

# *C# FUNDAMENTALS*

ABSTRACTE KLASSEN INTERFACES



# Objectgeorianteerd Programmeren (OOP) **ABSTRACTIE**

## **Abstracte klassen en interfaces**

---



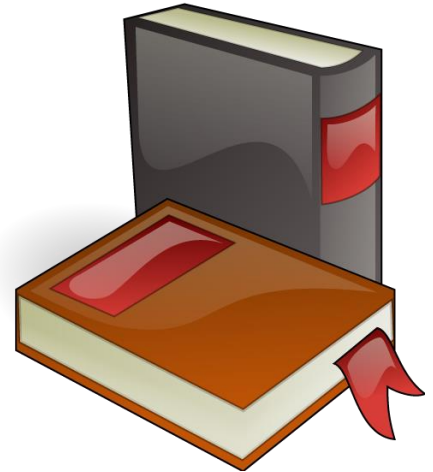
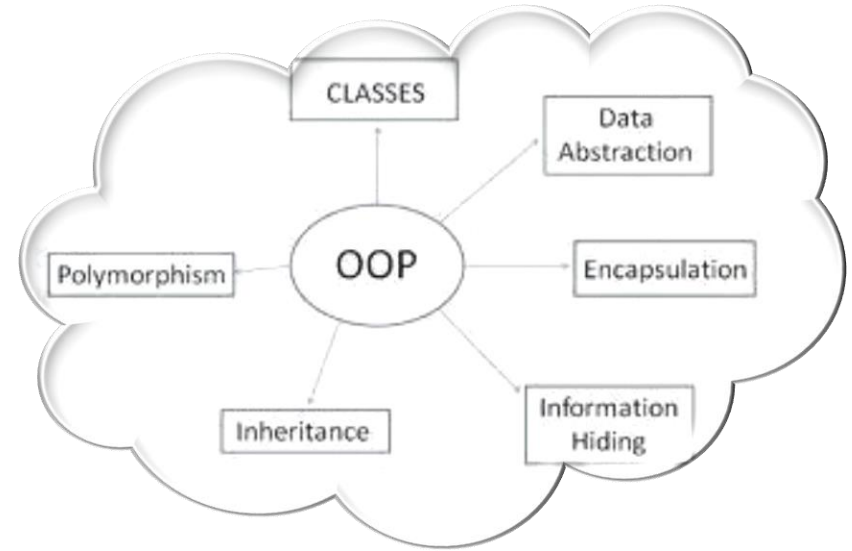
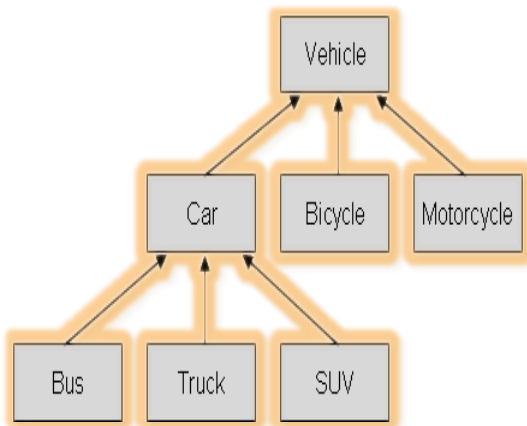
# Inhoud

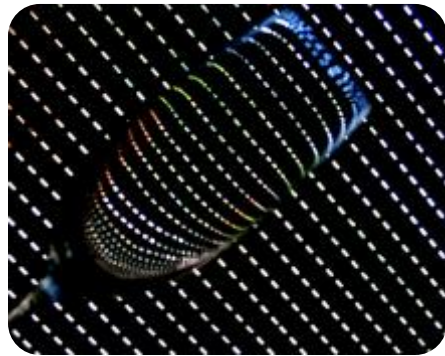
## Abstractie

- Abstracte Klassen
- Interfaces

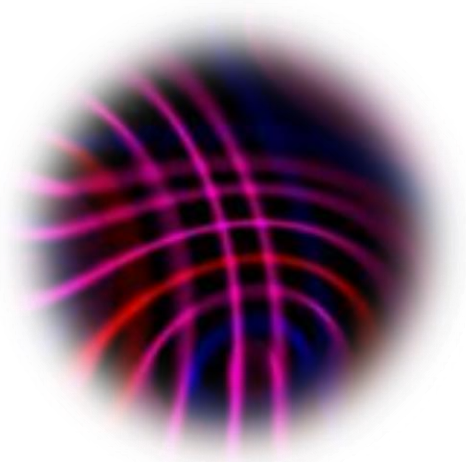
## .NET Klasse-hierarchieën

## Beste Praktijken





Abstractie



# Abstracte Methoden

- **Abstracte methoden** zijn lege methods d.w.z. zonder implementatie
  - De implementatie is opzettelijk **opzettelijk leeg gelaten** voor de afgeleide classes die verplicht zijn deze te implementeren
- Wanneer een class **ten minste 1 abstracte method** bevat, wordt deze een **abstracte klasse**
- Abstracte klassen worden gebruikt voor het modelleren van abstracte concepten
  - Bv. Persoon, Figuur,...

# Abstracte Klassen in C#

- Abstracte klassen zijn speciale klassen, gedefinieerd met het keyword **abstract**
  - Bevat 1 of meer methoden die zonder implementatie
  - Niet geïmplementeerde methoden worden **abstract** gedeclareerd en leeg gelaten
  - **Kunnen niet geïntantieerd worden**
- Afgeleide klassen moeten de abstracte methoden implementeren of ze zelf als abstract declareren

# Abstracte methode - Voorbeeld

```
abstract class Figuur  
{....  
    public abstract int BerekenOppervlakte();  
}
```

- **Abstracte methoden** zijn lege methods d.w.z. zonder implementatie
  - De implementatie is opzettelijk **opzettelijk leeg gelaten** voor de afgeleide classes die verplicht zijn deze te implementeren

# Abstracte methode – Voorbeeld(2)

```
class Rechthoek : Figuur
{
    public int Breedte { get; set; }
    public int Hoogte { get; set; }
    public override double BerekenOmtrek()
    {
        return (this.Breedte + this.Hoogte) / 2.0; ;
    }
    public override int BerekenOppervlakte()
    {
        return this.Breedte * this.Hoogte;
    }
}
```

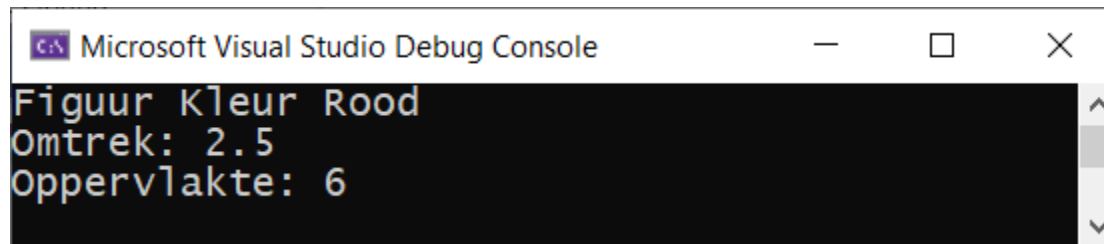


# Abstracte Klasse – Voorbeeld

```
public abstract class Figuur
{
    public string Kleur { get; set; }
    public abstract int BerekenOppervlakte(); //abstracte methode
    public abstract double BerekenOmtrek(); //abstracte methode
    public void Print()
    {
        Console.WriteLine($"Figuur Kleur {this.Kleur}");
    }
}
```

# Abstracte Klasse – Voorbeeld(3)

```
class Program
{
    static void Main()
    {
        //Figuur figuur = new Figuur(); Kan niet, abstracte klasse kan niet worden geïnstantieerd
        Rechthoek rechthoek = new Rechthoek() {Kleur = "Rood", Breedte = 2, Hoogte = 3 };
        rechthoek.Print();
        Console.WriteLine($"Omtrek: {rechthoek.BerekenOmtrek()}");
        Console.WriteLine($"Oppervlakte: {rechthoek.BerekenOppervlakte()}");
    }
}
```



The screenshot shows a window titled "Microsoft Visual Studio Debug Console". The output text is as follows:

```
Figuur Kleur Rood
Omtrek: 2.5
Oppervlakte: 6
```



# Abstracte klasse

Demo

# Oefening abstracte klasse

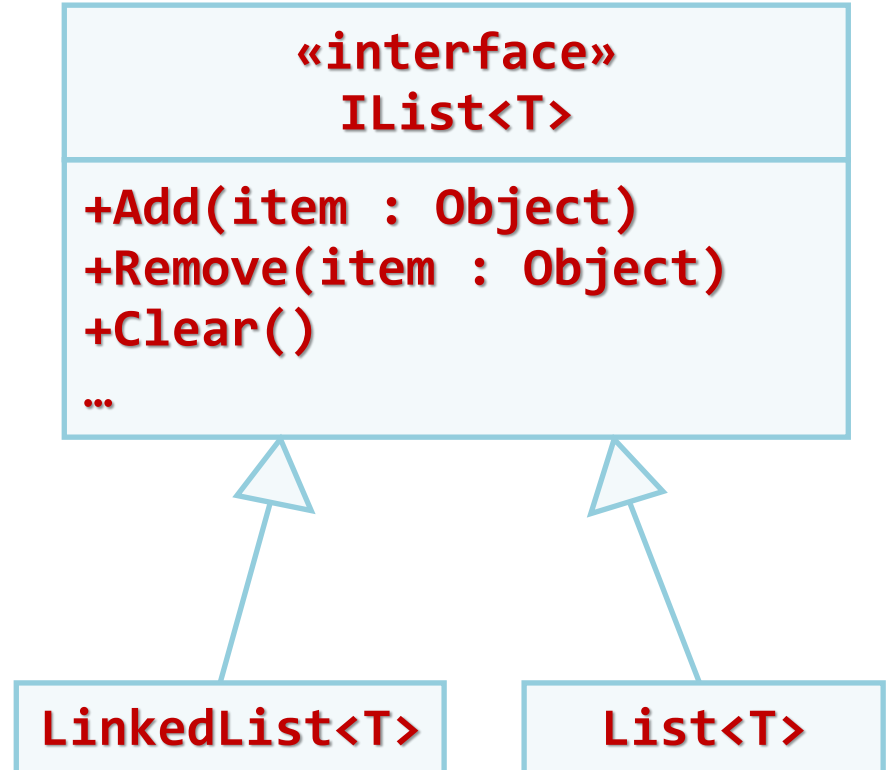
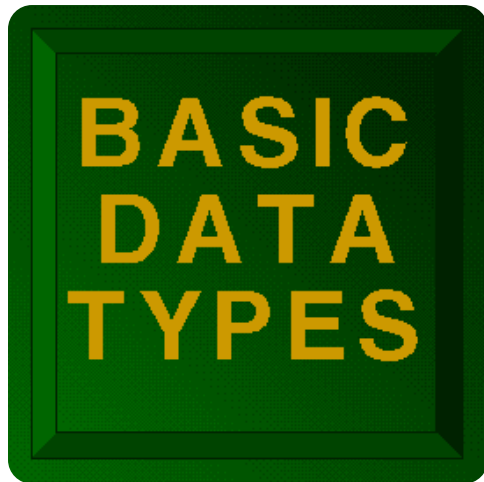
- Maak de klasse Dier abstract. Er kunnen nu geen instanties meer worden gemaakt van deze klasse
- Zet de methode Spreek() abstract in klasse Dier. Je moet dus de implementatie verwijderen
- Maak een afgeleide klasse Hond en een Kat van Dier. override de methode Spreek() in beide klassen
- Maak in Main() een object van Hond en Kat en laat ze beide spreken.



# Interfaces

# Interface IList<T>

- In .Net zijn er veel Interfaces ingebouwd
- Voorbeeld:



# Interfaces in C#

- Een **interface** definieert een verzameling van properties, methoden ,..., zoals Klassen
- In tegenstelling tot een class bevat een interface **geen implementatie**.
- Een interface kan je zien als een contract:  
Een klasse die een interface implementeert, moet ieder aspect van deze class implementeren, precies zoals in de interface is vastgelegd

# Interfaces (2)

1. een interface is anders dan een basisklasse.
2. Er kan geen rechtstreekse instantie van worden gemaakt
3. Een interface kan worden uitgebreid (extended) door andere interfaces
4. Een klasse implementeert een interface (ipv afleiden).
5. een klasse kan meerdere interfaces implementeren.
6. een interface bevat geen data declaraties, maar kan wel properties en methoden bevatten
7. alle methode declaraties in een interface zijn automatisch public.
8. een interface bevat geen implementatie-code.
9. een klasse die de interface implementeert moet voor iedere member van de interface een implementatie bevatten (of abstracte member zijn). (Daarom soms ook "contract" genoemd)



# Interfaces vs. Abstracte klassen

- C# **interfaces** verschillen van **abstracte klassen**, namelijk:
  - Interfaces kunnen geen methods bevatten met implementatie
    - Alle interface methods zijn abstract
  - Members van Interfaces hebben geen scope modifiers
    - Hun scope is altijd public
    - Scope wordt niet expliciet gespecificeerd
  - Interfaces kunnen geen fields, constants, inner types of constructors definiëren

# Interfaces – Voorbeelden

```
public interface IFiguur
{
    int BerekenOppervlakte();
    double BerekenOmtrek();
}

public interface IResizable
{
    void Resize(int weight);
    void Resize(int weightX, int weightY);
    void ResizeByX(int weightX);
    void ResizeByY(int weightY);
}
```

# Voorbeeld Interface Implementaties (2)

```
class Rechthoek : IFiguur
{
    public int Breedte { get; set; }
    public int Hoogte { get; set; }
    public double BerekenOmtrek()
    {
        return (this.Breedte + this.Hoogte)/2.0; ;
    }
    public int BerekenOppervlakte()
    {
        return this.Breedte * this.Hoogte;
    }
}
```

# Voorbeeld Interface Implementaties (3)

```
class ResizableRechthoek : IFiguur, IResizable
{
...
    public double BerekenOmtrek() {...return ..;}
    public int BerekenOppervlakte() {...return...;}
    public void Resize(int weight) {...}
    public void Resize(int weightX, int weightY) {...}
    public void ResizeByX(int weightX) {...}
        public void ResizeByY(int weightY) {...}
}

class Program
{
    static void Main()
    {
        IFiguur figuur = new Rechthoek();
        Rechthoek rechthoek = new Rechthoek() { Breedte = 2, Hoogte = 3 };
        Console.WriteLine($"Omtrek: {rechthoek.BerekenOmtrek()}");
        Console.WriteLine($"Oppervlakte: {rechthoek.BerekenOppervlakte()}");
    }
}
```

- Klassen kunnen 1 of meer interfaces implementeren
- Implementerende klassen moeten:
  - alle interface members implementeren
  - Of moeten abstract worden gedeclareerd

# Interface definitie

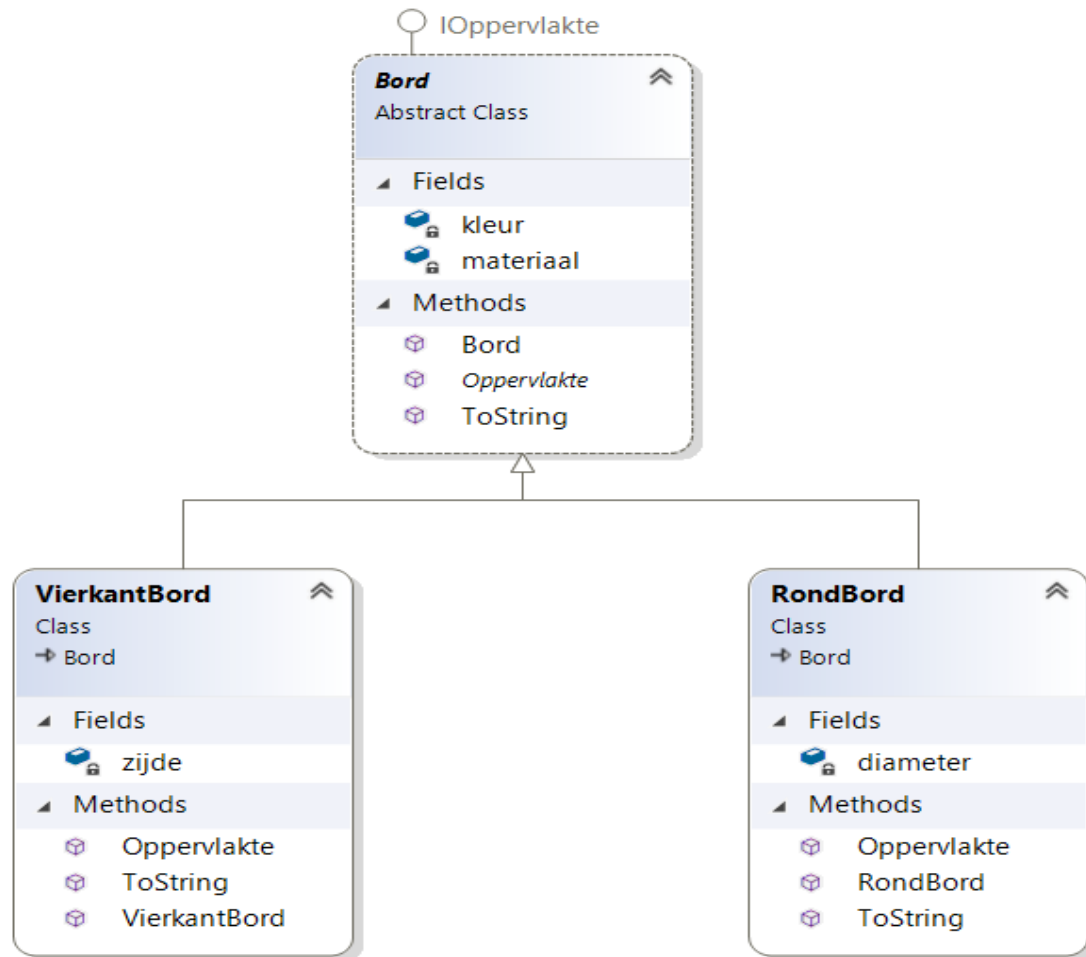
## Implementatie

Demo



# Oefening Interface (1)

Download de oefening [Bord\\_Interface \(Interface en Abstract\)\\_OEF-20-11-2020.pdf](#) van [GitHub CSharpSyntraWest/LESSEN\\_PROGCSHARP1/SLIDES](#)

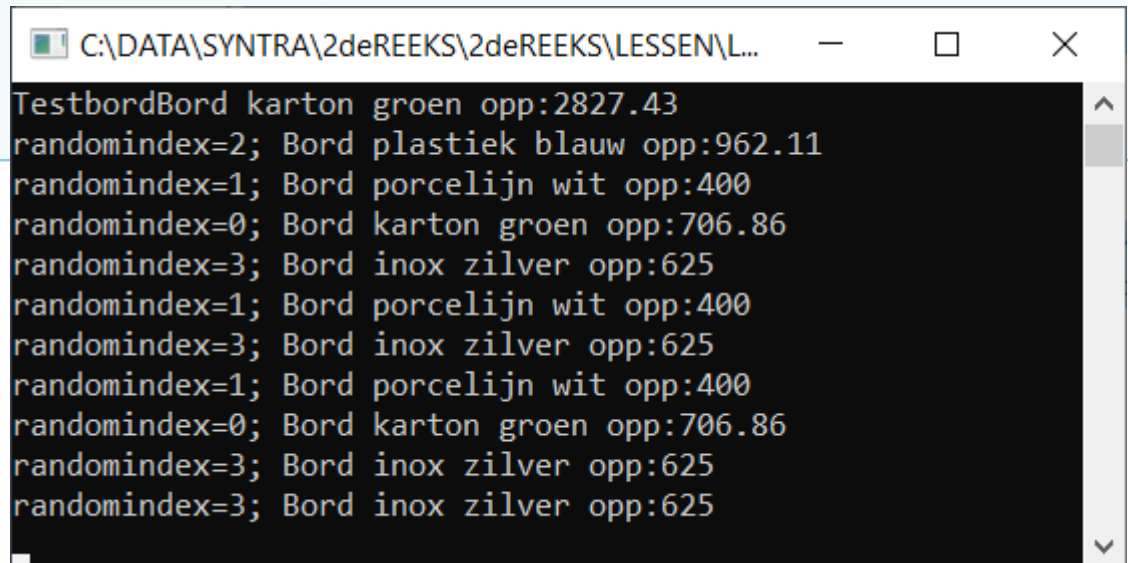


# Oefening abstract – Interface(2)

```
static void Main()
{
    Random random = new Random();
    Bord[] borden =
    {
        new RondBord ("karton", "groen",30),
        new VierkantBord ("porcelijn","wit",20),
        new RondBord ("plastiek","blauw",35),
        new VierkantBord ("inox","zilver",25)
    };
    RondBord rond = new RondBord("karton", "groen", 60);
    Console.WriteLine("Testbord" + rond);

    for (int i = 1; i <= 10; i++)
    {
        int bordindex = random.Next(4);
        Console.Write("randomindex=" + bordindex + "; ");
        Console.WriteLine(borden[bordindex].ToString());
    }

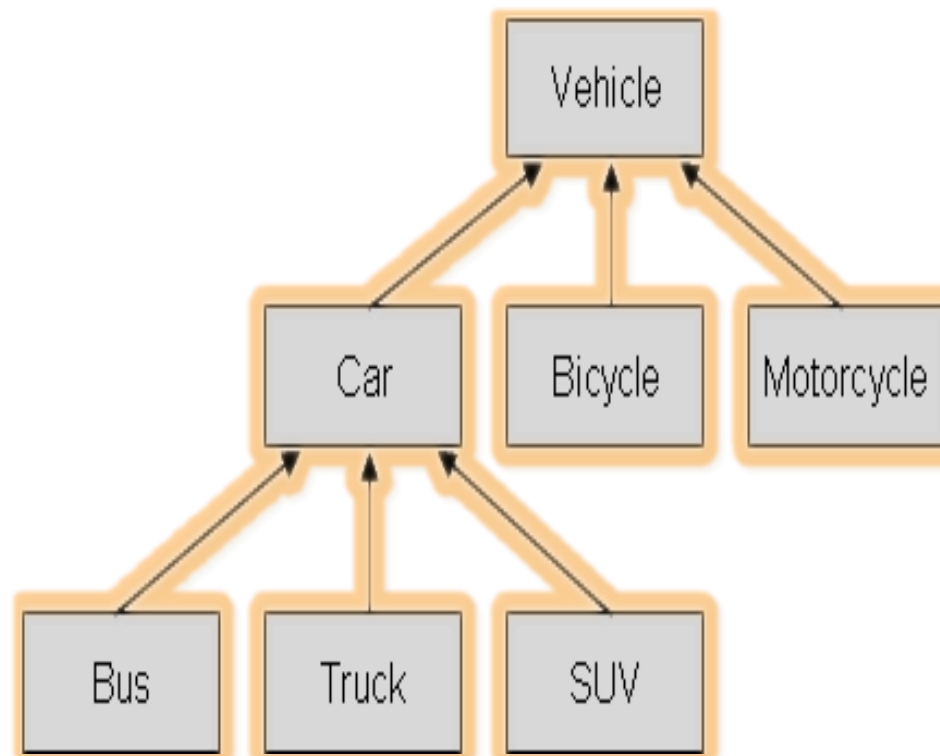
    Console.ReadKey();
}
```



```
C:\DATA\SYNTRA\2deREEKS\2deREEKS\LESSEN\L...
TestbordBord karton groen opp:2827.43
randomindex=2; Bord plastiek blauw opp:962.11
randomindex=1; Bord porcelijn wit opp:400
randomindex=0; Bord karton groen opp:706.86
randomindex=3; Bord inox zilver opp:625
randomindex=1; Bord porcelijn wit opp:400
randomindex=3; Bord inox zilver opp:625
randomindex=1; Bord porcelijn wit opp:400
randomindex=0; Bord karton groen opp:706.86
randomindex=3; Bord inox zilver opp:625
randomindex=3; Bord inox zilver opp:625
```

# Overervings-hiërarchieën

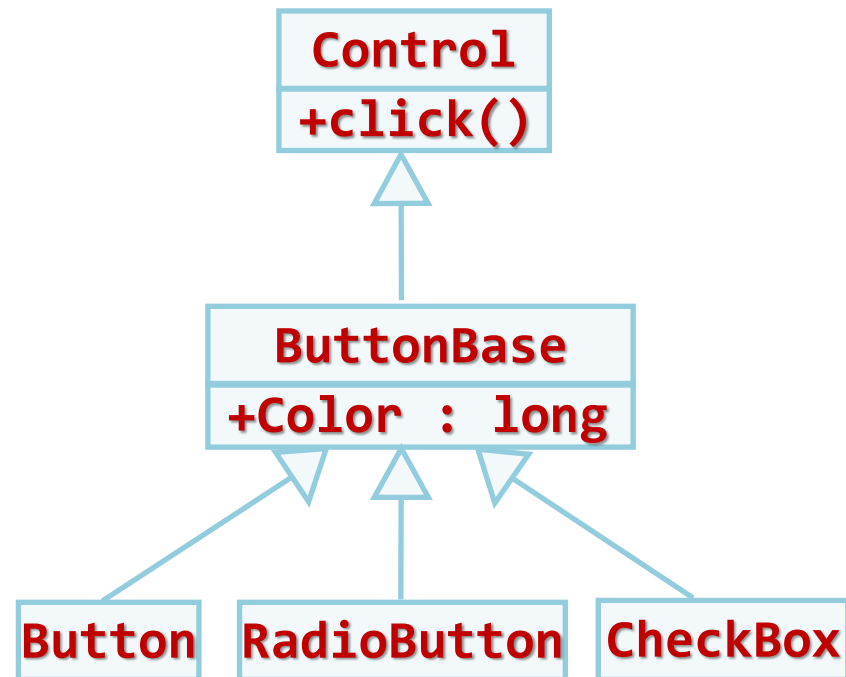
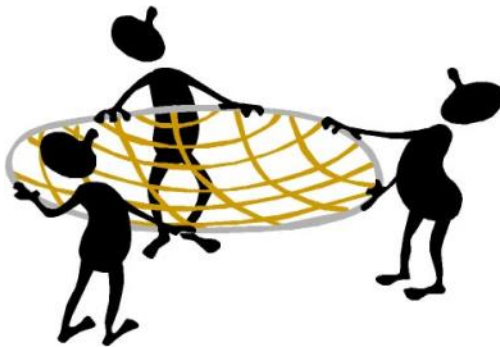
## Real-World Voorbeeld



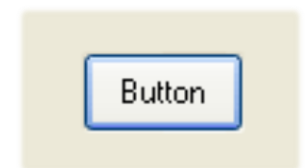
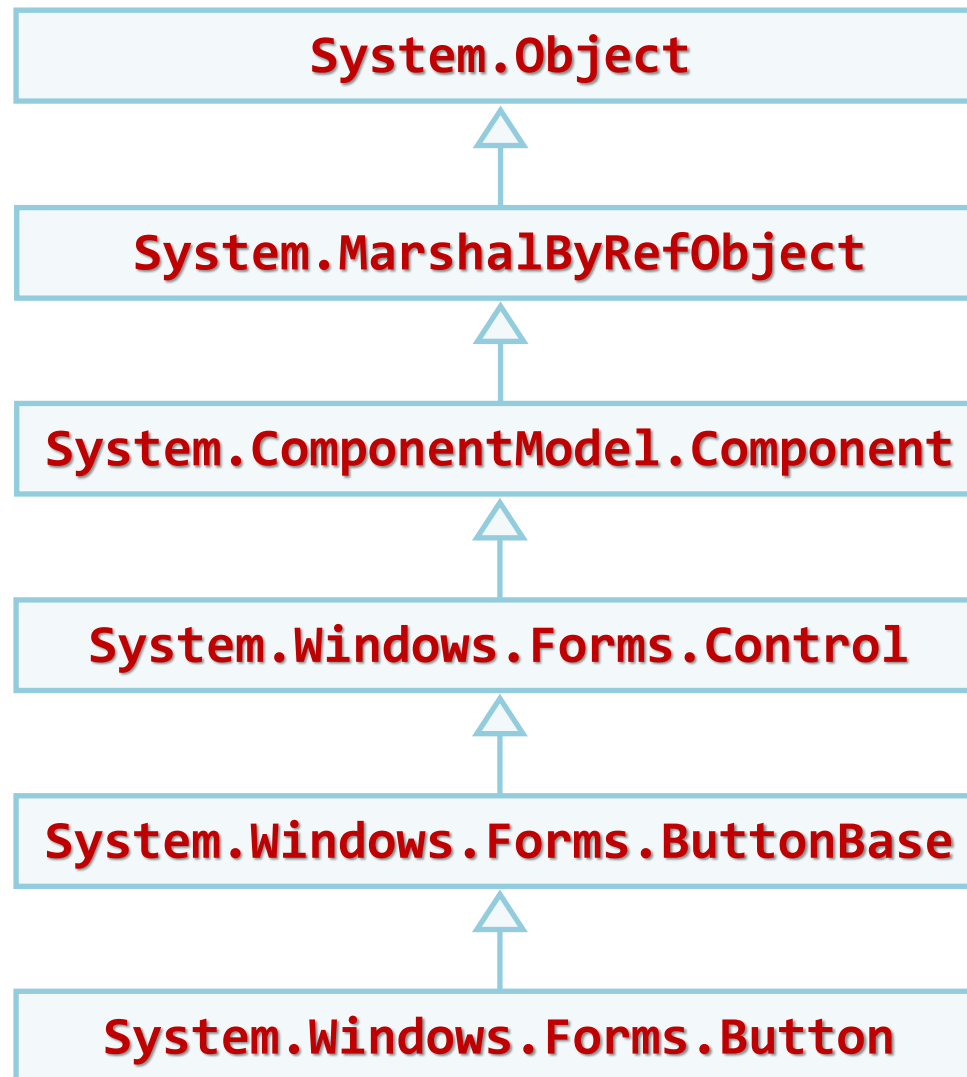


# Klassehiërarchieën in .NET

- In .NET ingebouwde klassen bestaan:
  - Abstracte klassen
  - Interfaces
  - Overerving

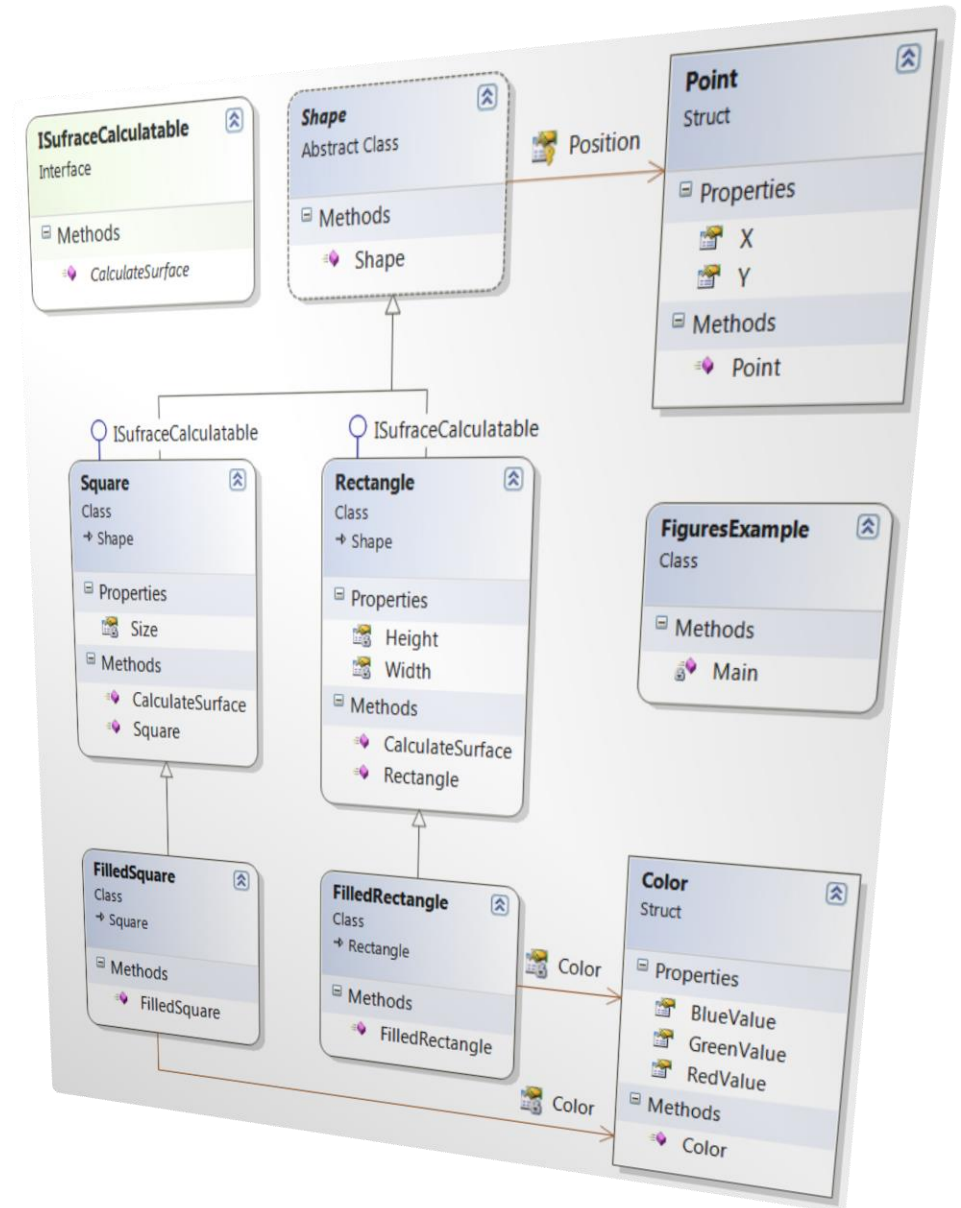


# Klassen .NET – Voorbeeld WinForms



# Klasse Diagrammen in Visual Studio

Demo

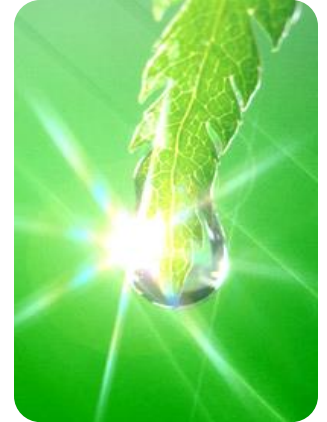


# Samenvatting

## Abstractie

- Abstracte Klassen
- Interfaces

.Net klasse-hierarchieën



## REFERENTIES

PRO C# 7 WITH .NET AND .NET CORE – ANDREW  
TROELSEN – PHILIP JAPIKSE

[HTTPS://WWW.LEARNCS.ORG/](https://www.learncs.org/)

FUNDAMENTALS OF COMPUTER PROGRAMMING WITH C#  
© SVETLIN NAKOV & CO