leren. durven. doen.

SYNTRA WEST

C# FUNDAMENTALS

LES - OOP OVERERVING (INHERITANCE)

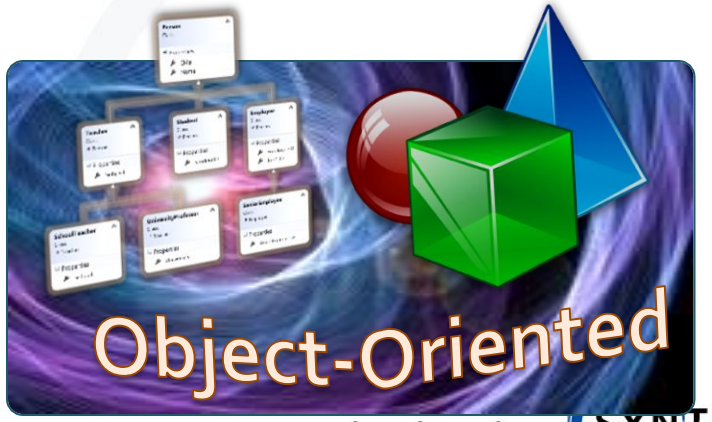
Vorige Les

Classes in C#

- Declaratie van Class?
- Access Modifiers
- Class Members
- Fields
- Constructors
- Properties
- Methods

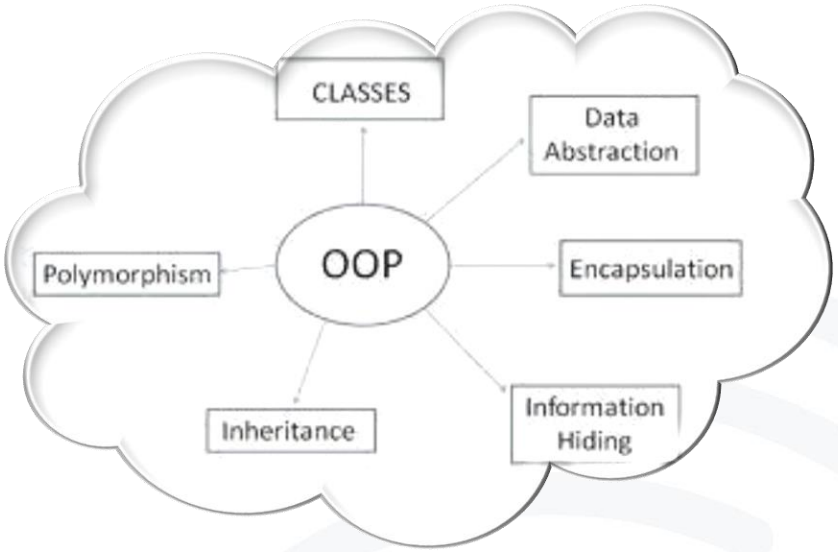
Object-Oriented Programming (OOP) INHERITANCE

Abstraction, Encapsulation, Polymorphism, Inheritance



Vandaag - Inhoud

Inheritance (overerving)

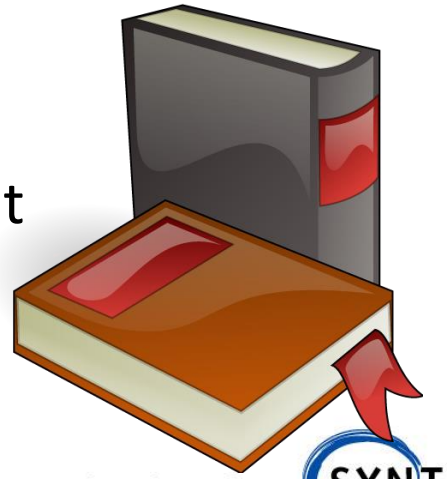


- Wat is Overerving
- Inheritance – Voordelen
- Overervingstypes
- Classes versus Interfaces
- Hierarchieën van Classes
- Access Levels

  Inheritance en Toegankelijkheid

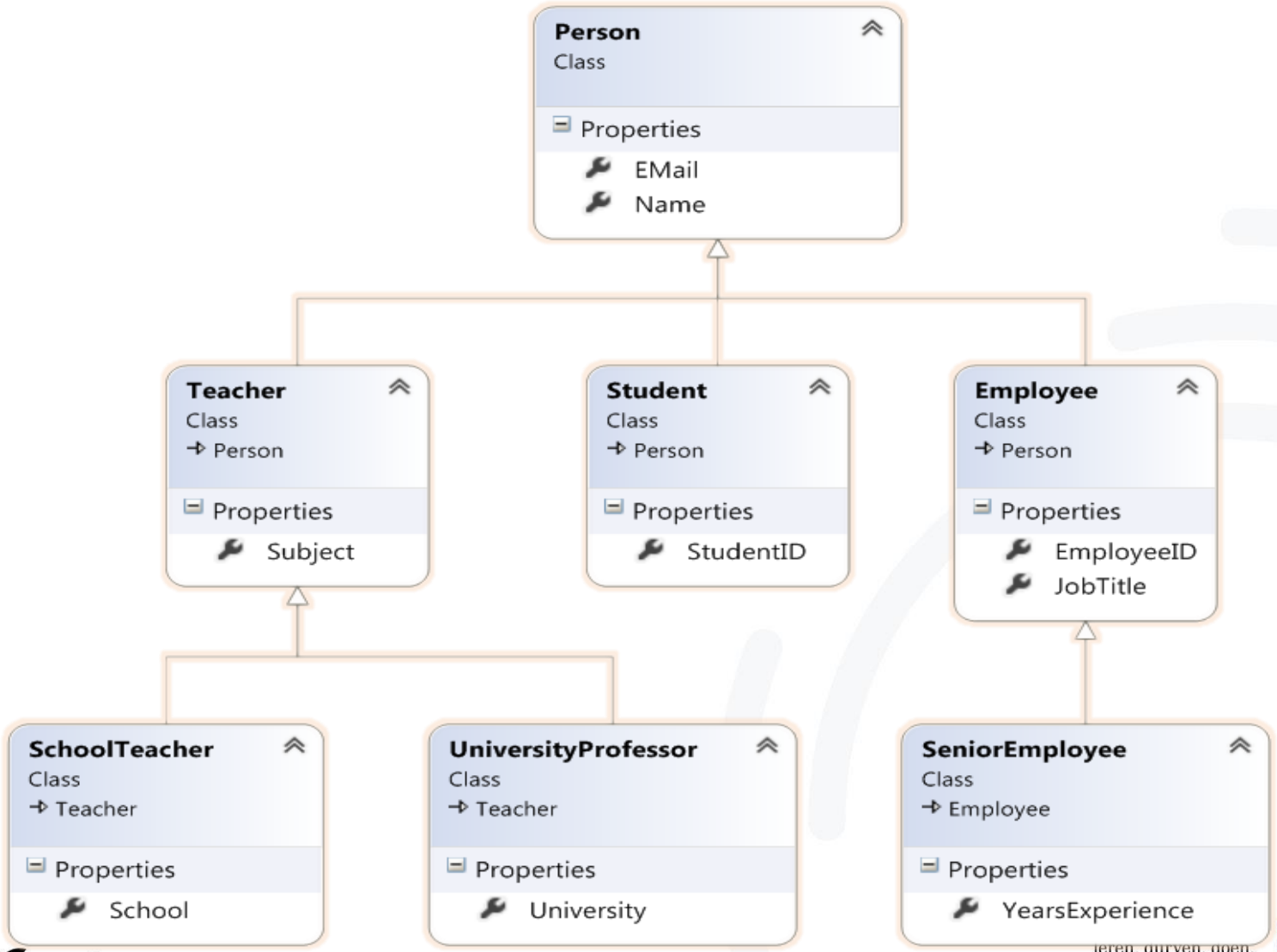
  Inheritance: Belangrijke Aspecten

  Inheritance: Belangrijke Functionaliteit



leren. durven. doen. SYNTRA WEST

Inheritance

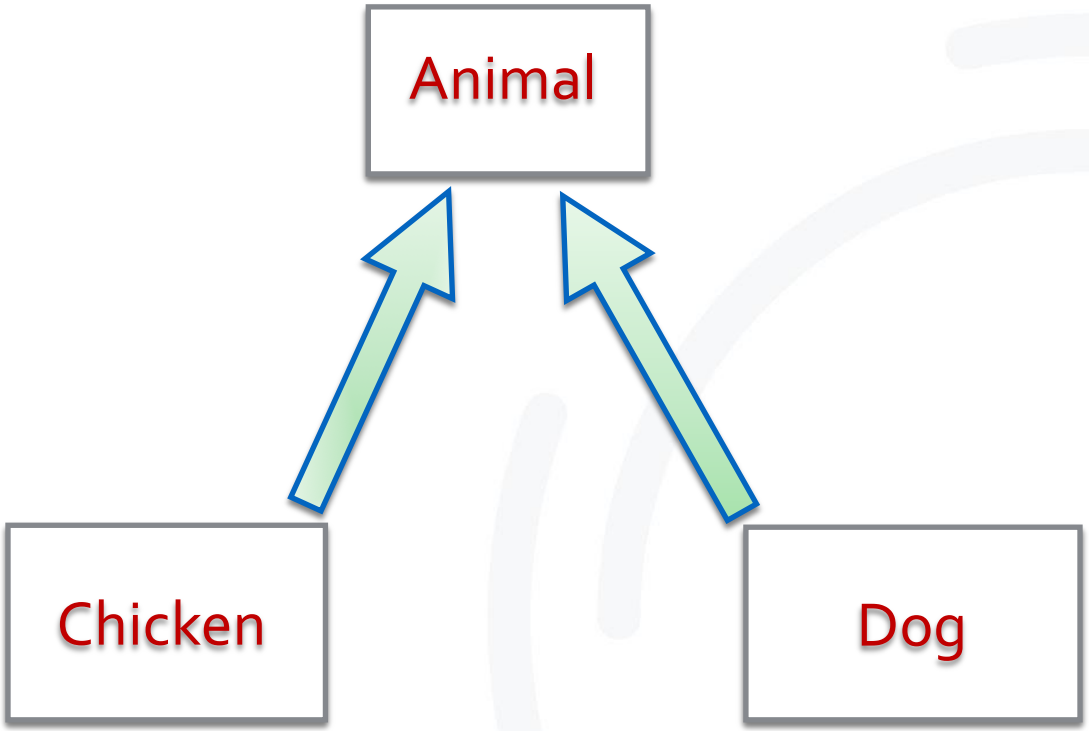


Wat is Overerving (Inheritance)

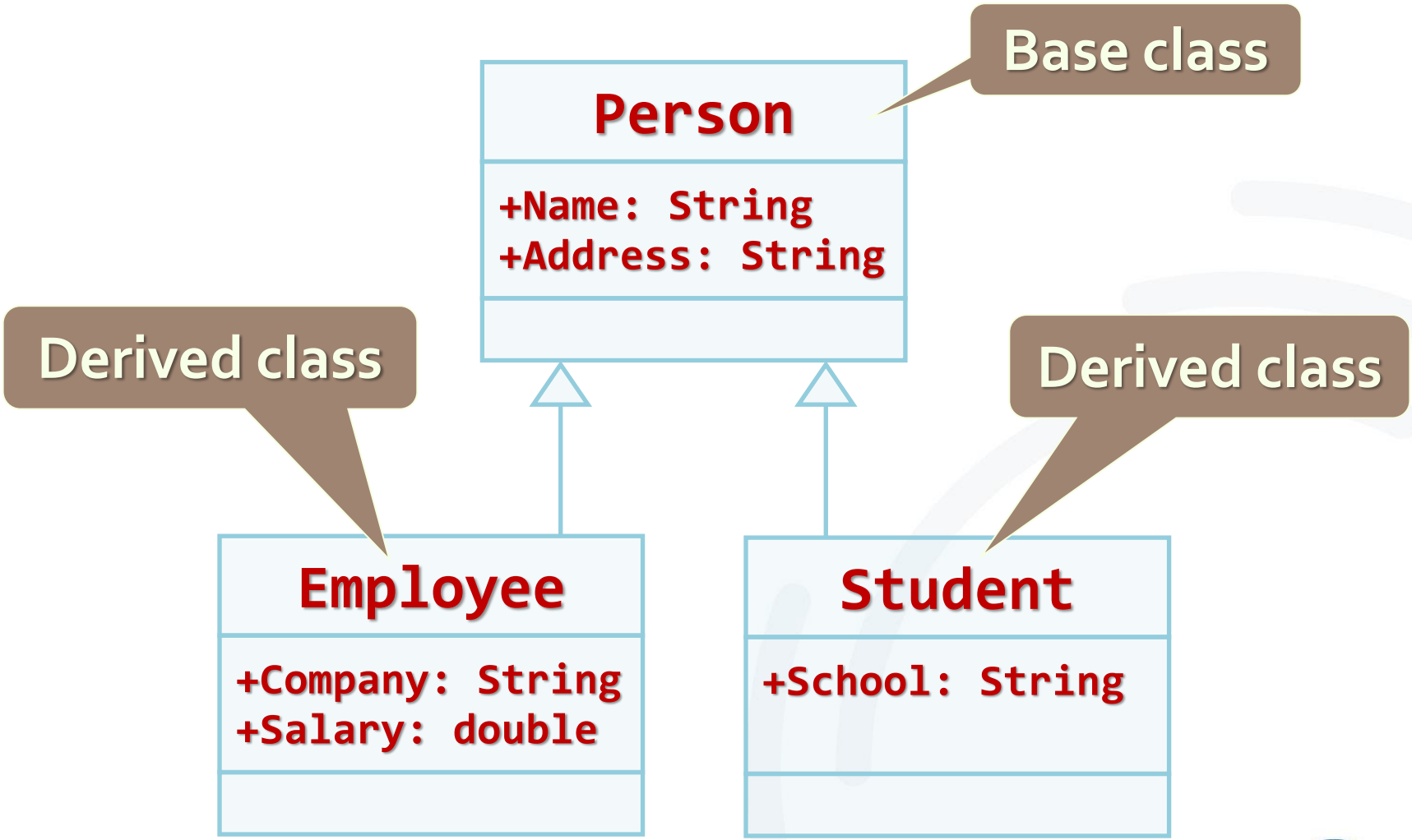
- **Bij Inheritance worden alle members van de parent class impliciet overgenomen**
  Alle fields, methods, properties, …
  Sommige members kunnen verborgen (hidden) worden
- **De class waarvan wordt overgeërfd is de base (parent) class (bv. Class Animal)**
- **De class die overerft is derived (child) class (bv. Class Cat)**

Inheritance – Voorbeeld Dieren

Zowel kip als hond zijn dieren
en erven m.a.w. alle eigenschappen en gedrag van type Dier



Inheritance – Voorbeeld Personen



Inheritance – Voordelen

- **Inheritance heeft meerdere voordelen**
  Uitbreidbaarheid
  Herbruikbaarheid (hergebruik van code)
  Voorzien van abstractie
  Overbodige code vermijden
- **Gebruik inheritance voor “is-a” relatie**
  Bv. dog “is-a” animal (honden zijn dieren)
- **<u>Niet</u> gebruiken voor “has-a” relatie**
  Bv. dog “has-a” name (hond is <u>geen</u> naam)

Inheritance in .NET

- **Een class kan van slechts 1 base class overerven**

  Bv. Class *Cat* wordt afgeleid (derived) van Class *Animal*

- **Een class/interface kan meerdere interfaces implementeren**

  Dit is de manier van .NET om soort van multiple inheritance te voorzien

Class Inheritance in .Net

- **Specifieer de naam van de base class achter de naam van de afgeleide (child) class (met : ertussen)**

```
public class Shape
{ … }
public class Circle : Shape
{ … }
```

- **Gebruik indien nodig het keyword base om de constructor van de parent aan te roepen**

```
public Circle (int x, int y) : base(x)
{ … }
```

Class Inheritance - Voorbeeld

Begin of class definition

```
public class Cat : Animal {
private string name;
private string owner;

public Cat(string name, string owner)
{
    this.name = name;
    this.owner = owner;
}

public string Name
{
    get { return this.name; }
    set { this.name = value; }
}
```

Inherited (base) class

Fields

Constructor

Property

Class Inheritance - Voorbeeld (vervolg)

```
 public string Owner
    {
        get { return this.owner; }
        set { this.owner = value; }
    }

    public void Speak()
    {
        Console.WriteLine("Miauuuuuuu!");
    }
}
```

Method

End of class definition

Overervingstypes

terminologie

derived class	inherits	base class / parent class
class	implements	interface
derived interface	extends	base interface

Overervingstypes

- **Inheritance laat toe dat child classes kenmerken van parent (base) class kunnen overnemen**
  - Attributes (fields en properties)
  - Operations (methods)
- **Child class kan een uitbreiding geven op parent class**
  - nieuwe fields en methods toevoegen
  - Herdefiniëren van bestaande methods (wijzigen van bestaand gedrag (behavior))
- **Een class kan een interface implementeren door een implementatie (code) te voorzien voor alle methods die vermeld worden in de interface**

Classes versus Interfaces

- **Classes definiëren eigenschappen (attributes) en gedrag (behavior)**
  Fields, properties, methods,.. zijn attributes
  Methods bevatten implementatie van gedrag

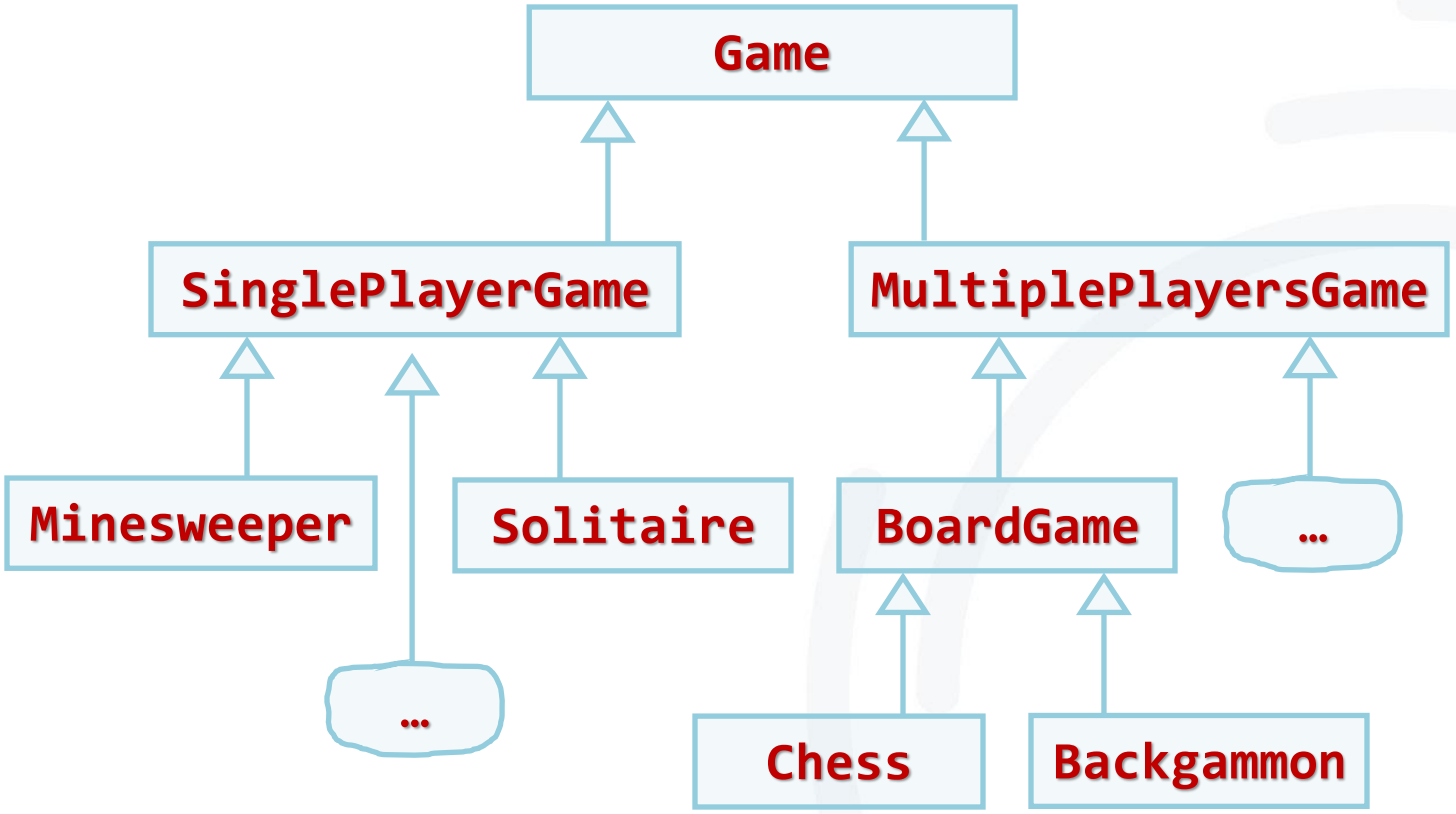
```
public class Labyrinth { public int Size { get; set; } }
```

- **Interfaces definiëren operaties**
  Lege methods en properties, die ingevuld moeten worden door class die de interface implementeert

```
public interface IFigure { void Draw(); }
```

Hierarchieën van Classes

- **Inheritance geeft hierarchieën van classes weer en/of interfaces in een applicatie:**

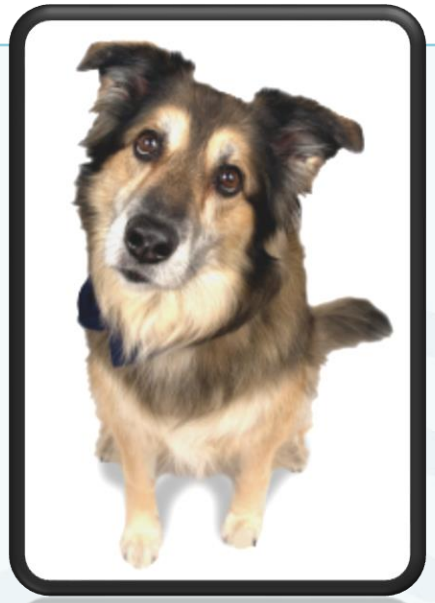


Voorbeeld Class Inheritance

```
public class Mammal
{
    public int Age { get; set; }

    public Mammal(int age)
    {
        this.Age = age;
    }

    public void Sleep()
    {
        Console.WriteLine("Shhh! I'm sleeping!");
    }
}
```

Voorbeeld Class Inheritance (2)

```
public class Dog : Mammal
{
    public string Breed { get; set; }

    public Dog(int age, string breed)
        : base(age)
    {
        this.Breed = breed;
    }

    public void WagTail()
    {
        Console.WriteLine("Tail wagging...");
    }
}
```



Inheritance

Demo

Access Levels

Access modifiers in C#

- `public` – toegankelijk vanuit alle classes
- `private` – enkel toegankelijk door class zelf
- `protected` – toegankelijk door class zelf en zijn afgeleide classes
- `internal` – (default) – toegankelijk door de eigen assembly, d.w.z. het huidig VS project
- `protected internal` – access is beperkt tot huidige assembly of afgeleide classes van de huidige class

Inheritance en Toegankelijkheid (accessibility)

```
class Creature
{
    protected string Name { get; private set; }
    private void Talk()
    {
        Console.WriteLine("I am creature ...");
    }
    protected void Walk()
    {
        Console.WriteLine("Walking ...");
    }
}
class Mammal : Creature
{
    // base.Talk() can be invoked here
    // this.Name can be read but cannot be modified here
}
```

Inheritance en Toegankelijkheid (accessibility) (2)

```
class Dog : Mammal
{
    public string Breed { get; private set; }
    // base.Talk() cannot be invoked here (it is private)
}
class Program
{
    static void Main()
    {
        Dog joe = new Dog(6, "Labrador");
        Console.WriteLine(joe.Breed);
        // joe.Walk() is protected and can not be invoked
        // joe.Talk() is private and can not be invoked
        // joe.Name = "Rex"; // Name cannot be accessed here
        // joe.Breed = "Shih Tzu"; // Can't modify Breed
    }
}
```

Inheritance en Accessibility

Demo

Inheritance: Belangrijke Aspecten

- **Structures kunnen niet worden overgeërfd**
- **In C# is er geen echte multiple inheritance**

  Enkel meerdere interfaces kunnen worden geïmplementeerd
- **Static members worden ook overgeërfd**
- **Constructors worden niet overgeërfd**
- **Inheritance is een transitieve relatie**

  Als C is afgeleid is van B en B is afgeleid van A, dan erft C ook over van A

Inheritance: Belangrijke functionaliteit

- **Wanneer een afgeleide class B zijn base (parent) class A uitbreidt:**
  B kan nieuwe members toevoegen
  B kan niet de overgeërfde members verwijderen
- **Nieuwe members in B met dezelfde naam of signatuur als A verbergt de members van B**
- **Een class kan virtual methods en properties declareren:**
  Afgeleide classes kunnen een override van de implementatie van de members voorzien
  Bv. `Object.ToString()` is een virtual method

Samenvatting

Inheritance (overerving)

Wat is Overerving (Inheritance)
Inheritance – Voordelen
Overervingstypes
Classes versus Interfaces
Hierarchieën van Classes

Access Levels
- Inheritance en Toegankelijkheid
- Inheritance: Belangrijke Aspecten
- Inheritance: Belangrijke Functionaliteit

leren. durven. doen.

SYNTRA WEST

REFERENTIES

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

HTTPS://WWW.LEARNCS.ORG/

FUNDAMENTALS OF COMPUTER PROGRAMMING WITH C#

© SVETLIN NAKOV & CO., 2013

WWW.INTROPROGRAMMING.INFO

Referenties

- Pro C# 7 with .NET and .NET CORE – Andrew TROELSEN – PHILIP JAPIKSE
- https://www.learncs.org/
- FUNDAMENTALS OF COMPUTER PROGRAMMING WITH C#
- © Svetlin Nakov & Co., 2013 www.introprogramming.info