Object Oriented programming

Deze les

- Uitwerking liftprobleem (programmeren)
- Uitwerking stoplichten (modelleren)
- Hashtables (programmeren)
- Vliegtuig (tentamenoefening modelleren)
- Donderdag tentamenoefening programmeren

Opdracht

- Lift programma verder uitwerken
- Casus Verkeerslichten op moodle
  - Klasse diagram
  - Sequence diagram
  - STD
- Tentamenvoorbereiding!
  - Geen programmeeropdracht

Uitwerking lift

- Hoofdprogramma
- Gebruiker vraagt om een lift
  - Huidige verdieping
  - Omhoog of omlaag
- Aktiveer
  - De liften gaan omhoog en omlaag

```
static void Main(string[] args)
{
    {
        LiftenBesturing LiftBesturing = new LiftenBesturing();

        String Commando = "";
        while (Commando != "s")
        {
            try
            {
                Console.Write("Geef commando:");
                Commando = Console.ReadLine();
                if (Commando == "v")
                {
                    Console.Write("Geef je huidige verdieping (0-3):");
                    int Verdieping = int.Parse(Console.ReadLine());
                    Console.Write("Wil je omhoog (h) of omlaag (l):");
                    string Richting = Console.ReadLine();
                    if (Richting == "h") Richting = "Omhoog"; else Richting = "Omlaag";
                    LiftBesturing.RegistreerVerzoek(Verdieping, Richting);
                }
                else if (Commando == "a")
                {

                    LiftBesturing.Aktiveerliften();
                }
                else if (Commando == "n")
                    LiftBesturing.NoodStop();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Uitwerking lift

- Liftobjecten zijn private variabelen
- Registreerverzoek
  - Om en om doorsturen naar lift 1 en lift2
- Aktiveer
  - De liften gaan 1X omhoog of omlaag
  - Ieder lift mekdt hoeveel vezoeken hij nog heeft

```
internal class LiftenBesturing
{
    public const int AantalVerdiepingen = 4;
    private Lift Lift1, Lift2;
    private Lift HuidigeLift;


    1 reference
    public LiftenBesturing()
    {
        Lift1 = new Lift(1);
        Lift2 = new Lift(2);
        HuidigeLift = Lift1;
    }
    1 reference
    public void NoodStop()...
    1 reference
    public void RegistreerVerzoek(int Verdieping, string Richting)
    {
        if (HuidigeLift == Lift1)
        {
            Lift1.RegistreerVerzoek(Verdieping, Richting);
            HuidigeLift = Lift2;
        }
        else
        {
            Lift2.RegistreerVerzoek(Verdieping, Richting);
            HuidigeLift = Lift1;
        }
    }
    1 reference
    public void Aktiveerliften()
    {
        while (Lift1.AantalVerzoeken() > 0 && Lift2.AantalVerzoeken() > 0)
        {
            Lift1.ActiveerLift();
            Lift2.ActiveerLift();
        }
    }
}
```

Uitwerking Lift

- Huidige verdieping
  - Begint op 0
- Richting
  - Omhoog of omlaag
- Deurstatus
  - Open of dicht
- Verzoekelijst omhoog/omlaag
  - Index voor verdieping
  - True als iemand naar die verdieping wil

```
internal class Lift
{

    public const int AantalVerdiepingen = 4;
    private int liftID;
    public int HuidigeVerdieping;
    public string Richting;
    public string DeurStatus;
    public bool[] VerzoekenLijstOmhoog;
    public bool[] VerzoekenLijstOmlaag;

    2 references
    public Lift(int LiftID)
    {
        liftID = LiftID;
        VerzoekenLijstOmhoog = new bool[AantalVerdiepingen];
        VerzoekenLijstOmlaag = new bool[AantalVerdiepingen];
        for (int i = 0; i < AantalVerdiepingen; i++)
        {
            VerzoekenLijstOmhoog[i] = false;
            VerzoekenLijstOmlaag[i]= false;
        }
        HuidigeVerdieping = 0;
        Richting = "Omhoog";
        DeurStatus = "Open";
    }
```

Uitwerking Lift

- RegistreerVerzoek
  - Bepaal de richting
  - Zet de boolean in het juiste array op true
- Ga naar verdieping
  - Zet de huidige verdieping
  - Meld de aankomst

```
public void RegistreerVerzoek(int Verdieping, string richting)
{
    if (richting == "Omhoog")
    {
        VerzoekenLijstOmhoog[Verdieping] = true;
    }
    else
    {
        VerzoekenLijstOmlaag[Verdieping] = true;
    }
}
3 references
public void GaNaarVerdieping(int Verdieping)
{
    HuidigeVerdieping = Verdieping;
    Console.WriteLine(liftID + ": Aankomst op verdieping " + HuidigeVerdieping);
}
2 references
public int AantalVerzoeken()
{
    int Aantal = 0;
    for (int i = 0; i < AantalVerdiepingen; i++)
    {
        if (VerzoekenLijstOmhoog[i])
        {
            Aantal++;
        }
        if (VerzoekenLijstOmlaag[i])
        {
            Aantal++;

        }
    }
    return Aantal;
}
```

Uitwerking Lift

- ActiveerLift
- Handel in een keer alle verzoeken omhoog of omlaag af
  - Verwijder het verzoek
  - Ga naar de verdieping
  - Laadpassagiers
  - Wijzig de richting voor de volgende aanroep

```
public void ActiveerLift()
{
    // Ga een keer omhoog en omlaag
    if (Richting == "Omhoog")
    {
        for (int i = 0; i < AantalVerdiepingen; i++)
        {
            if (VerzoekenLijstOmhoog[i])
            {
                VerzoekenLijstOmhoog[i] = false;
                GaNaarVerdieping(i);
                LaadPassagiers();
            }
        }
            Richting = "Omlaag";
    }
    else
    {
        for (int i = AantalVerdiepingen - 1; i >= 0; i--)
        {
            if (VerzoekenLijstOmlaag[i])
            {
                VerzoekenLijstOmlaag[i] = false;
                GaNaarVerdieping(i);
                LaadPassagiers();
            }
        }
            Richting = "Omhoog";
    }
}
```

Uitwerking Lift

- Laad Passagiers
- Open de deur
  - Meld dat hij open is
- Vraag om de verdieping
  - Kan ook in een lus
- Bepaal de richting
- Registreer verzoek
- Sluit de deur
  - Meld dat hij gesloten is

```
private void LaadPassagiers()
{
    OpenDeur();
    Console.Write(liftID + ": Naar welke verdieping (0 - "
        + (AantalVerdiepingen - 1) + "? Enter voor geen:");
    string Commando = Console.ReadLine();
    if (Commando != "")
    {
        int Verdieping = int.Parse(Commando);
        if (Verdieping > HuidigeVerdieping)
        {
            RegistreerVerzoek(Verdieping, "Omhoog");
        }
        else
        if (Verdieping < HuidigeVerdieping)
        {
            RegistreerVerzoek(Verdieping, "Omlaag");
        }
    }
    SluitDeur();
}
2 references
public void OpenDeur()
{
    DeurStatus = "Open";
    Console.WriteLine(liftID + ": Deur is open");
}
2 references
public void SluitDeur()
{
    DeurStatus = "Dicht";
    Console.WriteLine(liftID + ": Deur is dicht");
}
```

Uitwerking Lift

```
Geef commando:v
Geef je huidige verdieping (0-3):0
Wil je omhoog (h) of omlaag (l):o
Geef commando:v
Geef je huidige verdieping (0-3):3
Wil je omhoog (h) of omlaag (l):l
Geef commando:a
1: Aankomst op verdieping 0
1: Deur is open
1: Naar welke verdieping (0 - 3? Enter voor geen:2
1: Deur is dicht
2: Aankomst op verdieping 3
2: Deur is open
2: Naar welke verdieping (0 - 3? Enter voor geen:1
2: Deur is dicht
2: Aankomst op verdieping 1
2: Deur is open
2: Naar welke verdieping (0 - 3? Enter voor geen:3
2: Deur is dicht
1: Aankomst op verdieping 2
1: Deur is open
1: Naar welke verdieping (0 - 3? Enter voor geen:3
1: Deur is dicht
1: Aankomst op verdieping 3
1: Deur is open
1: Naar welke verdieping (0 - 3? Enter voor geen:0
1: Deur is dicht
2: Aankomst op verdieping 3
2: Deur is open
2: Naar welke verdieping (0 - 3? Enter voor geen:
2: Deur is dicht
```

Uitwerking Lift

- Noodstop

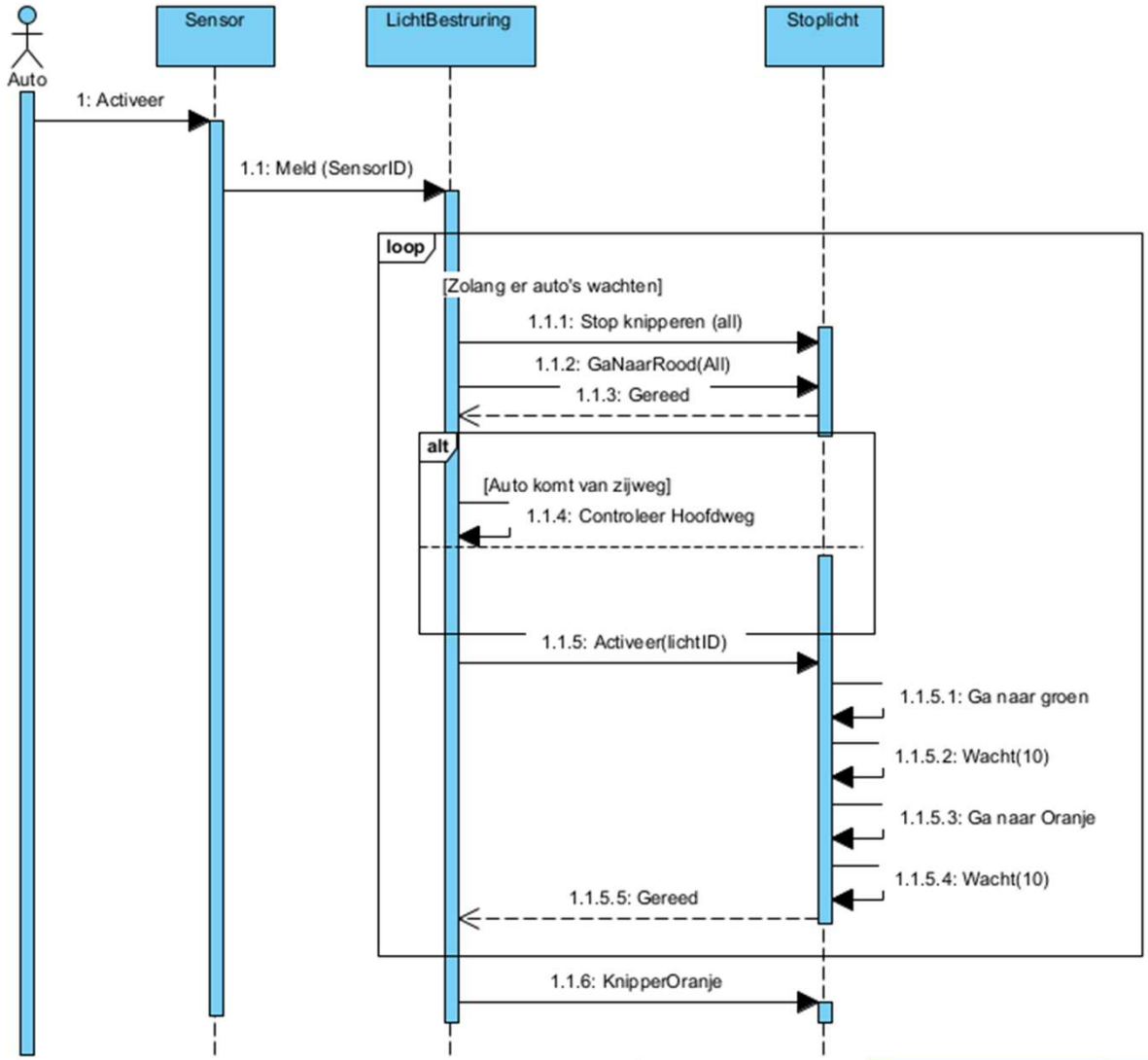
```
1 reference
public void NoodStop()
{
    Lift1.NoodStop();
    Lift2.NoodStop();
}
```

```
public void NoodStop()
{
    SluitDeur();
    GaNaarVerdieping(0);
    OpenDeur();
    MaakVerzoekenLijstenLeeg();

}
```

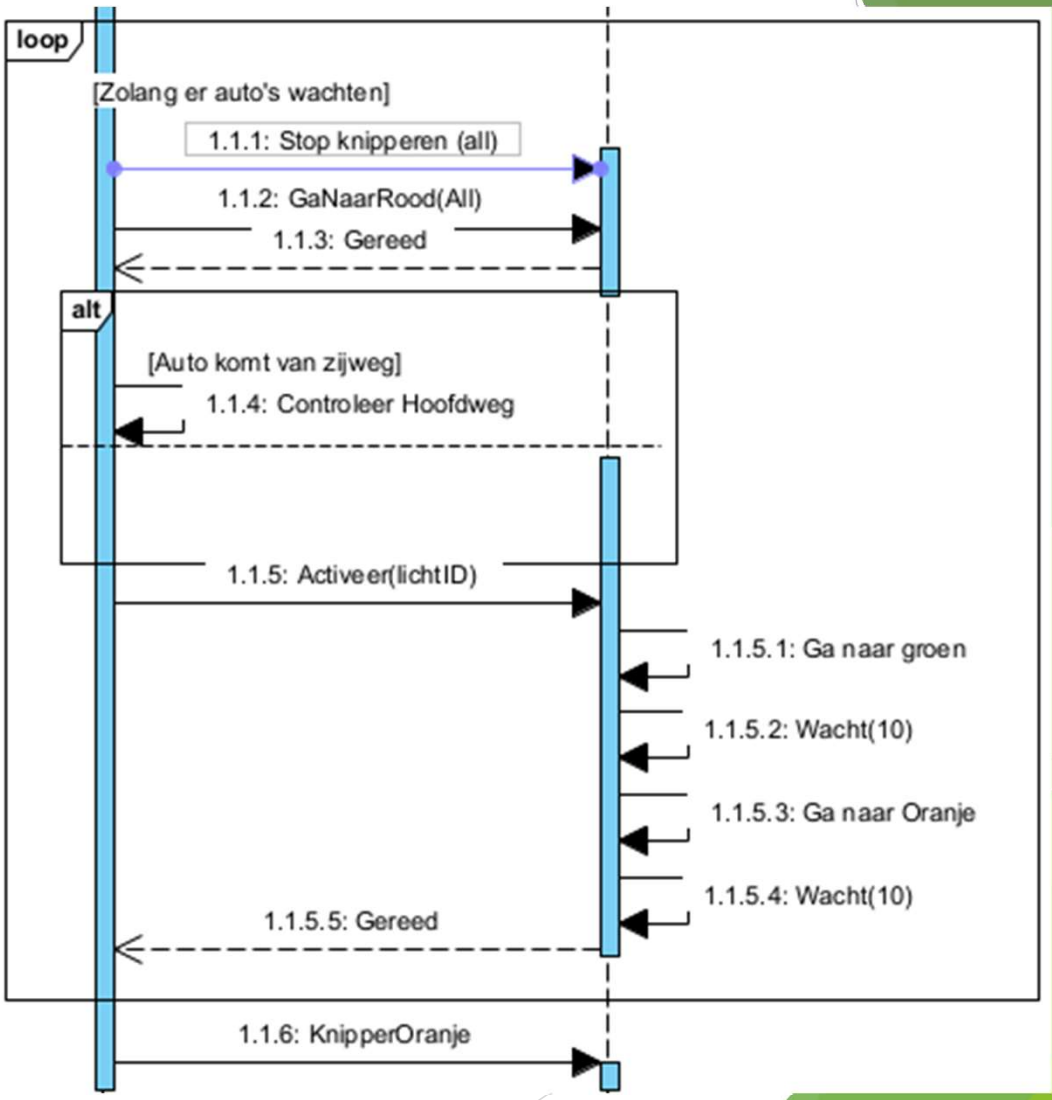
Stoplichten

- Sequence diagram
- Auto nadert
  - Hoofdweg
  - Zijweg

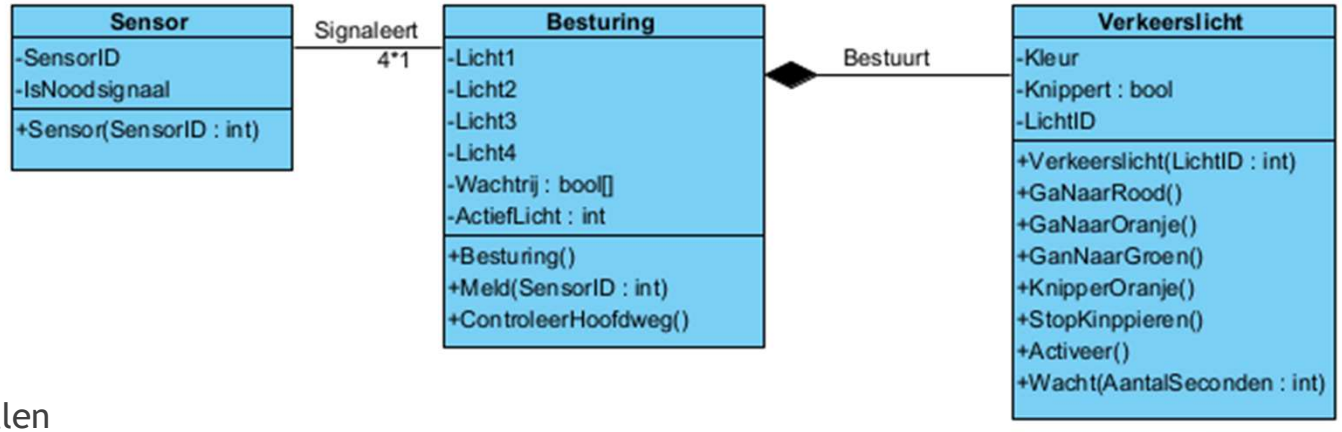


Stoplichten

- Sequence diagram
  - Detail
- Auto nadert
  - Hoofdweg
  - Zijweg



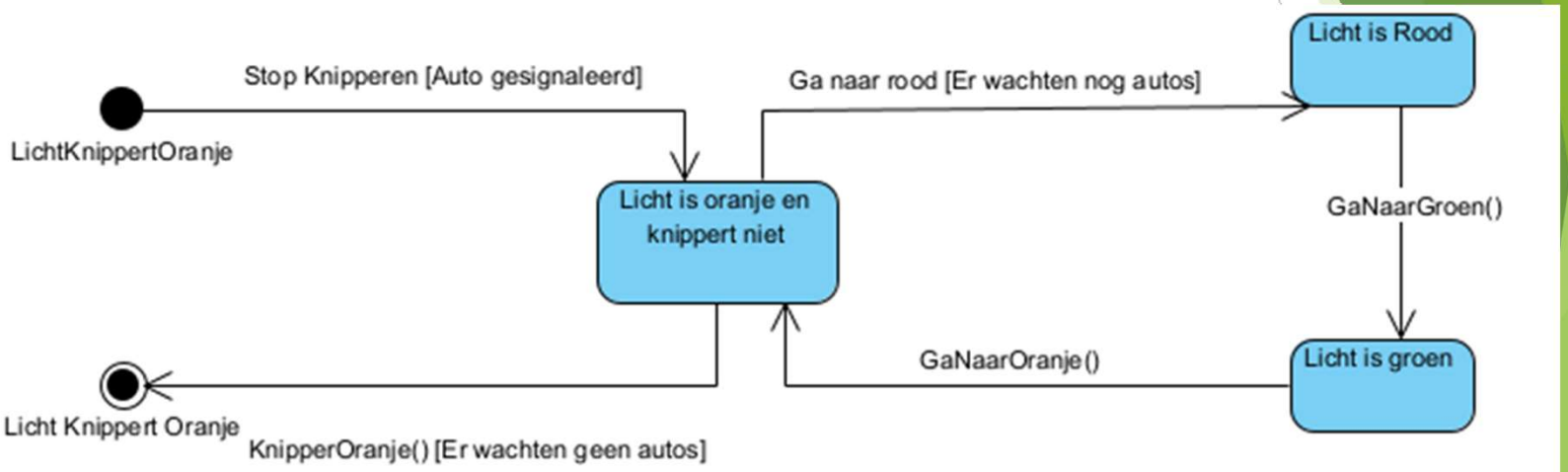
Stoplichten



- Klassediagram
- Attributen
  - Wachtrij voor de signalen
- Iedere message heeft een methode

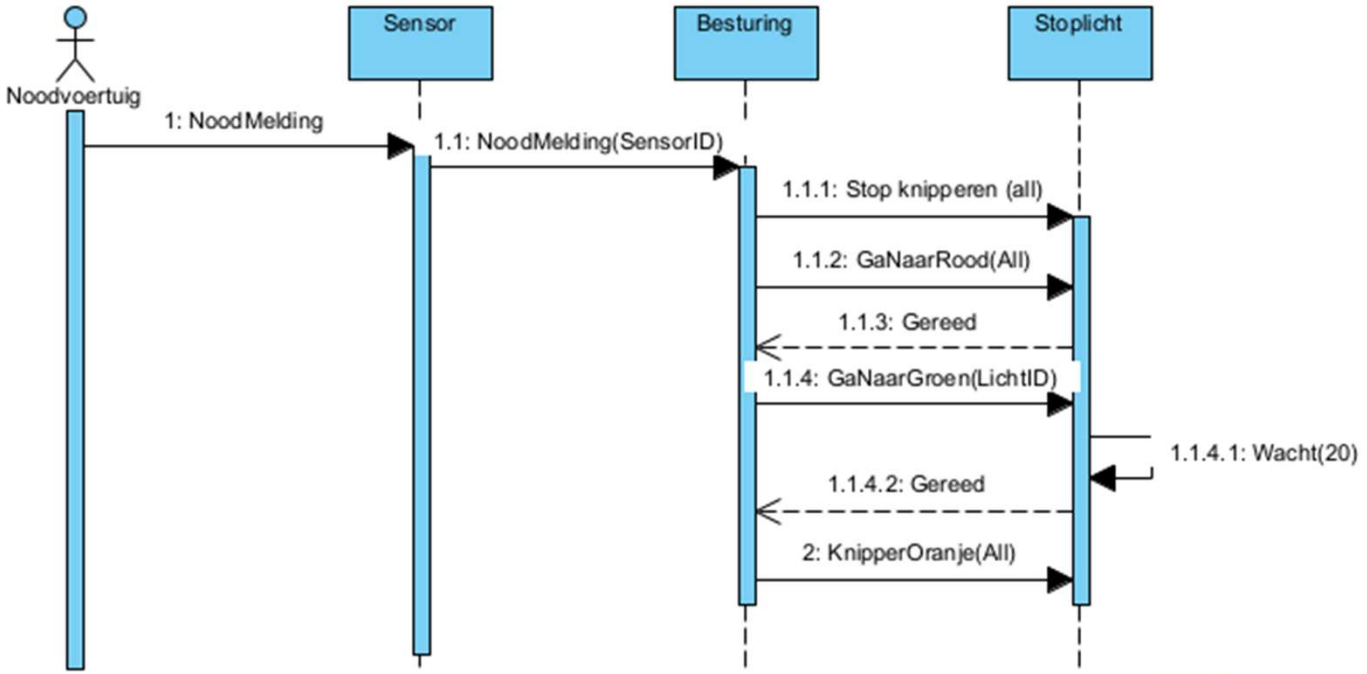
Stoplichten

- STD



Stoplichten

- Noodvoertuig



Hash table

- Erg snelle manier om iets op te zoeken
- Slaat sleutel/waarde paren op in een array
- Via een hash functie bereken je een waarde (int) van een woord
- Het woord wordt in het array opgeslagen
- De hash waarde wordt als index gebruikt
- Voorbeeld hashfuntie:
  - Tel de ordinale waarde van alle letters op
  - Bereken de rest door deling door de tabelgrootte

HashTable

- Declaratie
- Constructor
- Hash functie

```
internal class HashTable
{
    private String[] tabel;
    private int maxElementen;

    1 reference
    public HashTable(int Grootte)
    {
        maxElementen = Grootte;
        tabel = new String[maxElementen];
        for (int i = 0; i < tabel.Length; i++)
        {
            tabel[i] = null;
        }
    }
    2 references
    private int Hash(string Woord)
    {
        int HashValue = 0;
        for (int i = 0; i < Woord.Length; i++)
        {
            HashValue += ((int)Woord[i]);

        }
        return HashValue % maxElementen;
    }
```

HashTable vullen

- Wat te doen als de plek al bezet is?
- Lineair probing
  - Probeer de volgende positie
  - Begin bovenaan opnieuw als dat niets oplevert
  - Vol = vol!
- Maakt de methode minder efficiënt

```
public void Add(string Woord)
{
    int hash = Hash(Woord);
    int position = hash;

    while (position != -1)
    {
        if (tabel[position] == null)
        {
            tabel[position] = Woord;
            return;
        }
        else
        {
            position--;
        }
    }
    // nog niets gevonden, begin van boven
    position = maxElementen - 1;
    while (position != hash)
    {
        if (tabel[position] == null)
        {
            tabel[position] = Woord;
            return;
        }
        else
        {
            position--;
        }
    }
    Console.WriteLine("Hastable overflow");
}
```

HashTable zoeken

- Rekening houden met lineair probing
- Maak ook een print methode om te kunnen zien hoe de tabel gevuld is.

```
public void Print()
{
    for (int i = 0; i < tabel.Length; i++)
    {
        Console.WriteLine(i + ";" + tabel[i]);
    }
}
```

```
public void Zoek(string Woord)
{
    int hash = Hash(Woord);
    int position = hash;

    while (position != -1)
    {
        if (tabel[position] == Woord)
        {
            Console.WriteLine("Gevonden");
            return;
        }
        else
        if (tabel[position] == null)
        {
            Console.WriteLine("Niet gevonden");
            return;
        }
        else
        {
            position--;
        }
    }
    // nog niets gevonden, begin van boven
    position = maxElementen - 1;
    while (position != hash)
    {
        if (tabel[position]==Woord)
        {
            Console.WriteLine("Gevonden");
            return;
        }
        if (tabel[position] == null)
        {
            Console.WriteLine("Niet gevonden");
            return;
        }
        else
        {
            position--;
        }
    }
    Console.WriteLine("Niet gevonden");
}
```

HashTable

- Hoofdprogramma
  - A: toevoegen
  - Z: Zoeken
  - P: print
  - S: stop

```
static void Main(string[] args)
{
    {
        HashTable MijnHashTable = new HashTable(10);

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

                    Console.Write("Geef zoekterm:");
                    MijnHashTable.Zoek(Console.ReadLine());
                }
                else if (Commando == "p")
                    MijnHashTable.Print();

            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Hash tabel met binaire boom

- De knoop slaat nu strings op

```
internal class HashTable
{
    private BinaireBoom [] tabel;
    private int maxElementen;

    0 references
    public HashTable(int Grootte)
    {
        maxElementen = Grootte;
        tabel = new BinaireBoom[maxElementen];
        for (int i = 0; i < tabel.Length; i++)
        {
            tabel[i] = null;
        }
    }
    2 references
```

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

Hash tabel met binaire boom

- Maak zelf de methode *zoek*

```
public void Add(Knoop NieuweKnoop)
{
    if (String.CompareOrdinal (NieuweKnoop.Data , Data)<=0)
    {
        // ga naar links}
        if (Links == null)

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

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

Toepassing: databases (SQL Server)

- Join operatie: samenvoegen van 2 input tabellen op basis van een sleutel
  - PersoonID
- Maak tijdelijke tabellen om te groeperen
  - Tabel0, Tabel1, Tabel2
- Stap 1: Groepeer en kopieër de gegevens van de eerste tabel naar de tijdelijke tabellen

PersoonID	Naam
1	Bjarne
2	Isabel
3	Jens
4	Lars
5	Daan

PersoonID	Vak	Cijfer
1	SQL	☺
2	SQL	☺
3	SQL	☺
4	SQL	☺
5	SQL	☺

Hash match operator

- Stap 1: Groepeer en kopieer de gegevens naar de tijdelijke tabellen
- Hashfunctie: PersoonID % 3
  - 1 % 3 = 1
  - 2 % 3 = 2
  - 3 % 3 = 0
  - 4 % 3 = 1
  - 5 % 3 = 2

PersoonID	Naam
1	Bjarne
2	Isabel
3	Jens
4	Lars
5	Daan

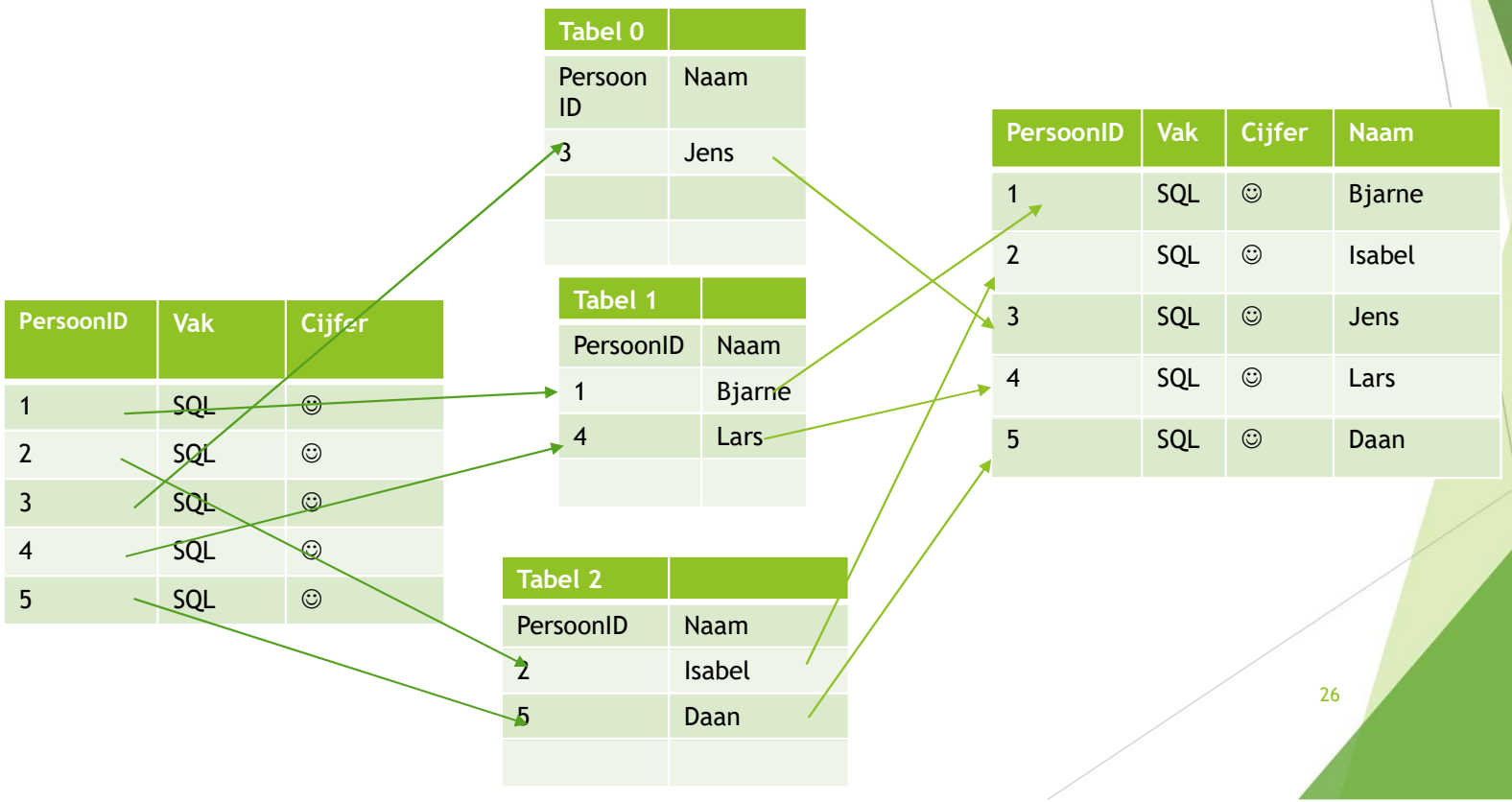
Tabel 0	
PersoonID	Naam
3	Jens

Tabel 1	
PersoonID	Naam
1	Bjarne
4	Lars

Tabel 2	
PersoonID	Naam
2	Isabel
5	Daan

Hash match operator

- Stap 2: Scan de tweede tabel
- Zoek met de hash functie in de tijdelijke tabel naar de juiste gegevens



UML modelleren

- Domeinklassen
  - Klassen die relevant zijn in de werkelijkheid van de gebruikers en domeinexperts.
- Applicatieklassen
  - Vertegenwoordigen alle aspecten van een applicatie die zichtbaar zijn voor de gebruiker. Vanuit het oogpunt van de gebruiker zijn de instanties van de applicatie klassen de applicatie. Het zijn de enige objecten waarmee hij rechtstreeks interactie heeft.
- Implementatieklassen
  - Klassen die niets te maken hebben met het probleemdomein, maar alleen met de specifieke implementatie in het huidige systeem zoals een database of eennetwerk.
- Hulpklassen
  - Algemeen bruikbare klassen, zoals generieke klassen BinaireBoom, Datum, Hashtabel, ... die in klassebibliotheken terug te vinden zijn.

Proeftentamen

- Autoverhuurbedrijf
  - Klassendiagram is gegeven
  - Maak een sequence diagram voor het reserveren van een auto
  - Maak een toestandsdiagram voor een auto (dus niet alleen voor reserveren!)
- Pinbetaling
  - Maak een klassen diagram