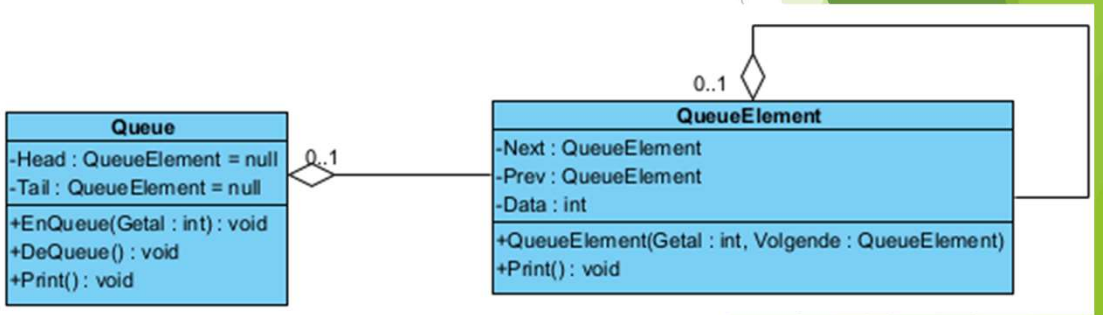
Object Oriented programming

Deze les

- UML oefenen (2 uur)
  - Class diagram
  - Sequence diagram
- Binaire boom programmeren (2 uur)

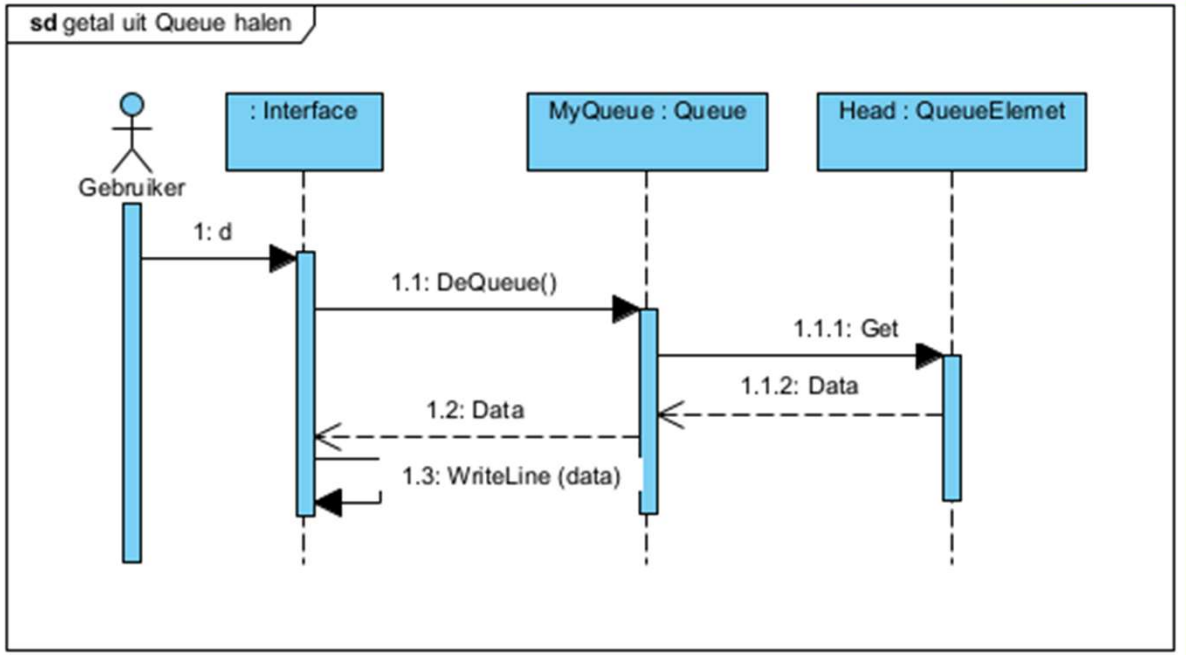
Class Diagram

- Klassen zijn datatypes
- De variabelen van het type heten objecten
- Attributen
- Methoden
- Relaties
  - Multipliciteit:
  - Een queue heeft 0 of 1 QueueElement
  - Een QueueElement heeft 0 of 1 QueueElement



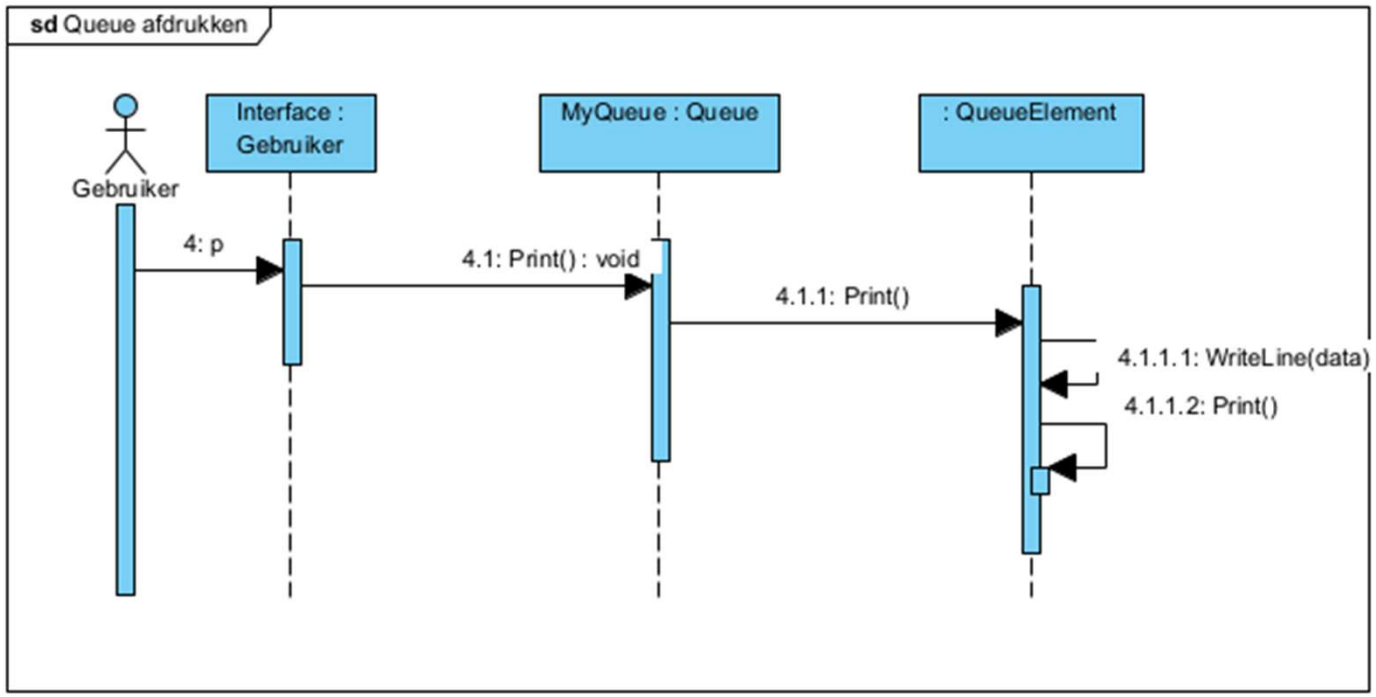
Sequence diagram

- Object(klassen) en actoren
- Geeft interactie tussen objecten weer
- Tijdlijnen
- Messages (methoden)
  - Aanroep van een methode op een ander andere object (klasse)
  - Self messages
  - Pijl met vaste lijn
- Antwoorden
  - Pijl met stippellijn



Sequence diagram

- Recursieve methoden
  - Een functie die zichzelf aanroept

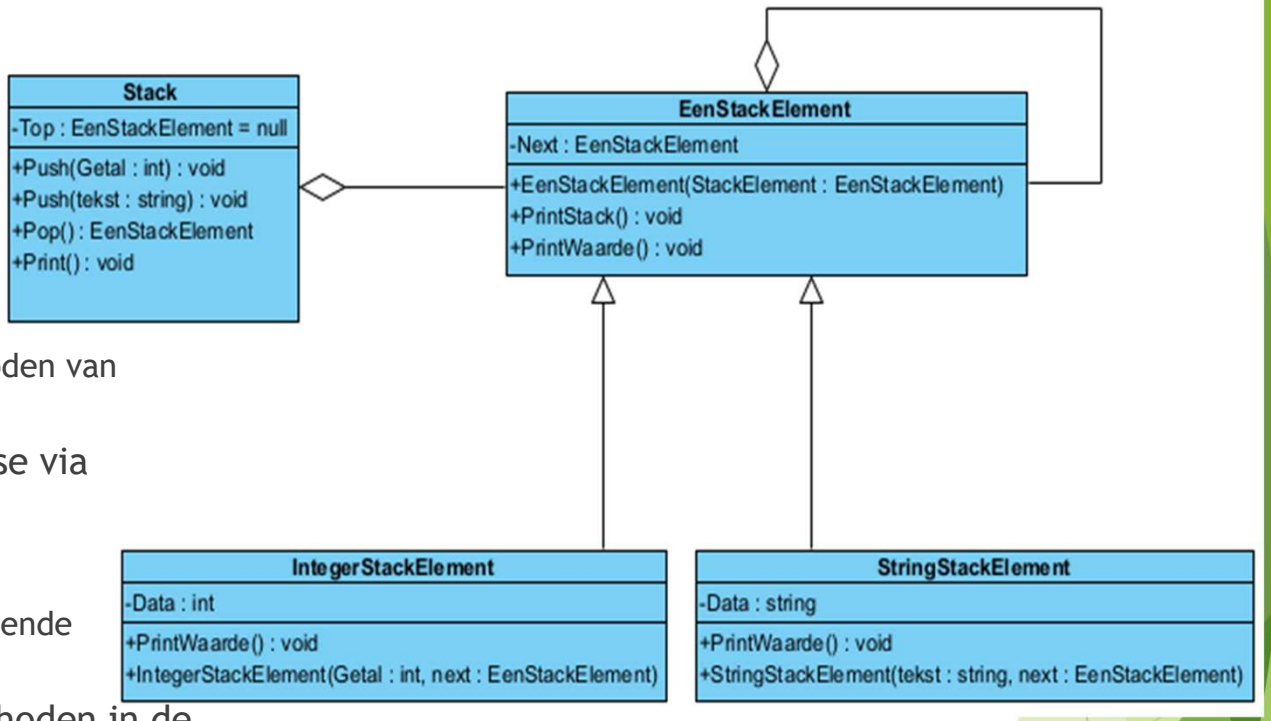


Overerving

- Een subklasse kan overerven van een andere klasse superklasse
- Een object krijgt alle methoden en attributen van zijn superklasse
- Van abstracte klassen kun je geen object maken, alleen overerven

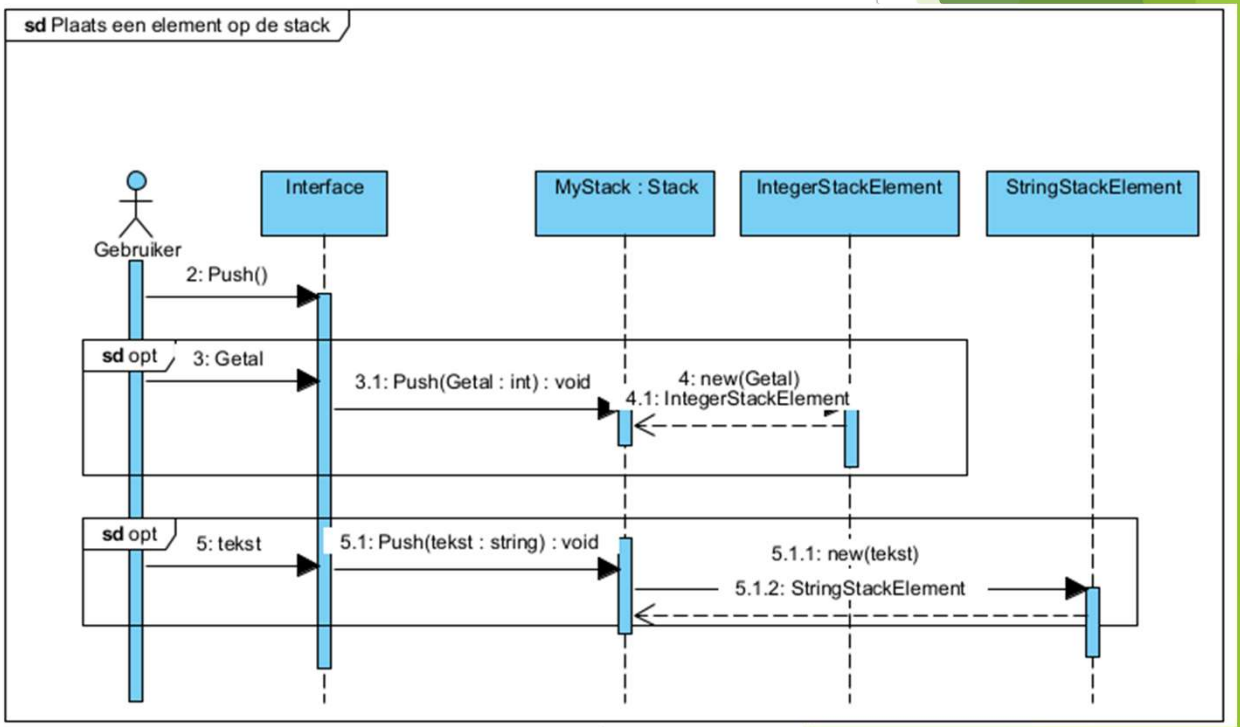
Uml Notatie

- Klassen diagram
- Geeft structuur weer
- IntegerStack
  - Heeft alle attributen en methoden van EenStackElement
- Aanroep constructor super klasse via :base()
- Polymorfie
  - Een stackelement kan verschillende gedaantes aannemen
- Override vervangt virtuele methoden in de superklasse
  - PrintWaarde()



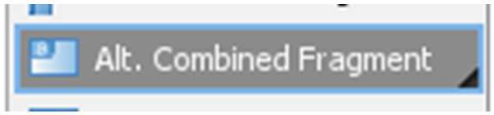
Huiswerkopdracht

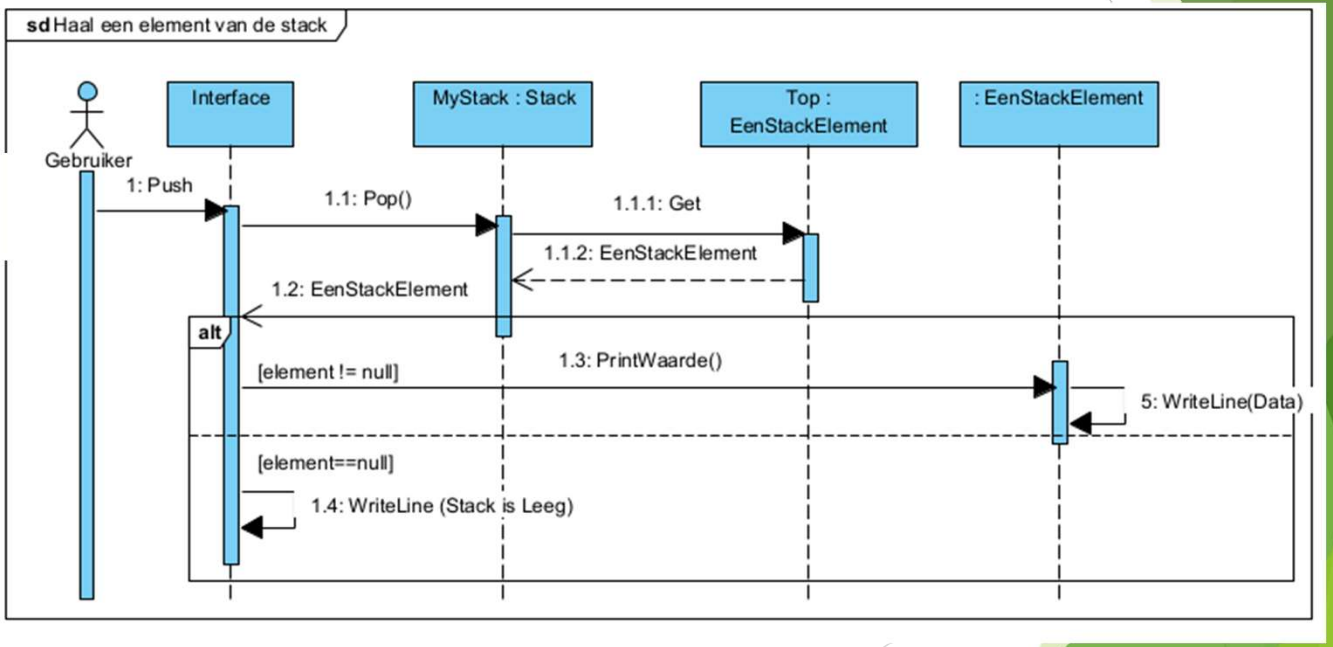
- Maak een sequence diagram voor het gelinkte stack programma
- Function Overloading
  - Push heeft 2 implementaties
- Frame opt: optioneel
  - Geen if, dat is conditioneel



Huiswerkopdracht

- Maak een sequence diagram voor het gelinkte stack programma
- Frame alt: conditie (if)
  - If (element !=null)
- Pardigm:



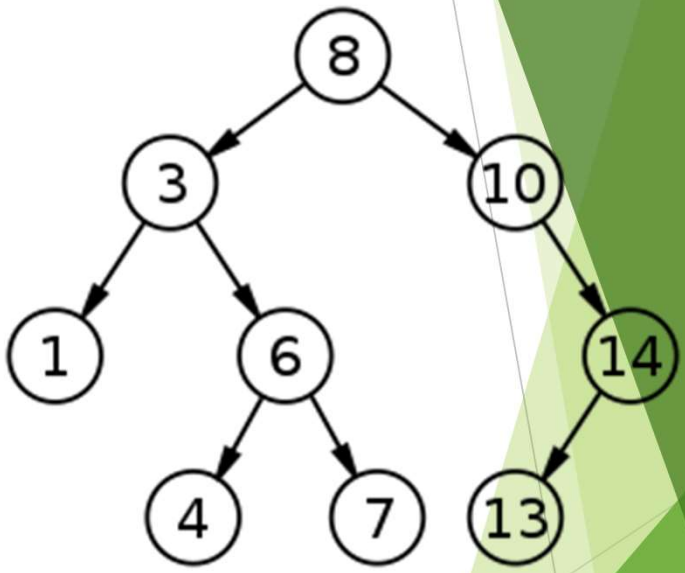


UML opdracht

- Liftcasus
  - Staat op Moodle
- Class diagram maken
- Sequence diagram(men) maken
- Werk iteratief
- Gebruik paradigm
- Mag in tweetallen
- Later presentaties

Binaire Zoekboom

- Een datastructuur met knopen
- Boomstructuur van knopen met data
- Bovenste knoop: root
- Alle waarden in de linker subboom van een knoop zijn kleiner dan of gelijk aan de waarde in de knoop
- Alle waarden in de rechter subboom van een knoop zijn kleiner dan of gelijk aan de waarde in de knoop



Binaire zoekboom

- Ontworpen om snel te kunnen zoeken
- Algoritme

```
if zoekwaarde < waarde in knoop
  then zoek in linkersubboom
else if zoekwaarde > waarde in knoop
  then zoek in rechtersubboom
else if zoekwaarde == waarde in knoop
  then return waarde
else if blad tegengekomen
  then return niet gevonden
```

Definitie Knoop

- ? Betekent dat de variabele nullable is

```
8 references
public Knoop? Links { get; set; }
8 references
public Knoop? Rechts { get; set; }
6 references
public int Data { get; }
1 reference
public Knoop(int data)
{
    Links = null;
    Rechts = null;
    Data = data;
}
```

AddKnoop

- Root: begin van de boom
- Bestaandeknoop: Knoop in de boom
  - In het begin null (root)
- Function overload

```
public void Add(int Getal)
{
    Knoop knoop = new Knoop(Getal);
    if (root == null)
    {
        root = knoop;
    }
    else
    {
        Add(knoop, root);
    }
}
```

```
internal class BinaireBoom
{
    Knoop? root = null;
    3 references
    private void Add(Knoop NieuweKnoop, Knoop BestaandeKnoop)
    {
        if (NieuweKnoop.Data < BestaandeKnoop.Data)
        {
            // ga naar links}
            if (BestaandeKnoop.Links == null)

            {
                BestaandeKnoop.Links = NieuweKnoop;
            }
            else
            {
                Add(NieuweKnoop, BestaandeKnoop.Links);
            }
        }
        else // ga naar rechts
        {
            if (BestaandeKnoop.Rechts == null)

            {
                BestaandeKnoop.Rechts = NieuweKnoop;
            }
            else
            {
                Add(NieuweKnoop, BestaandeKnoop.Rechts);
            }
        }
    }
```

Zoeken in Boom

- Weer rekening houden met een lege boc
- Weer een function overload

```
public void Zoek(int Getal)
{
    if (root == null)
    {
        Console.WriteLine("Boom is leeg");
    }
    else
    {
        Zoek(root, Getal);
    }
}
```

```
public void Zoek(Knoop BestaandeKnoop, int Getal)
{

    if (Getal == BestaandeKnoop.Data)
    {
        Console.WriteLine("Gevonden!");
    }
    else
    {
        if (Getal < BestaandeKnoop.Data)
        {
            // ga naar links}
            if (BestaandeKnoop.Links == null)

            {
                Console.WriteLine("Niet Gevonden");
            }
            else
            {
                Zoek(BestaandeKnoop.Links, Getal);
            }
        }
        else // ga naar rechts
        {
            if (BestaandeKnoop.Rechts == null)

            {
                Console.WriteLine("Niet Gevonden");
            }
            else
            {
                Zoek(BestaandeKnoop.Rechts, Getal);
            }
        }
    }
}
```

Boom verwijderen

- Supersimpel: de garbage collector doet het werk!

```
1 reference
public void Delete()
{
    root = null;
}
```

Boom afdrukken

- Een knoop kan zichzelf afdrukken
- Diepte geeft aan hoe diep hij in de boom zit
- Eerst links afdrukken
- Dan jezelf
- Tenslotte rechts afdrukken
- Afdrukken kan starten als root niet nulll is

```
public void Print()
{
    if (root != null)
    {
        root.Print(0);
    }
    else
    {
        Console.WriteLine("Boom is leeg");
    }
}
```

```
public void Print(int diepte)
{
    diepte++;
    if (Links != null)
    {
        Links.Print(diepte );
    }
    for (int i = 0;i < diepte; i++)
    {
        Console.Write("- ");
    }
    Console.WriteLine(Data);
    if (Rechts != null)
    {
        Rechts.Print(diepte);
    }
}
```

Hoofdprogramma

- Denk om Exception handling!
- Test je programma met de volgende getalreeksen en kijk hoe de boom eruit komt te zien:
  - 50 25 75 10 40 100 60
  - 50 25 75 100 10 40 60
  - 50 40 25 10 60 100
  - 10 25 40 50 60 75 100
  - 100 75 60 50 40 25 10
  - 50 25 75 10 40 100 60 50 25 75 10 40 100 60

```
BinaireBoom MijnBoom = new BinaireBoom();

String Commando = "";
while (Commando != "s")
{
    try
    {
        Console.Write("Geef commando:");
        Commando = Console.ReadLine();
        if (Commando == "a")
        {
            Console.Write("Geef getal:");
            MijnBoom.Add(int.Parse(Console.ReadLine()));
        }
        else if (Commando == "z")
        {

            Console.Write("Geef getal:");
            MijnBoom.Zoek(int.Parse(Console.ReadLine()));
        }
        else if (Commando == "p")
            MijnBoom.Print();
        else if (Commando == "d")
            MijnBoom.Delete();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
}
```

Huiswerk

- Het Queue programma kon alleen elementen van 1 kant van de queue halen. Het lift programma moet de queue langslopen, elk element opvragen en indien nodig verwijderen.
- Maak een Queue die verzoeken op kan slaan en deze functionaliteit bevat
- Bouw het liftbesturingssysteem
  - Laat de objecten met writeline statements zien wat ze doen