Commando == "Push")
    {
        Console.Write("Geef getal:");
        MyStack.Push(int.Parse(Console.ReadLine()));
    }
    else if (Commando == "Pop")
    {
        Console.WriteLine(MyStack.Pop());
    }
    else if (Commando == "Print")
    {
        MyStack.Print();
    }
    else if (Commando == "Aantal")
    {
        Console.WriteLine(MyStack.Aantal + " element(en)");
    }
}
```

Voordelen van objectoriëntatie

Hergebruik

- Klassen kunnen worden hergebruikt,
- Maakt software-ontwikkeling efficiënter

Verbergen van interne werking

- Information hiding/encapsulation
- Als je intern iets verandert in een klasse, merkt de rest van de applicatie daar niets van
- Maakt applicaties robuuster

Scheiden van verantwoordelijkheden

- Elke klasse handelt zijn eigen zaakjes af, andere klassen/programmeurs kunnen op de publieke interface v.e. klasse vertrouwen en er gebruik van maken
- Compartmentalisatie van complexiteit: je hoeft met maar één 'compartiment' tegelijk te ontwikkelen – de realisatie wordt hierdoor makkelijker en beter testbaar!

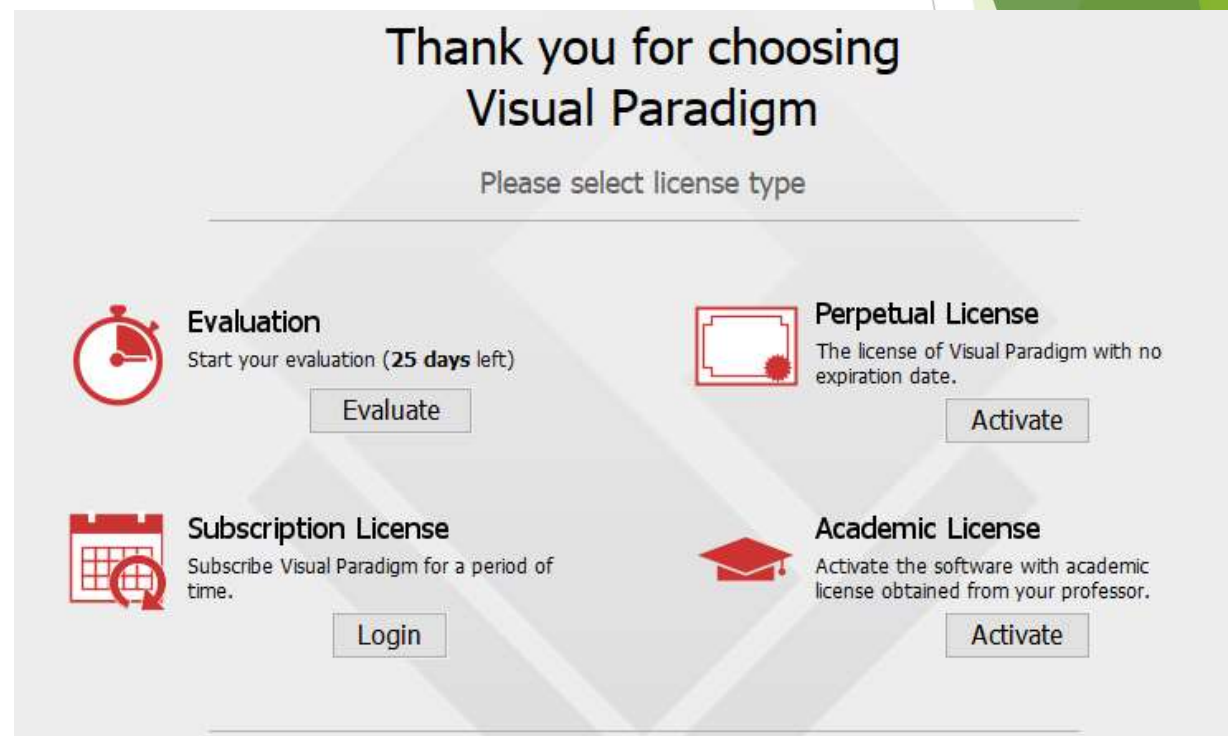
OO modelleren

- ▶ Unified Modelling language
- ▶ Meer dan 10 verschillende diagrammen
- ▶ Class diagram
 - ▶ Beschrijving van de structuur van de objecten
- ▶ Sequence diagram
 - ▶ Beschrijving van de interactie tussen objecten
- ▶ State transition diagram
 - ▶ Beschrijving van de toestanden (state) van objecten
- ▶ Gebruikte tool: Paradigm



OO Model

- Tekentool: Paradigm
- Selecteer Academic license



Paradigm activeren

- ▶ Open firefox
- ▶ Kopieer de Activatiecode
- ▶ Gebruik school email

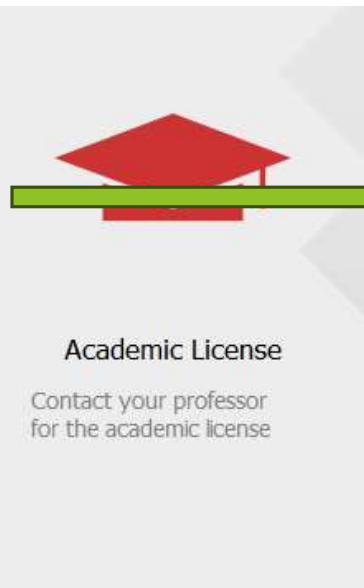
Here is your latest license:

Institute:	Hogeschool InHolland
Product:	Visual Paradigm Standard 16.3
Activation Code:	78C98-D43Z2-67959-9C99F-6W9FY
Expiry Date:	2024-09-24

Product installer of Visual Paradigm Standard 16.3 at:

64 Bit
Installer
InstallFree

Windows



Academic License

Contact your professor for the academic license

Academic Partner Program License

Copy the activation code of your license and paste into the following text field. **Internet access** is required.

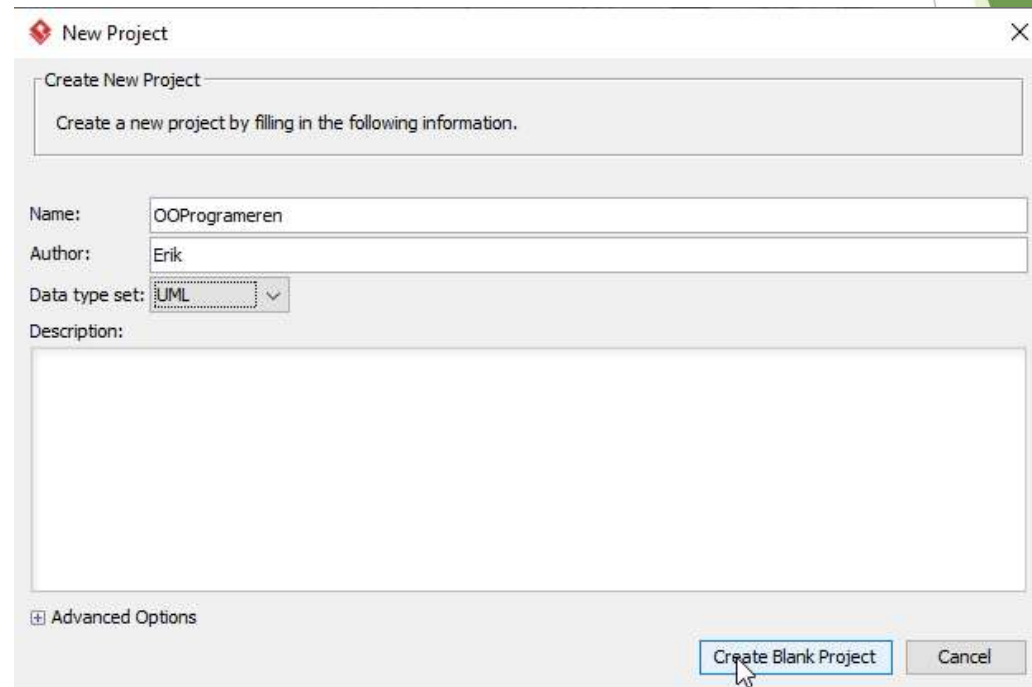
Activation Code: - - - -

Name:

Email:

Paradigm

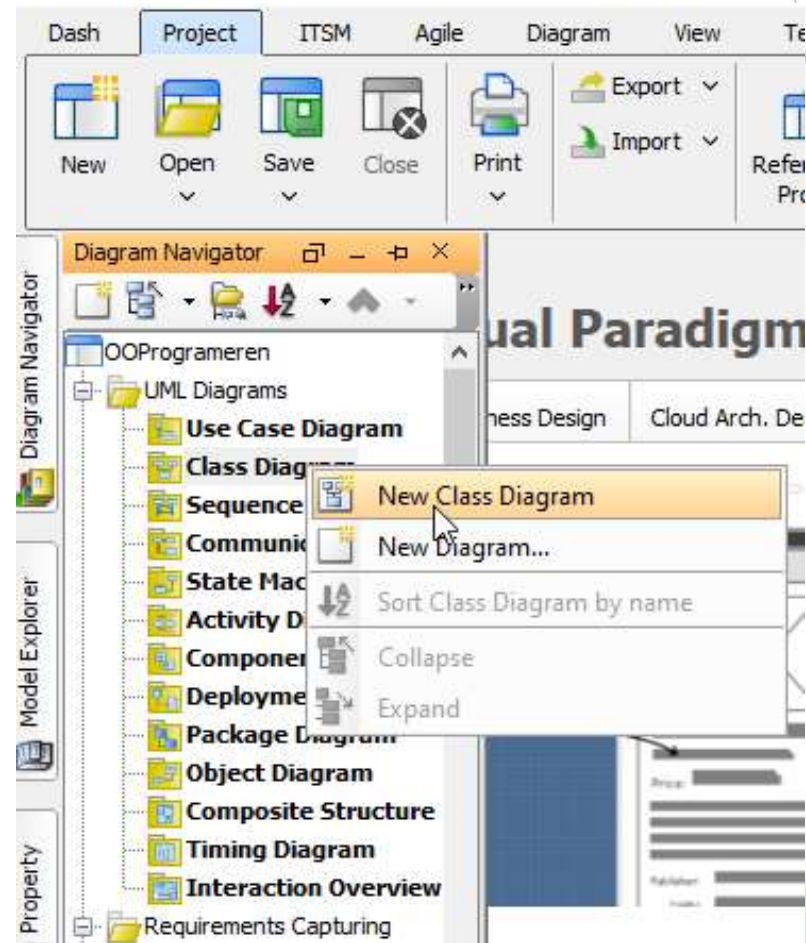
- Maak een nieuw project



The image shows a 'New Project' dialog box with a title bar containing a red diamond icon and the text 'New Project'. The dialog has a close button (X) in the top right corner. Inside, there is a section titled 'Create New Project' with the instruction 'Create a new project by filling in the following information.' Below this, there are four input fields: 'Name:' with the text 'OOProgrameren', 'Author:' with the text 'Erik', 'Data type set:' with a dropdown menu showing 'UML', and 'Description:' with a large empty text area. At the bottom left, there is a collapsed 'Advanced Options' section. At the bottom right, there are two buttons: 'Create Blank Project' and 'Cancel'. A mouse cursor is pointing at the 'Create Blank Project' button.

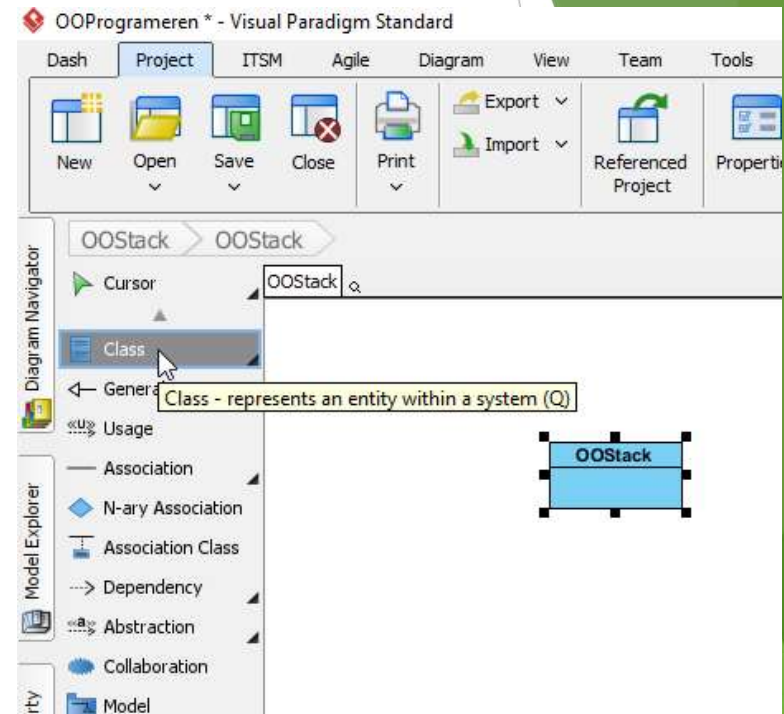
Paradigm

- ▶ Diagram navigator
- ▶ UML Diagrams
- ▶ Nieuw Class diagram: OOSTack



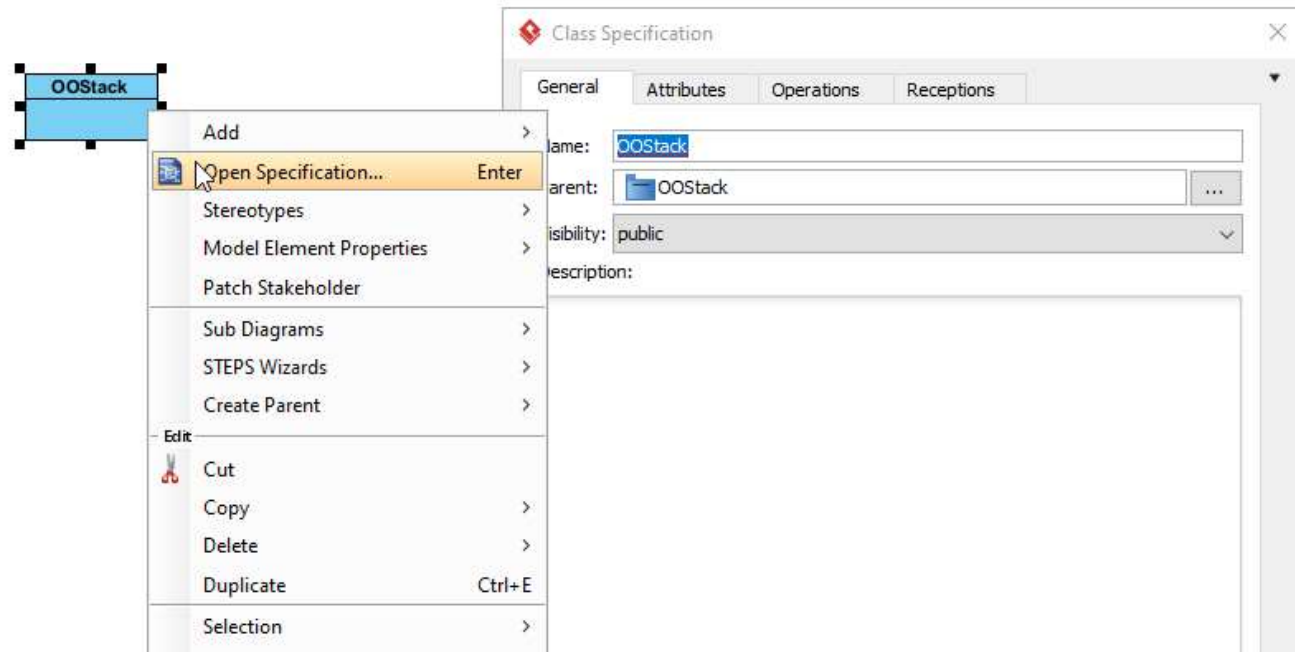
Paradigm

- Sleep een class op je diagram
- Hernoem naar OOSTack



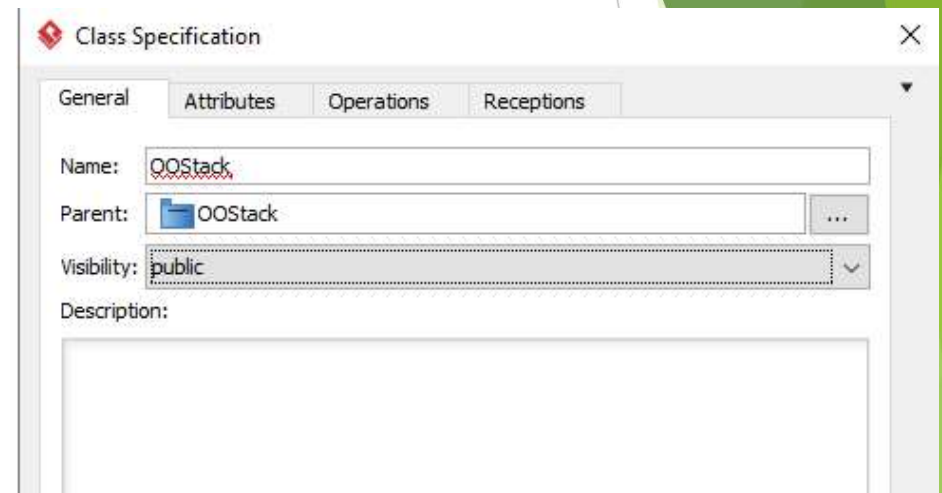
Paradigm

- Open specification



Paradigm

- ▶ Opdracht
- ▶ Vul de specificaties van de class OOSTack in
 - ▶ Attributen
 - ▶ Operations
- ▶ Denk om datatypen, visibility, properties, ...



Paradigm

- ▶ Uitwerking
- ▶ Paradigm ondersteunt code generatie in C++ en Java

OOSTack
-maxAantalElementen : int = 10
-aantal : int = 0
-stack : int[] = 10
+Pop() : int
+Push(getal : int) : void
+Print() : void

► Gegeneerde C++ code

- .h file
- .cpp file

► Throw wordt later behandeld

```
#include "OOSTack.h"
```

```
int OOSTack::OOSTack::Pop() {  
    // TODO - implement OOSTack::Pop  
    throw "Not yet implemented";  
}  
  
void OOSTack::OOSTack::Push(int getal) {  
    // TODO - implement OOSTack::Push  
    throw "Not yet implemented";  
}  
  
void OOSTack::OOSTack::Print() {  
    // TODO - implement OOSTack::Print  
    throw "Not yet implemented";  
}
```

```
namespace OOSTack {  
    class OOSTack {  
  
    private:  
        /**  
         * Het maximaal aantal elementen in de stack  
         */  
        int maxAantalElementen;  
        /**  
         * Het actuele aantal elementen op de stack  
         */  
        int aantal;  
        /**  
         * De stack met elementen  
         */  
        int stack[];  
  
    public:  
        /**  
         * Haalt een element van de stack  
         */  
        int Pop();  
  
        /**  
         * Plaatst een element op de stack  
         */  
        void Push(int getal);  
  
        /**  
         * Drukt alle elementen van de stack af  
         */  
        void Print();  
    };  
}
```

Opdracht

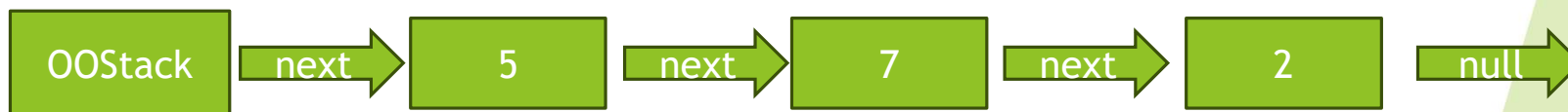
- ▶ Maak een OO variant van Pong
- ▶ Maak een class diagram van de class(en)



Stack met array

- ▶ Nadeel: maximaantal elementen
- ▶ Nieuwe opzet:
 - ▶ De stack bevat een gelinkte lijst met stackelementen die in principe onbeperkt kan groeien.
 - ▶ Een stackelement bevat data en een verwijzing naar het volgende stackelement
- ▶ De functies veranderen niet.

```
const int maxAantalElementen = 10;  
int aantal = 0;  
int[] stack = new int[maxAantalElementen];
```



Nieuw project

► GelinkteStack

Configure your new project

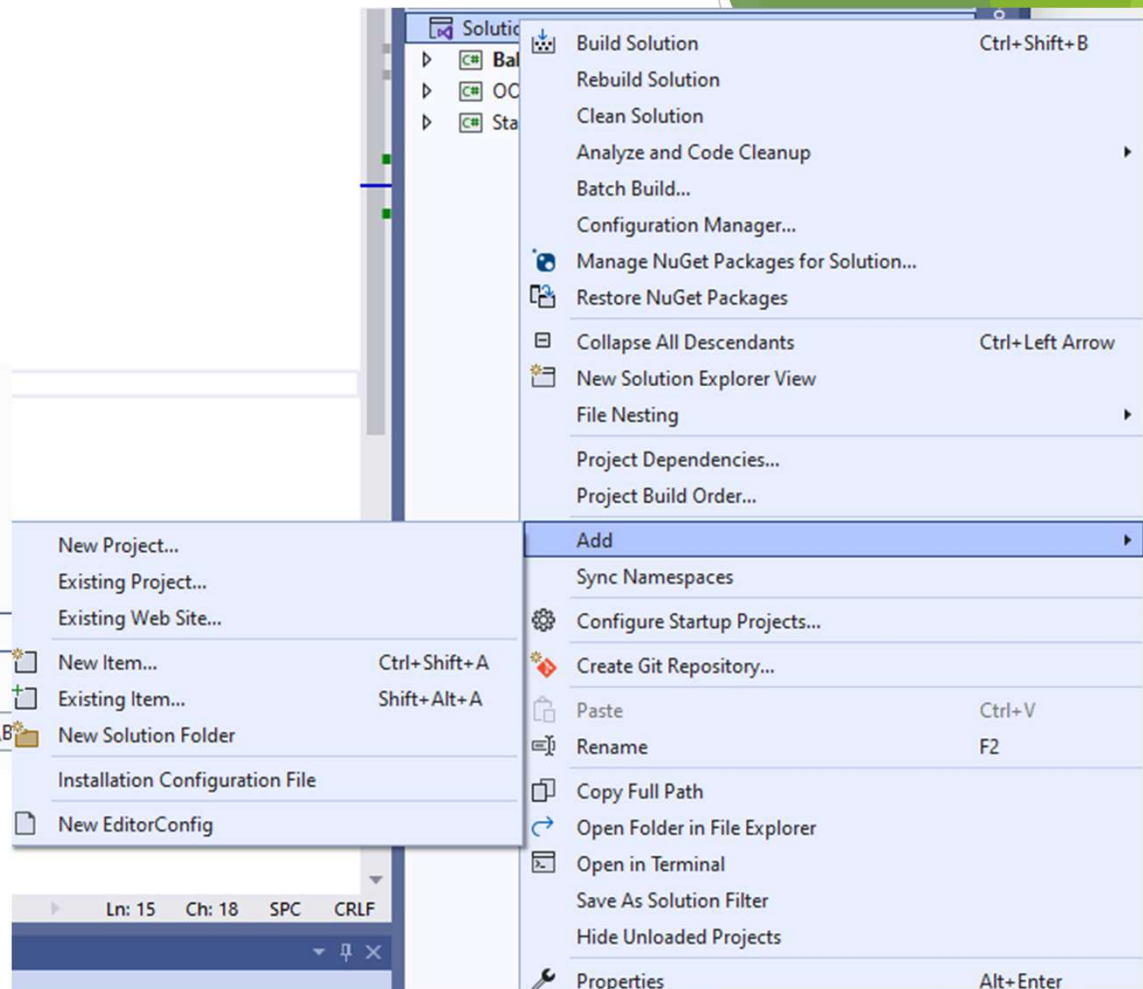
Console App C# Linux macOS Windows Console

Project name

GelinkteStack

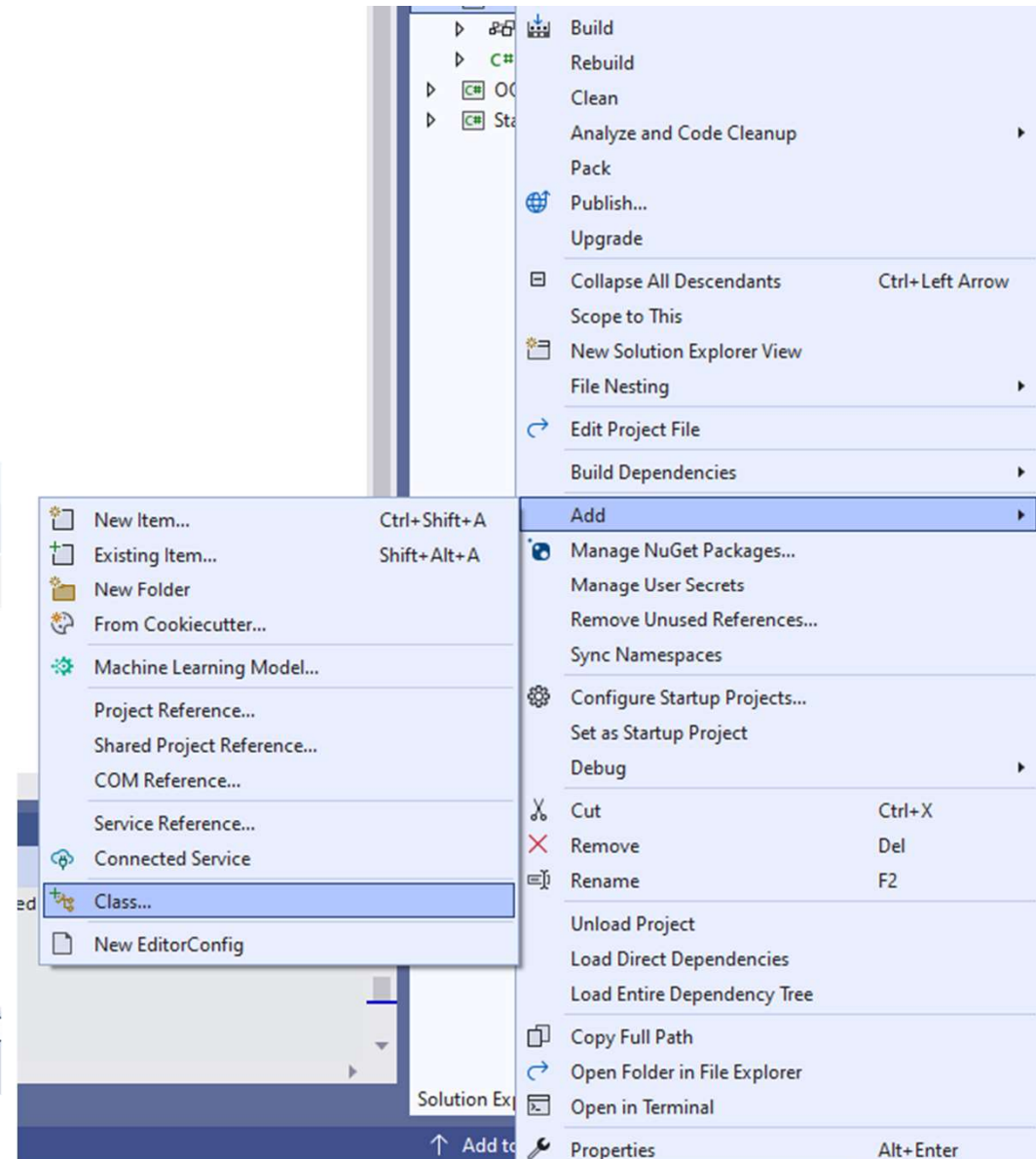
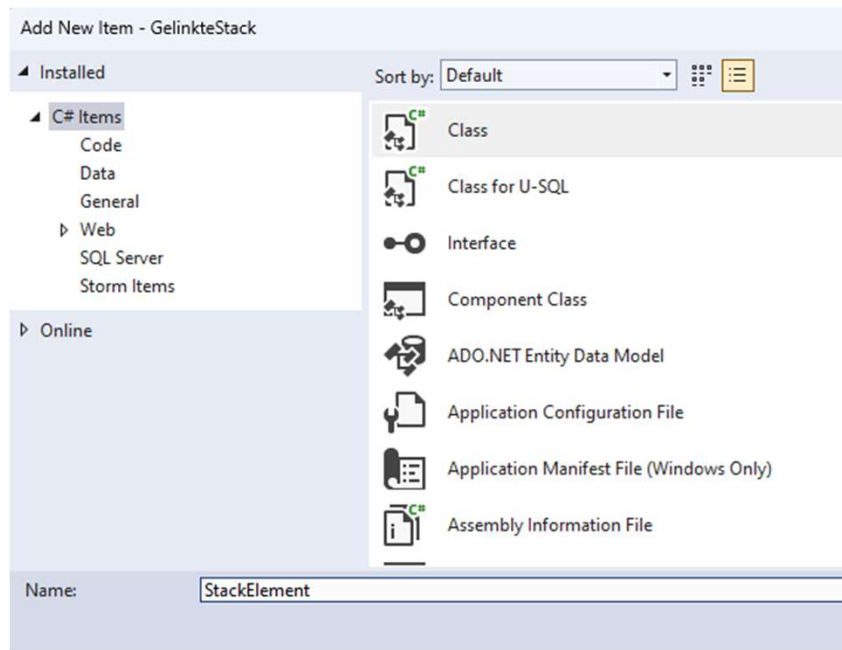
Location

C:\Users\Erik.Ellinger\OneDrive - Hogeschool Inholland\Documents\VSProjects\LessenT12\B



Nieuwe Class

► StackElement



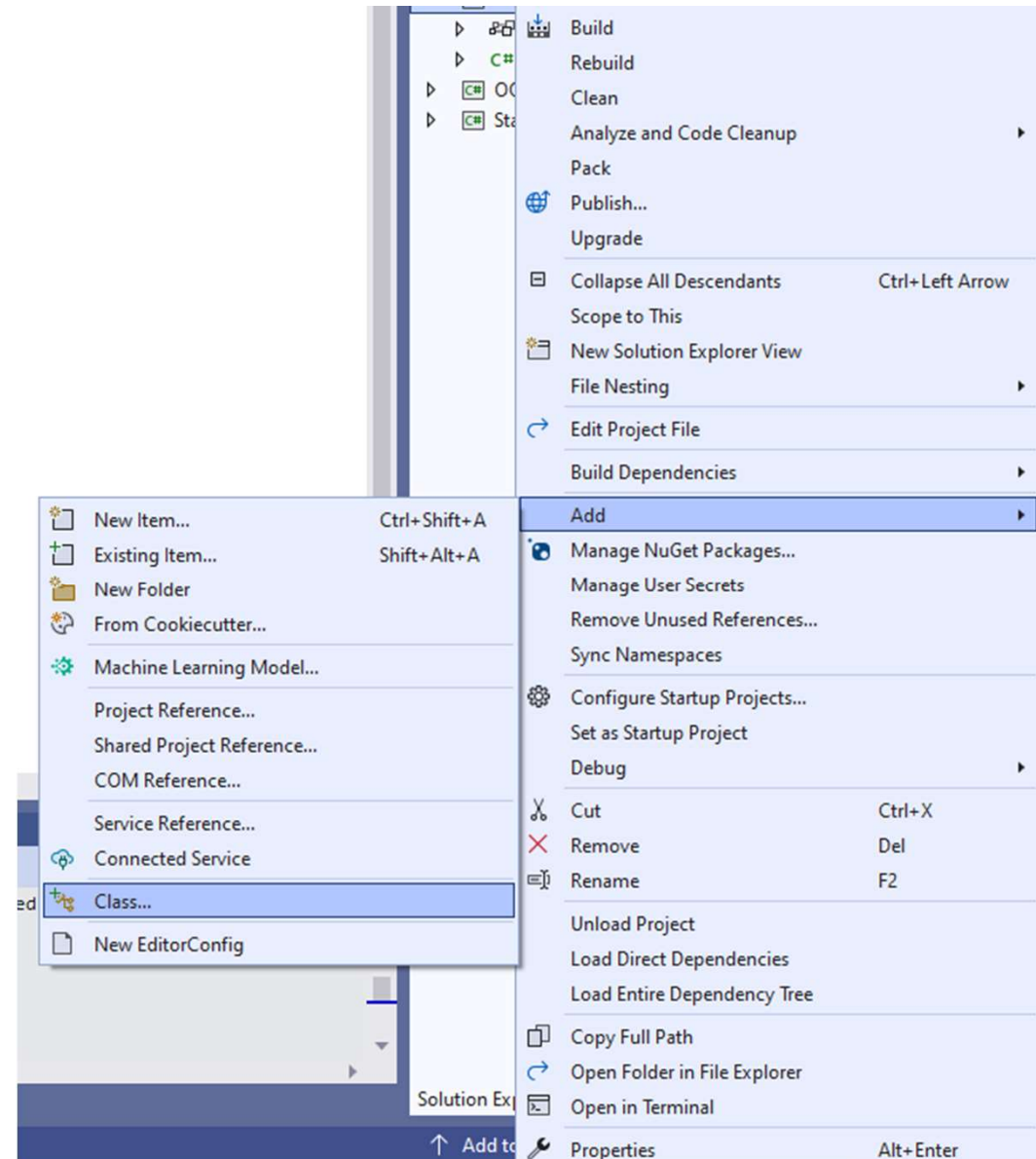
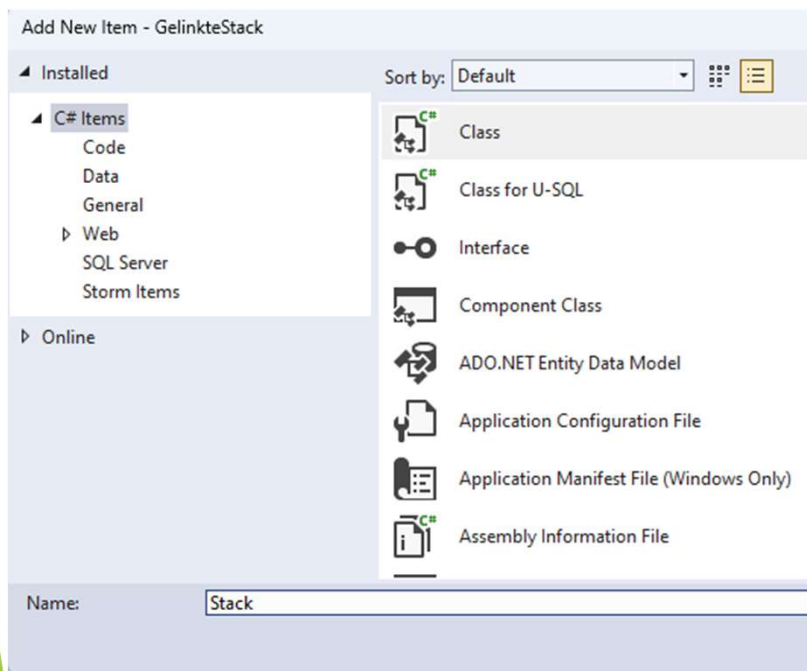
Class StackElement

- Eenvoudige versie. Alles public

```
4 references  
internal class StackElement  
{  
    public int Data = 0;  
    public StackElement Next = null;  
    .....  
}
```

Nieuwe Class

► Stack



Push en Pop

► Push

- Maak een nieuw element
- Vul het data veld
- Het nieuwe element wordt top

► Pop

- Check of de stack gevuld is
- Haal de data op
- Maak het volgende elementtop
- Retourneer hetgetal

```
internal class Stack
{
    StackElement Top = null;

    0 references
    public void Push(int Getal)
    {
        StackElement NieuweTop = new StackElement();
        NieuweTop.Data = Getal;
        NieuweTop.Next = Top;
        Top = NieuweTop;
    }

    0 references
    public int Pop()
    {
        if (Top != null)
        {
            int Getal = Top.Data;
            Top = Top.Next;
            return Getal; ;
        }
        else
        {
            Console.WriteLine("Stack is leeg");
            return 0;
        }
    }
}
```


Print

- ▶ Ieder element kan zichzelf afdrukken
- ▶ De stack vraagt het eerste element om zichzelf af te drukken
- ▶ Het element vraagt het volgende element af te drukken

```
2 references  
public void Print()  
{  
    Top.Print();  
}
```

```
internal class StackElement  
{  
    public int Data = 0;  
    public StackElement Next = null;  
    2 references  
    public void Print()  
    {  
        // Druk jezelf af  
        Console.WriteLine(Data);  
        //druk daarna de het volgende element af  
        if (Next != null)  
        {  
            Next.Print();  
        }  
    }  
}
```

Hoofdprogramma

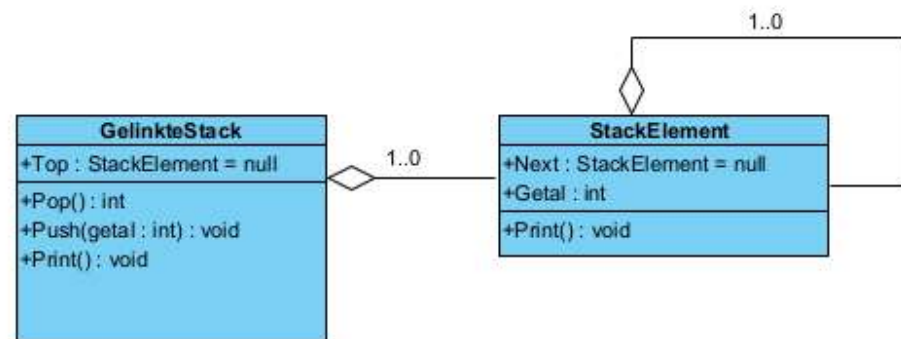
- In Main()
- En voer uit

```
static void Main(string[] args)
{
    Stack MyStack = new Stack();

    String Commando = "";
    while (Commando != "Stop")
    {
        Console.Write("Geef commando:");
        Commando = Console.ReadLine();
        if (Commando == "Push")
        {
            Console.Write("Geef getal:");
            MyStack.Push(int.Parse(Console.ReadLine()));
        }
        else if (Commando == "Pop")
        {
            Console.WriteLine(MyStack.Pop());
        }
        else if (Commando == "Print")
            MyStack.Print();
    }
}
```

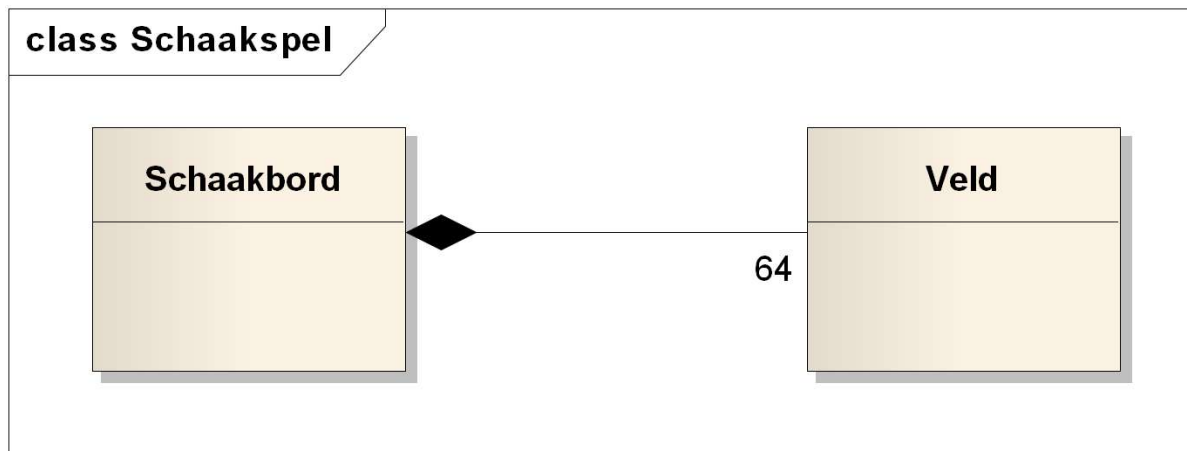
Uml Notatie

- ▶ Aggregatie
 - ▶ *Heeft een relatie*
- ▶ Een stack heeft 0 of 1 stackelement
- ▶ Een stackelement heeft 0 of 1 stackelement



UML notatie

- ▶ Verwant aan aggregatie: *Compositie*
 - ▶ Een schaakbord **bestaat uit** 64 velden
 - ▶ Een veld is onderdeel van *precies één* schaakbord
 - ▶ Als je dit minder strikt wilt: *aggregatie*



Huiswerk

- ▶ Lees hoofdstuk 9 door
- ▶ 9.4 is zelfstudie
- ▶ Maak de oefeningen in 9.5
 - ▶ Oefeningen met * komen in volgende opdrachten terug!
- ▶ Maak classdiagrams van de klassen in de opdrachten

