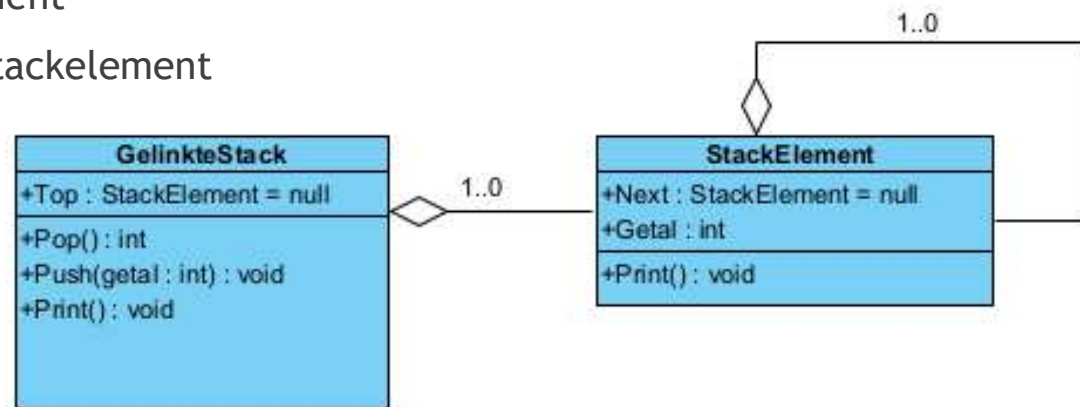


The background features abstract green geometric shapes. On the left, a solid green trapezoid points upwards. On the right, a complex arrangement of overlapping translucent green triangles and polygons creates a layered, crystalline effect. The text is centered in a clean, green, sans-serif font.

Object Oriented programming

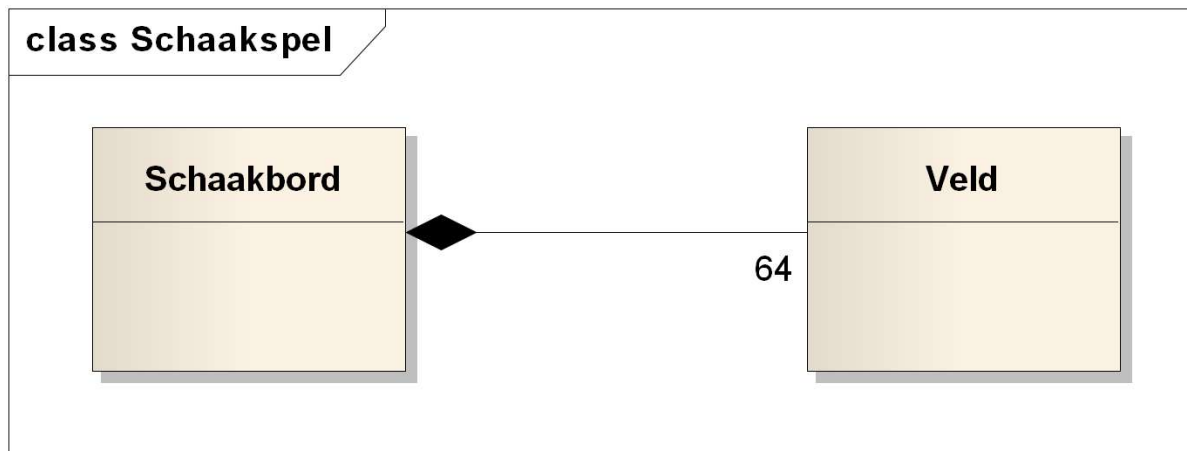
Uml Notatie

- ▶ Aggregatie
 - ▶ *Heeft een relatie*
- ▶ Kijk voor meer uitleg op <https://www.visual-paradigm.com/guide/>
- ▶ 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*



Voordelen van objectoriëntatie

Hergebruik

- Klassen kunnen worden hergebruikt, zowel binnen als buiten de applicatie
- 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
- Compartimentalisatie van complexiteit: je hoeft met maar één 'compartiment' tegelijk te ontwikkelen – de realisatie wordt hierdoor makkelijker en beter testbaar!

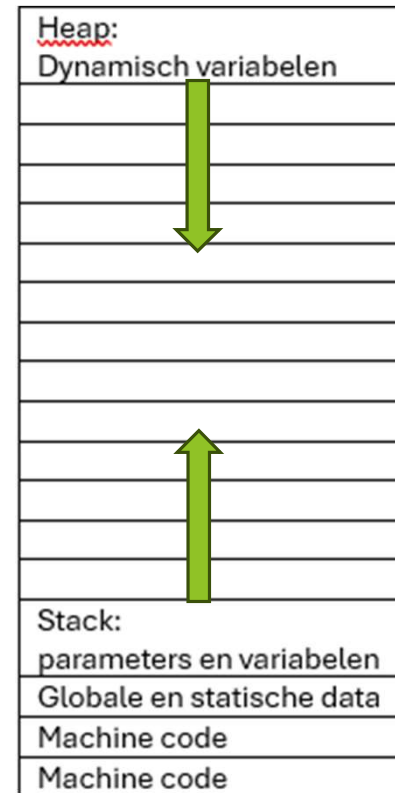
Stack en Heap

► Stack

- ▶ Start bij lage geheugenadressen
- ▶ Variabelen
- ▶ Parameters
- ▶ Return adres van functies

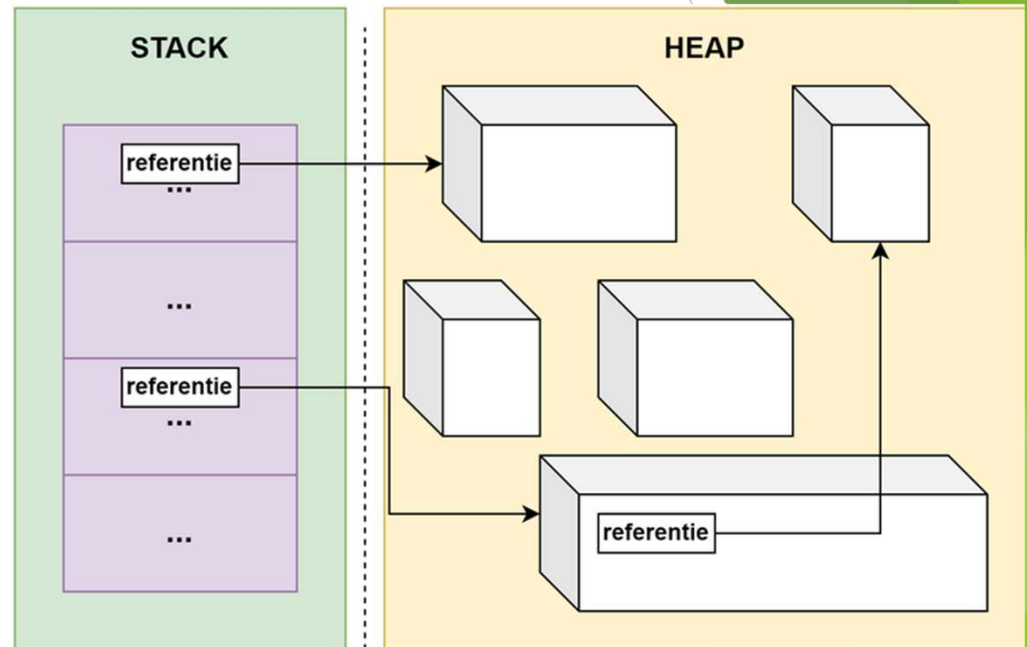
► Heap

- ▶ Start bij hoge geheugenadressen
- ▶ Dynamische variabelen
- ▶ Objecten



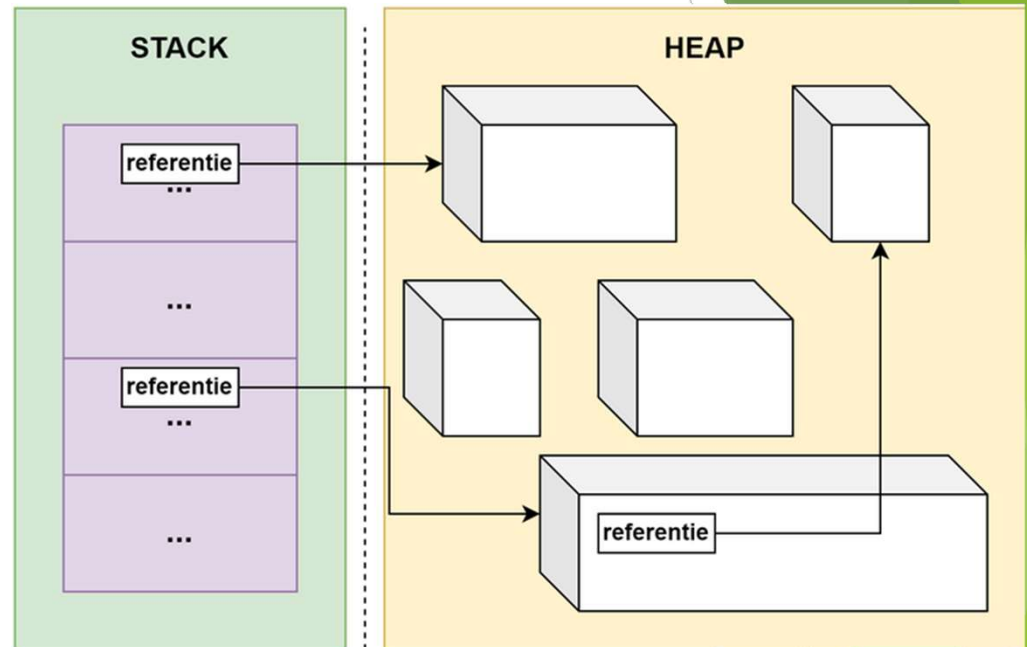
Stack en Heap in C#

- ▶ Variabelen op de stack
 - ▶ MyStack
 - ▶ Verwijst naar het object
- ▶ Object op de Heap
 - ▶ Push, Pop, Top,..
- ▶ Geen pointers?
 - ▶ Alles is een pointer
 - ▶ Arrays zijn objecten



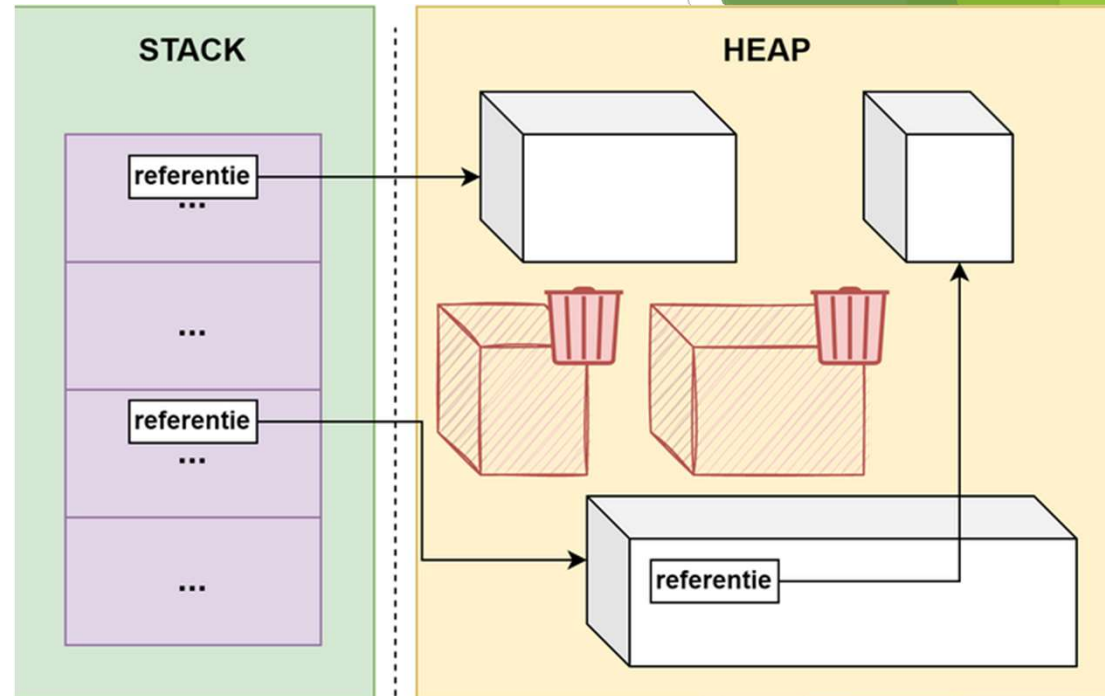
Stack en Heap in C#

- ▶ Stack is sneller
 - ▶ Eenvoudig algoritme: Push, Pop
- ▶ Heap is trager
 - ▶ Eerst een variabele maken op de stack
 - ▶ Ruimte voor het object zoeken op de Heap
 - ▶ `New Stack()`
 - ▶ Adres in de variabele plaatsen
- ▶ Stackgrootte is tevoren te berekenen, de heapgrootte niet



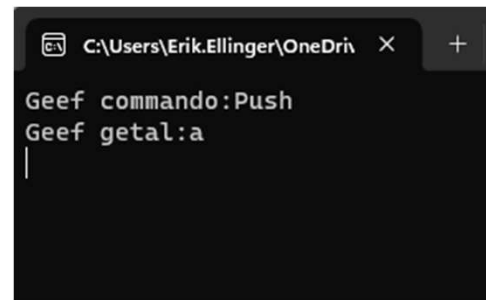
Garbage collector

- ▶ Geen delete opdracht (C, C++)
- ▶ Als het geheugen volloopt, komt de Garbage collector in actie
 - ▶ GC.Collect(x)
- ▶ Elk object op de heap wordt gemarkeerd
 - ▶ InUse = false
- ▶ De stack wordt langsgelopen. Elk object dat in gebruik is wordt de markering weggehaald
 - ▶ InUse = true
- ▶ De heap wordt gedefragmenteerd
 - ▶ Alle objecten bij elkaar



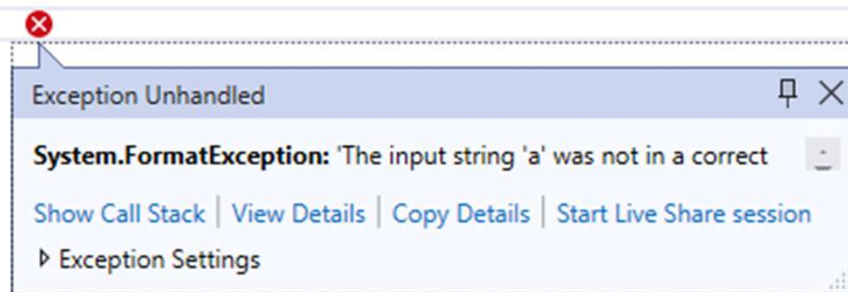
Exceptions

- ▶ Fout afhandelen van onverwachte fouten
 - ▶ Programma crashed
 - ▶ Het programma stopt 'gooit een exceptie'
- ▶ FormatException
 - ▶ De waarde kan niet naar int omgezet worden



```
C:\Users\Erik.Ellinger\OneDrive\Documents> Geef commando: Push
Geef getal: a
```

```
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();
```



Exceptions

- ▶ Crashen kun je voorkomen met een try..catch blok
 - ▶ Denk om de { }!

```
Geef commando:Push
Geef getal:t
Verkeerde invoer
Geef commando:|
```

Try

catch

```
while (Commando != "Stop")
{
    Console.Write("Geef commando:");
    try
    {
        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)");
        }
    }
    catch
    {
        Console.WriteLine("Verkeerde invoer");
    }
}
```

Exceptions

- ▶ Je kunt ook specifiek excepties opvangen
- ▶ FormatException is een class
 - ▶ e is de objectvariabele

catch

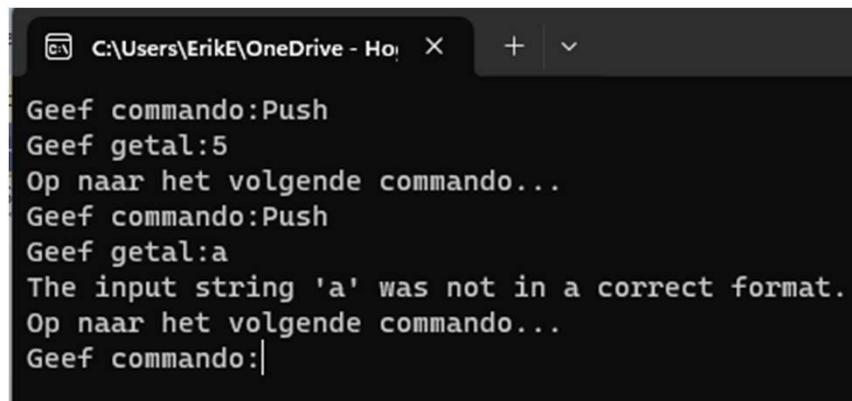
```
C:\Users\ErikE\OneDrive - Ho... x + v
Geef commando:Push
Geef getal:t
The input string 't' was not in a correct format.
Geef commando:|
```

```
while (Commando != "Stop")
{
    Console.Write("Geef commando:");
    try
    {
        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)");
        }
    }
    catch (FormatException e)
    {
        Console.WriteLine(e.Message);
    }
}
```

Exceptions

- ▶ Meerdere catchblokken zijn mogelijk
 - ▶ Exception
 - ▶ SystemException
 - ▶ IndexOutOfRangeException
 - ▶ NullReferenceException
- ▶ Finally blok wordt altijd uitgevoerd

```
}  
catch (FormatException e)  
{  
    Console.WriteLine( e.Message);  
}  
catch(SystemException e)  
{  
    Console.WriteLine(e.Message);  
}  
finally  
{  
    Console.WriteLine("Op naar het volgende commando...");  
}
```



```
C:\Users\ErikE\OneDrive - Ho...  
Geef commando:Push  
Geef getal:5  
Op naar het volgende commando...  
Geef commando:Push  
Geef getal:a  
The input string 'a' was not in a correct format.  
Op naar het volgende commando...  
Geef commando:|
```

Exceptions

- Je kunt ook zelf excepties gooien

```
try
{
    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();
    }
    else if (Commando == "Aantal")
    {
        Console.WriteLine(MyStack.Aantal + " element(en)");
    }
    else throw new Exception("Onbekend commando");
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
finally
{
    Console.WriteLine("Op naar het volgende commando...");
}
```

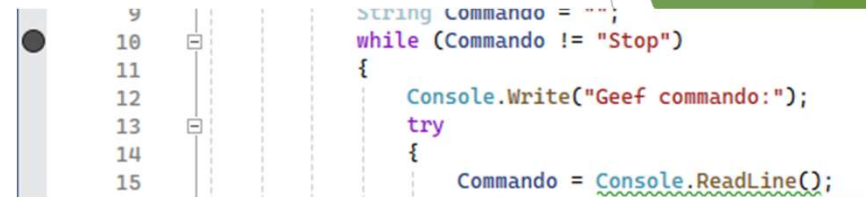
throw

catch

```
C:\Users\ErikE\OneDrive - Ho... X + v
Geef commando:sdsd
Onbekend commando
Op naar het volgende commando...
Geef commando:|
```

Foutopsporing

- ▶ Je kunt een breakpoint in de grijze kantlijn zetten
- ▶ Het programma stop als de regel bereikt is
- ▶ Daarna kun je het programma regel voor regel uitvoeren
 - ▶ F10 Volgende regel
 - ▶ F11 stap in functie
 - ▶ Shift F11 verlaat functie
 - ▶ F5 doorgaan met het programma



Constructors

- ▶ Worden uitgevoerd als new wordt aangeroepen
- ▶ New zoekt ruimte op de heap
- ▶ Voorbeeld class StackElement
 - ▶ Je wilt de members niet public maken
 - ▶ De initialisatie is niet netjes
- ▶ Door er properties van te maken kunnen ze alleen opgevraagd worden
 - ▶ Alleen een get
- ▶ Hoe geef je ze nu een waarde?

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

2 references



```
internal class StackElement
{
    private int data;
    private StackElement next;

    3 references
    public int Data { get { return data; } }
    2 references
    public StackElement Next { get { return next; } }
}
```


Constructors

- ▶ Een constructor krijgt de naam van de class
 - ▶ Geen returnwaarde
- ▶ Overloading: Meerdere constructors mogelijk
 - ▶ Als de parameters verschillen
 - ▶ Default constructor heeft geen parameters
 - ▶ De default constructor kun je ontoegankelijk maken door hem private te maken
- ▶ Push geeft nu de waarden voor stackelement mee in de constructor

Private!

```
1 reference
public void Push(int Getal)
{
    StackElement NieuweTop = new StackElement(Getal, Top);
    Top = NieuweTop;
}
```

```
internal class StackElement
{
    private int data;
    private StackElement next;

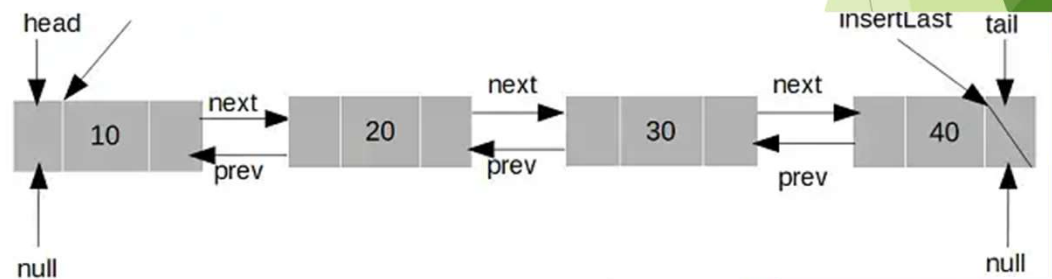
    2 references
    public int Data { get { return data; } }
    1 reference
    public StackElement Next { get { return next; } }

    1 reference
    public StackElement(int data, StackElement stackElement)
    {
        this.data = data;
        this.next = stackElement;
    }

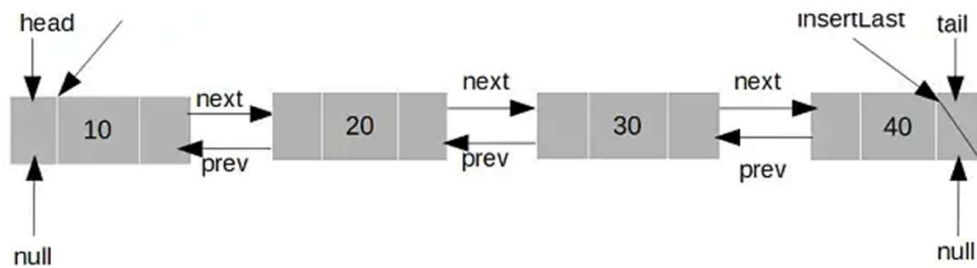
    0 references
    private StackElement() { }
}
```


Queue Opdracht

- ▶ FiFo: First in, first out
- ▶ Enqueue: voeg een element toe
 - ▶ Tail: nieuwste element
- ▶ DeQueue: verwijder het oudste element
 - ▶ Head: oudste element
 - ▶ Retourneer de waarde

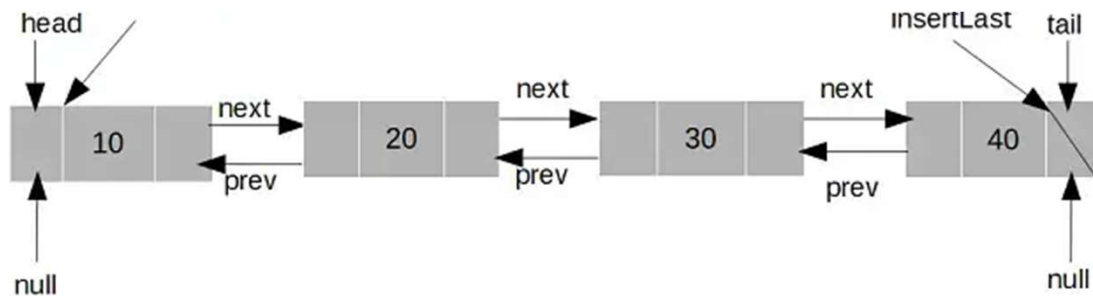


QueueElement



```
internal class QueueElement
{
    int data;
    2 references
    public int Data { get { return data; } }
    5 references
    public QueueElement Next { get; set; }
    2 references
    public QueueElement Prev { get; set; }
    1 reference
    public QueueElement(int Getal, QueueElement TailElement)
    {
        data = Getal;
        Prev = TailElement;
        Next = null;
    }
    2 references
    public void Print()
    {
        Console.WriteLine(Data);
        if (Next != null)
        {
            Next.Print();
        }
    }
}
```

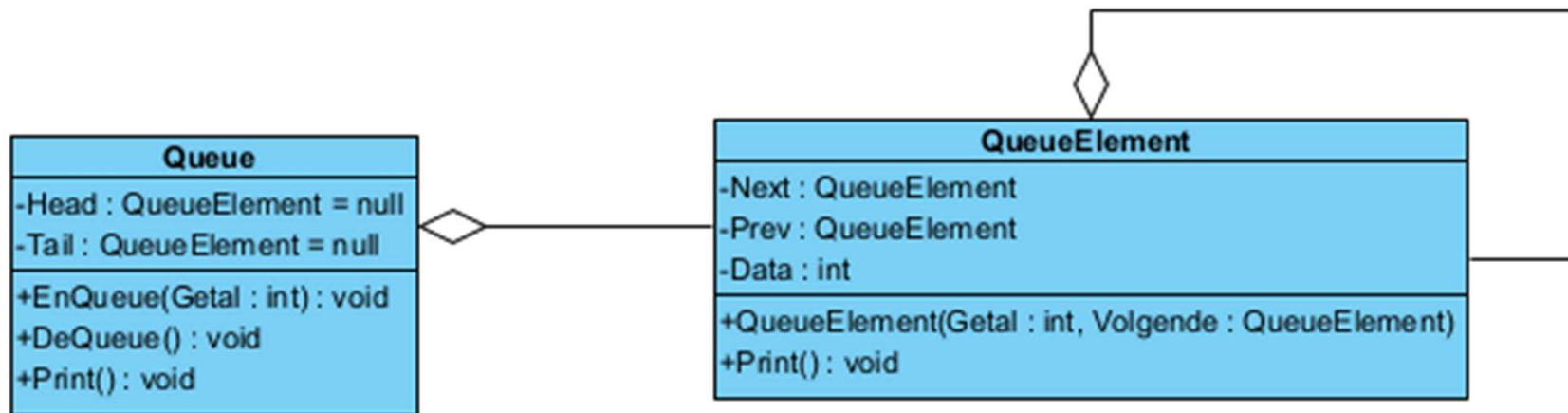
Queue



```
internal class Queue
{
    private QueueElement Tail = null, Head = null ;
    1 reference
    public void EnQueue(int Getal)
    {
        QueueElement NieuwElement = new QueueElement(Getal, Tail);
        if (Tail != null)
        {
            Tail.Next = NieuwElement;
        }
        Tail = NieuwElement;
        if (Head == null) { Head = Tail; }
    }
    1 reference
    public int DeQueue()
    {
        if (Head == null)
        {
            Console.WriteLine("Queue is leeg");
            return 0;
        }
        else
        {
            QueueElement LastElement = Head;
            Head = Head.Next;
            if (Head != null)
            {
                Head.Prev = null;
            }
            return LastElement.Data;
        }
    }
    1 reference
    public void Print()
    {
        if (Head != null)
        {
            Head.Print();
        }
    }
}
```

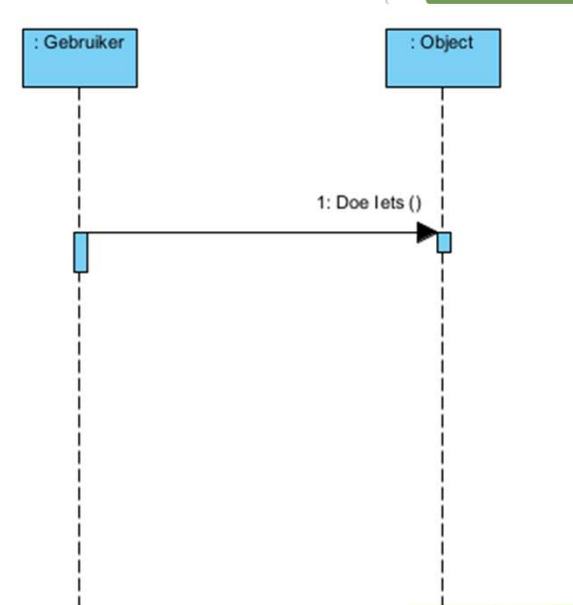
Class Diagram

- ▶ Maak gebruik van een linked list, Constructors, properties en foutafhandeling
- ▶ Commando's voor EnQueue, Dequeue en Print



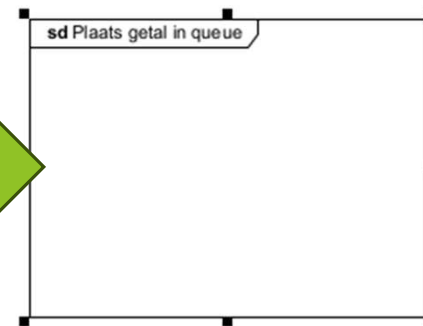
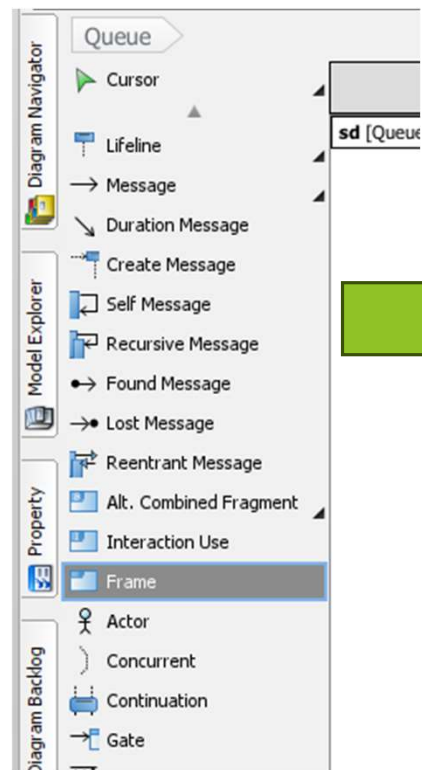
Sequence diagram

- ▶ Geeft de interactie tussen objecten weer
- ▶ Objecten met een tijdlijn
- ▶ Start punt: actor
- ▶ Aanroep van een methode heet een message
- ▶ Per activiteit van de gebruiker een Sequence diagram
 - ▶ Getal in queue plaatsen
 - ▶ Getal uit queue halen
 - ▶ Queue afdrukken
- ▶ Ieder sequence diagram in een apart frame

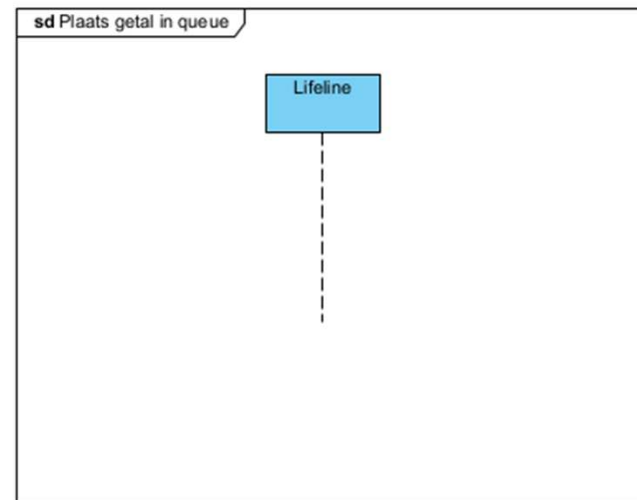
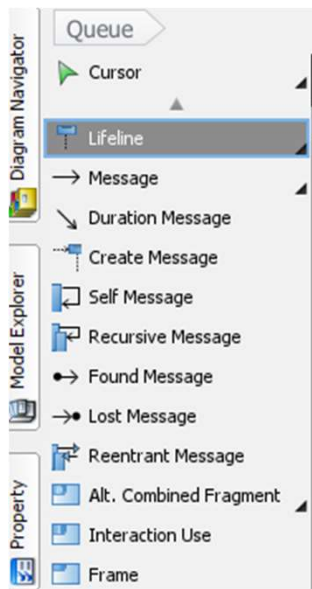


Frame voor getal in queue plaatsen

- Geef het frame een goede naam

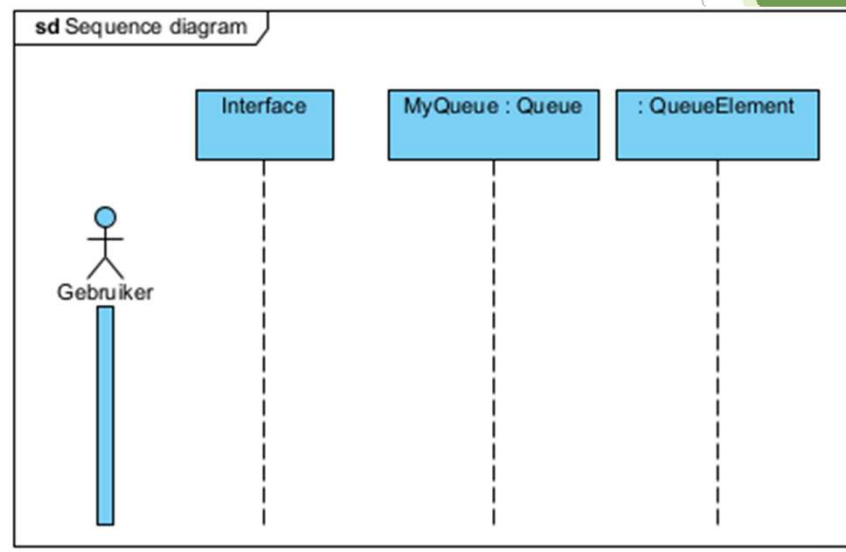


Plaats een lifeline in het frame



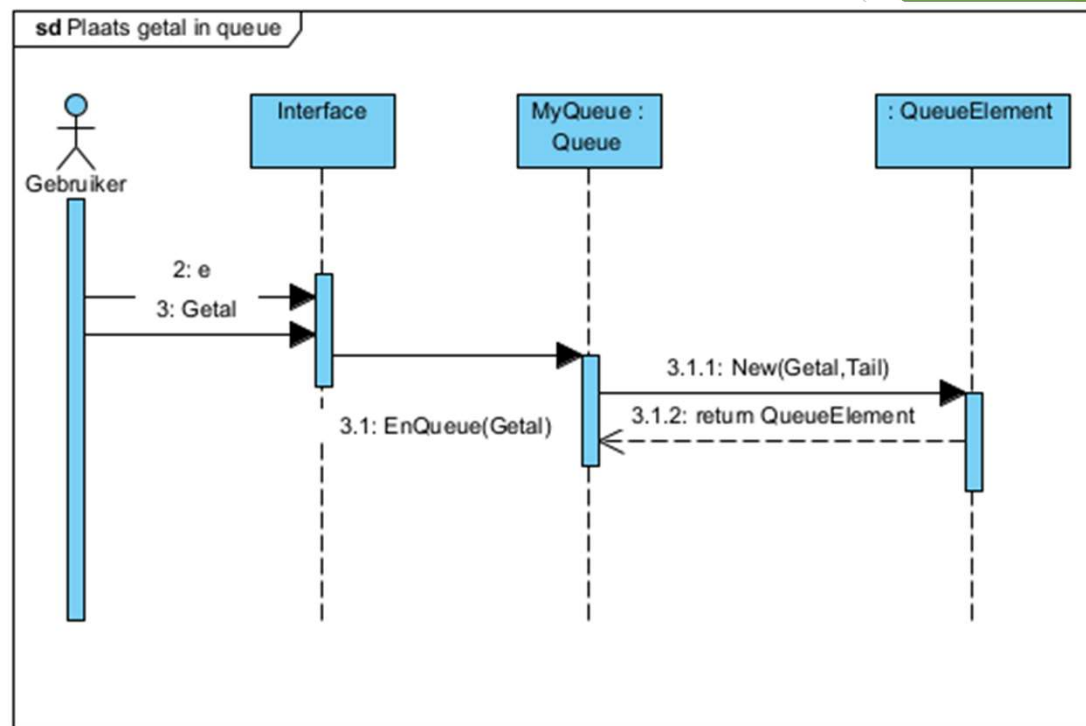
Maak voor ieder betrokken object een lifeline

- ▶ Eerste object is de actor
- ▶ Naam begint met :
 - ▶ Geeft aan dat het een object zonder naam is, of dat het hier niet relevant is
 - ▶ MyQueue: Queue is een queue object met de naam MyQueue
- ▶ Class Program niet in het diagram
 - ▶ Technische oplossing, geen ontwerp



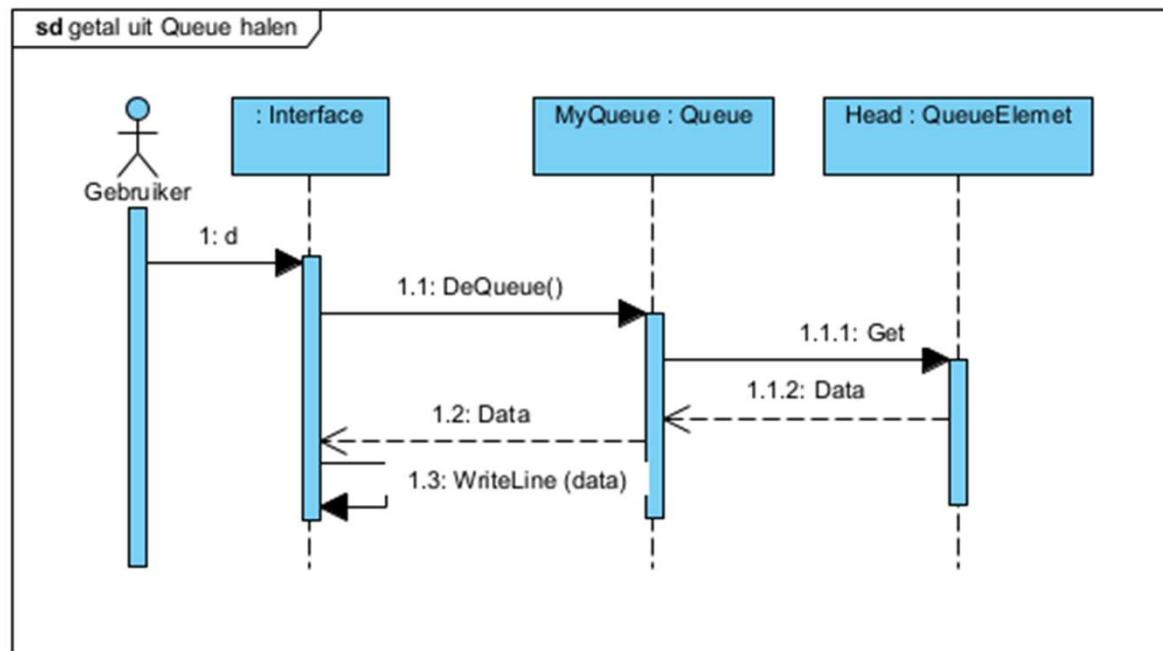
Maak messages voor de aanroepen

- Dichte pijl: aanroep van een methode
- Gestippelde pijl: return waarde



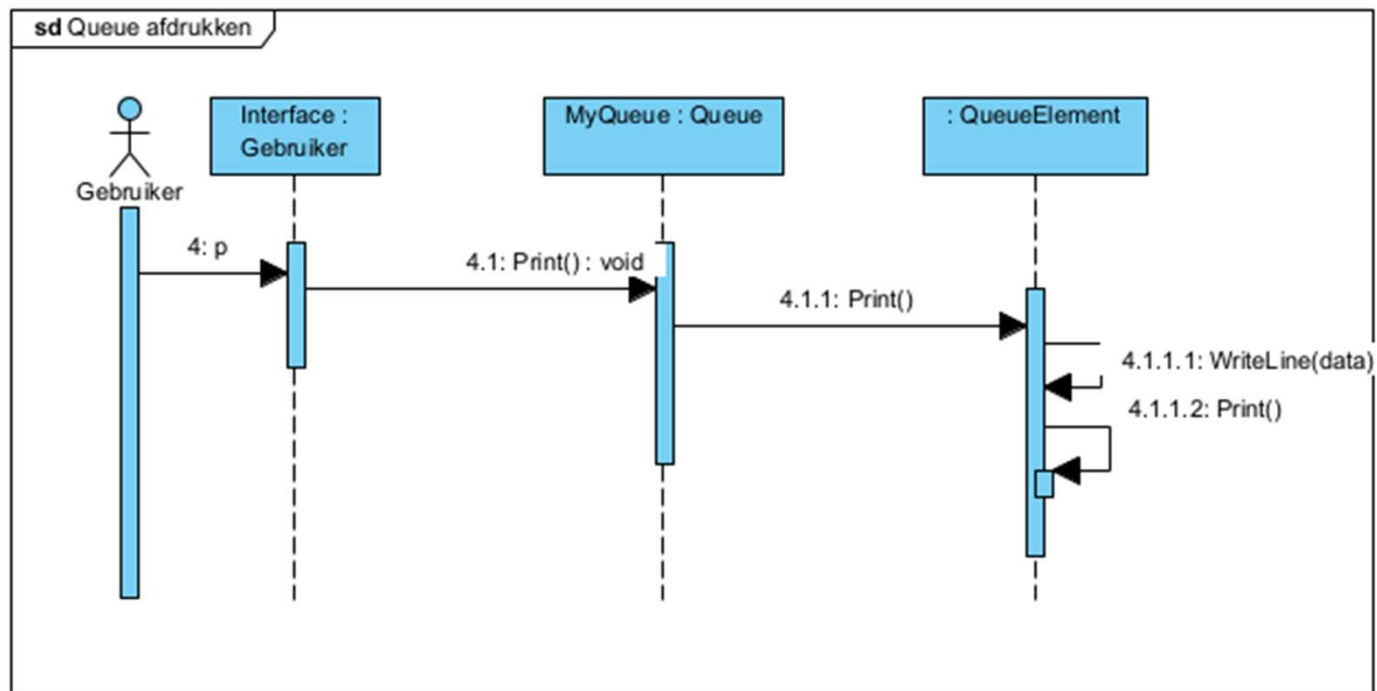
Getal uit Queue halen

- Head is een specifiek QueueElement



Queue afdrukken

- Recursie
 - Een functie die zichzelf aanroept



Overerving

- ▶ De stack die we nu hebben gebouwd kan alleen integers opslaan
- ▶ We willen een stack die alle typen elementen op kan slaan
- ▶ ‘Echte’ Stackelement eigenschappen in een superclass/parentclass
- ▶ Data in een subclass/Childclass
- ▶ Voorbeeld: integer en string
- ▶ Nieuw project: AbstracteStack
- ▶ SuperClass EenStackElement
- ▶ SubClasses: IntegerStackElement en StringStackElement

Class EenStackElement

- ▶ SuperClass
- ▶ Functionaliteit om de lijst op te bouwen in de constructor
- ▶ Functionaliteit om de stack af te drukken
- ▶ Geen functionaliteit om de data af te drukken
 - ▶ De subklasse moet hem implementeren
 - ▶ Virtuele methode
- ▶ Abstract
 - ▶ Kan geen instantie van gemaakt worden

```
internal abstract class EenStackElement
{
    private EenStackElement next;

    3 references
    public EenStackElement Next { get { return next; } }

    2 references
    public EenStackElement(EenStackElement stackElement)
    {
        this.next = stackElement;
    }

    2 references
    public void PrintStack()
    {
        PrintWaarde();
        if (Next != null)
        {
            Next.PrintStack();
        }
    }

    4 references
    public virtual void PrintWaarde()
    {
        Console.WriteLine("Ik heb geen data.");
    }
}
```

Class IntegerStackElement

- ▶ SubClass
 - ▶ Erft over van EenStackElement
- ▶ Bevat data
- ▶ Constructor vult data
- ▶ Base(next) roept de constructor van de superclass aan (voor het koppelen in de lijst)
- ▶ Override:
 - ▶ PrintWaarde() vervangt PrintWaarde() in de superclass
- ▶ PrintStack() wordt overgeerfd

```
internal class IntegerStackElement : EenStackElement
{
    int data;
    1 reference
    public int Data { get { return data; } }
    1 reference
    public IntegerStackElement(int Getal, EenStackElement next)
        : base(next)
    {
        data = Getal;
    }

    3 references
    public override void PrintWaarde()
    {
        Console.WriteLine(Data);
    }
}
```

Class StringStackElement

- ▶ SubClass
 - ▶ Erft over van EenStackElement
- ▶ Bevat data
- ▶ Constructor vult data
- ▶ Base(next) roept de constructor van de superclass aan (voor het koppelen in de lijst)
- ▶ Override:
 - ▶ PrintWaarde() vervangt PrintWaarde() in de superclass
- ▶ PrintStack() wordt overgeerfd

```
internal class StringStackElement:EenStackElement
{
    string data;
    1 reference
    public string Data { get { return data; } }
    1 reference
    public StringStackElement(string tekst, EenStackElement next)
        : base(next)
    {
        data = tekst;
    }
    3 references
    public override void PrintWaarde()
    {
        Console.WriteLine(Data);
    }
}
```

Class Stack

- ▶ Top is van het Type EenStackElement. Hier kun je ook alles aan koppelen wat een subclass van EenStackElement is.
- ▶ Overloading: twee Push functies
 - ▶ Parameters verschillen
- ▶ Pop Retourneert EenStackElement

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

    1 reference
    public void Push(int Getal)
    {
        EenStackElement NieuweTop = new IntegerStackElement(Getal, Top );
        Top = NieuweTop;
    }

    1 reference
    public void Push( string tekst)
    {
        EenStackElement NieuweTop = new StringStackElement(tekst, Top);
        Top = NieuweTop;
    }

    1 reference
    public EenStackElement Pop()
    {
        if (Top != null)
        {
            EenStackElement OudeTop = Top;
            Top = Top.Next;
            return OudeTop;
        }
        else return null;
    }

    1 reference
    public void Print()
    {
        Top.PrintStack();
    }
}
```


Main

- ▶ Het programma probeert de invoer te parsen (try)
- ▶ Als dat niet lukt, veroorzaakt hij een exceptie en pushed hij de invoer zonder Parse op de stack
- ▶ De stack bevat dus zowel integers als strings
- ▶ Kan ook met float, objecten, Queues...
- ▶ Kijk met de debugger wat er gebeurt

Integer op de stack

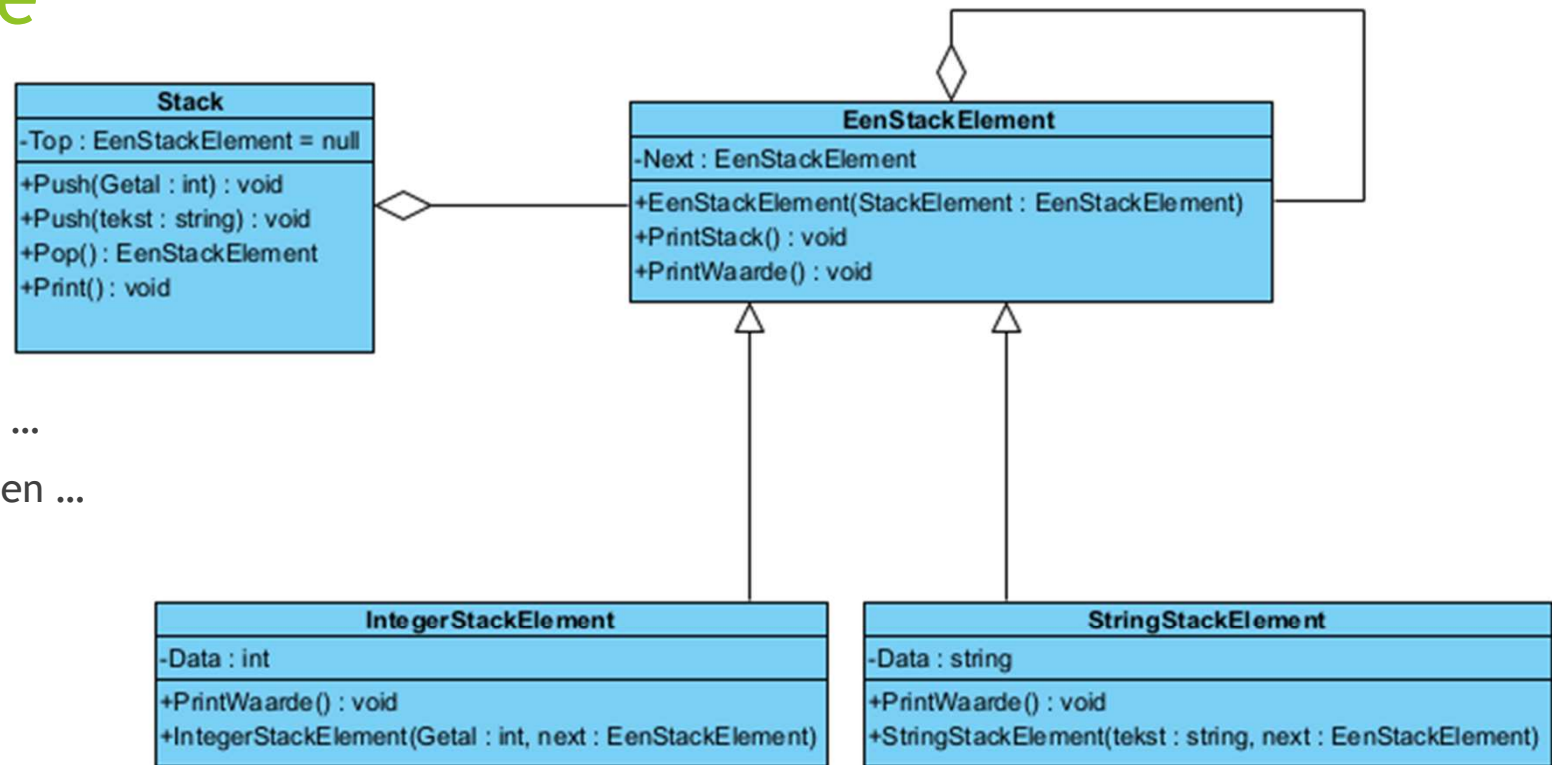
```
Geef commando:Push
Geef invoer:1
Geef commando:Push
Geef invoer:2
Geef commando:Push
Geef invoer:drie
Geef commando:Push
Geef invoer:Vier
Geef commando:Print
Vier
drie
2
1
Geef commando:Pop
Vier
Geef commando:
```

String op de stack

```
Stack MyStack = new Stack();

String Commando = "";
string Invoer = "";
while (Commando != "Stop")
{
    Console.Write("Geef commando:");
    try
    {
        Commando = Console.ReadLine();
        if (Commando == "Push")
        {
            Console.Write("Geef invoer:");
            Invoer = Console.ReadLine();
            MyStack.Push(int.Parse(Invoer));
        }
        else if (Commando == "Pop")
        {
            EenStackElement element = MyStack.Pop();
            if (element != null)
            {
                element.PrintWaarde();
            }
            else
            {
                Console.WriteLine("Stack is Leeg");
            }
        }
        else if (Commando == "Print")
        {
            MyStack.Print();
        }
    }
    catch
    {
        MyStack.Push(Invoer);
    }
}
```

UML notatie



- Oververving: Is een ...
- Aggregatie: heeft een ...

Opdracht

- ▶ Maak een sequence diagram voor het gelinkte stack programma
- ▶ Hoofdstuk 9 en 10 doorlezen
- ▶ Opgaven H10: Bankmanager 2

