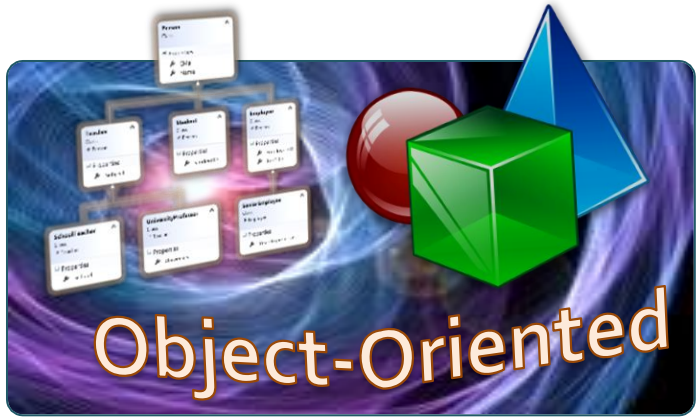
leren. durven. doen.

SYNTRA WEST

C# FUNDAMENTALS

C# PROGRAMMEREN 1

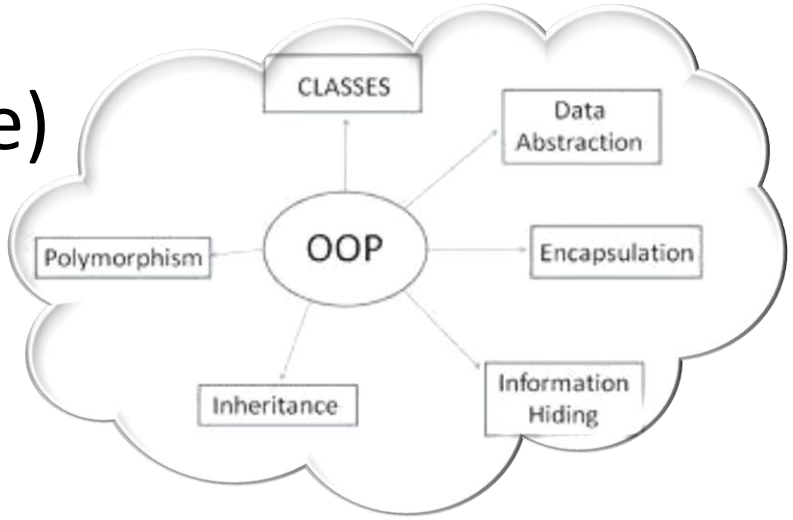
Object Georiënteerd programmeren (OOP)
Inleiding

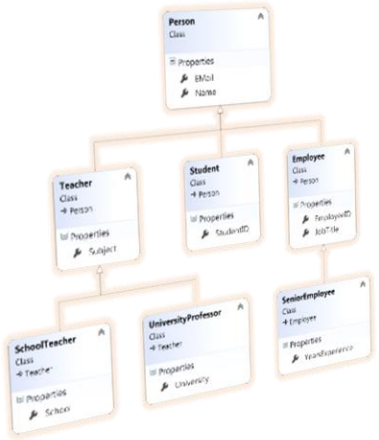


Inhoud

Basispricipes van OOP:

- Abstraction (abstractie)
- Encapsulation (inkapseling)
- Inheritance (overerving)
- Polymorphism(polymorfisme)

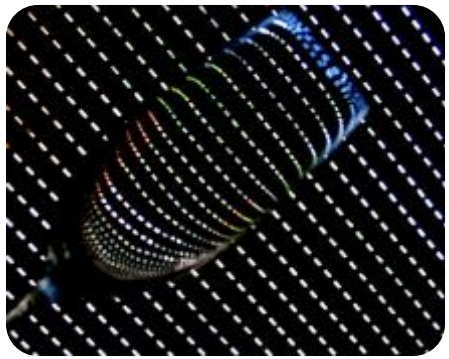




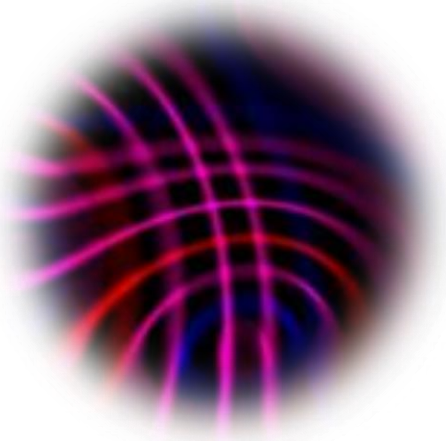
Basisprincipes van OOP

Basisprincipes van OOP

- Abstraction (abstractie)
  - Definiëren en gebruiken van abstracte concepten
- Encapsulation (inkapseling)
  - Verbergen van het inwendige, de implementatie
- Inheritance (overerving)
  - Overerven van ouder (parent) class
- Polymorphism (polymorfisme)
  - Het kunnen aannemen van verschillende vormen



Abstractie



Abstractie

- Abstractie is iets dat we elke dag meemaken

  - Wanneer we naar een object bekijken, kijken we naar zijn kenmerken die voor ons van belang zijn

  - We nemen de eigenschappen van het object en we behouden enkel wat we nodig hebben

  - Bv voor studenten vermelden we de "naam" maar niet “kleur ogen"

- Abstractie laat toe om een complexe realiteit voor te stellen met een vereenvoudigd model
- Abstractie benadrukt de eigenschappen dat we nodig hebben en verbergt de andere

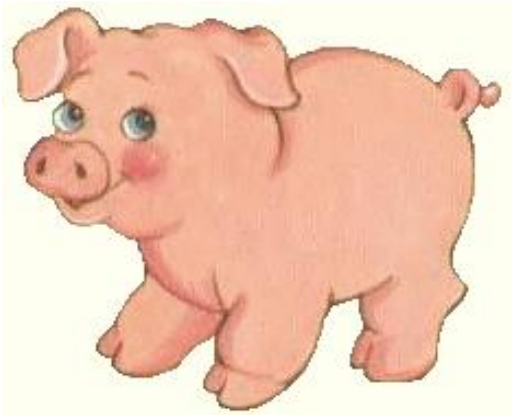
Abstractie

- Abstractie betekent dat enkel relevante zaken in acht worden genomen. Niet relevante eigenschappen , functies en kenmerken worden buiten beschouwing gelaten

- ... relevant voor het project (met het oog op toekomstig hergebruik voor gelijkaardige projecten)
- Abstractie = complexiteit beheren

Abstractie - Voorbeeld

- Project = kinderboerderij houden
- Relevante zaken: Dieren, zoals kippen, varkens, geiten honden en katten, bezoekers, verzorgers,...



Voorbeeld: Abstractie Hond

- Real-world objecten = honden op de kinderboerderij: Lassie, Bobby en Kirby
- Abstract bekijken: kenmerken en gedrag die van belang zijn:
- Hond:
  - Eigenschappen: ras, kleur vacht, leeftijd
  - Gedrag : blaffen, eten, zitten

- Abstracte definitie van hond -> Class Hond
- Objecten van Class Hond: Lassie, Bobby en Kirby

Verschil tussen Object en Klasse

- Software objecten modelleren real-world objecten of abstracte concepten
- Classes definiëren de structuur van objecten
  - soort prototype, template voor objecten

Class *Hond*

Eigenschappen:

- Ras
- Kleur
- Leeftijd

Gedrag:

- Spreek()
- Zit()
- Eet()

Object *Lassie*

Eigenschappen:

- Ras: “Collie”
- Kleur: “Bruin”
- Leeftijd: 2 jaar

Gedrag:

- Spreek()
- Zit()
- Eet()

Instanties van class Hond (Objecten) Voorbeelden

Klasse

Dog

```
ras: string
kleur: string
leeftijd: int
```

```
Spreek()
Zit()
Eet()
```

Object

Lassie

```
ras = "Collie"
kleur = "Bruin"
leeftijd = 2
```

Object

Bobby

```
ras ="WhiteTerrier"
kleur = “Wit”
leeftijd = 3
```

Object

Kirby

```
ras = "Pitbull"
kleur = “Beige”
leeftijd = 8
```

Instanties van class Kip (Objects) Voorbeelden

Klasse

Kip

kleur: string
EierenPerDag: short

Spreek()
LegEi()
Eet()

Object

Louise

kleur = “Bruin”
EierenPerDag = 2

Object

Sofie

kleur = “Bruin”
EierenPerDag = 1

Object

Claire

kleur = “Bruin”
EierenPerDag = 0

Encapsulatie

Microsoft .net

Encapsulatie (inkapseling)

- Encapsulation = verbergen van de details van de implementatie
- Class definieert
  - Toestand (gegevens): Velden en Eigenschappen via Fields en Properties
  - gedrag (operaties) via methoden
- Alle data members (fields) van een class zouden verborgen moeten blijven
  - Toegang regelen via properties (read-only en read-write)

Encapsulation – Voorbeeld

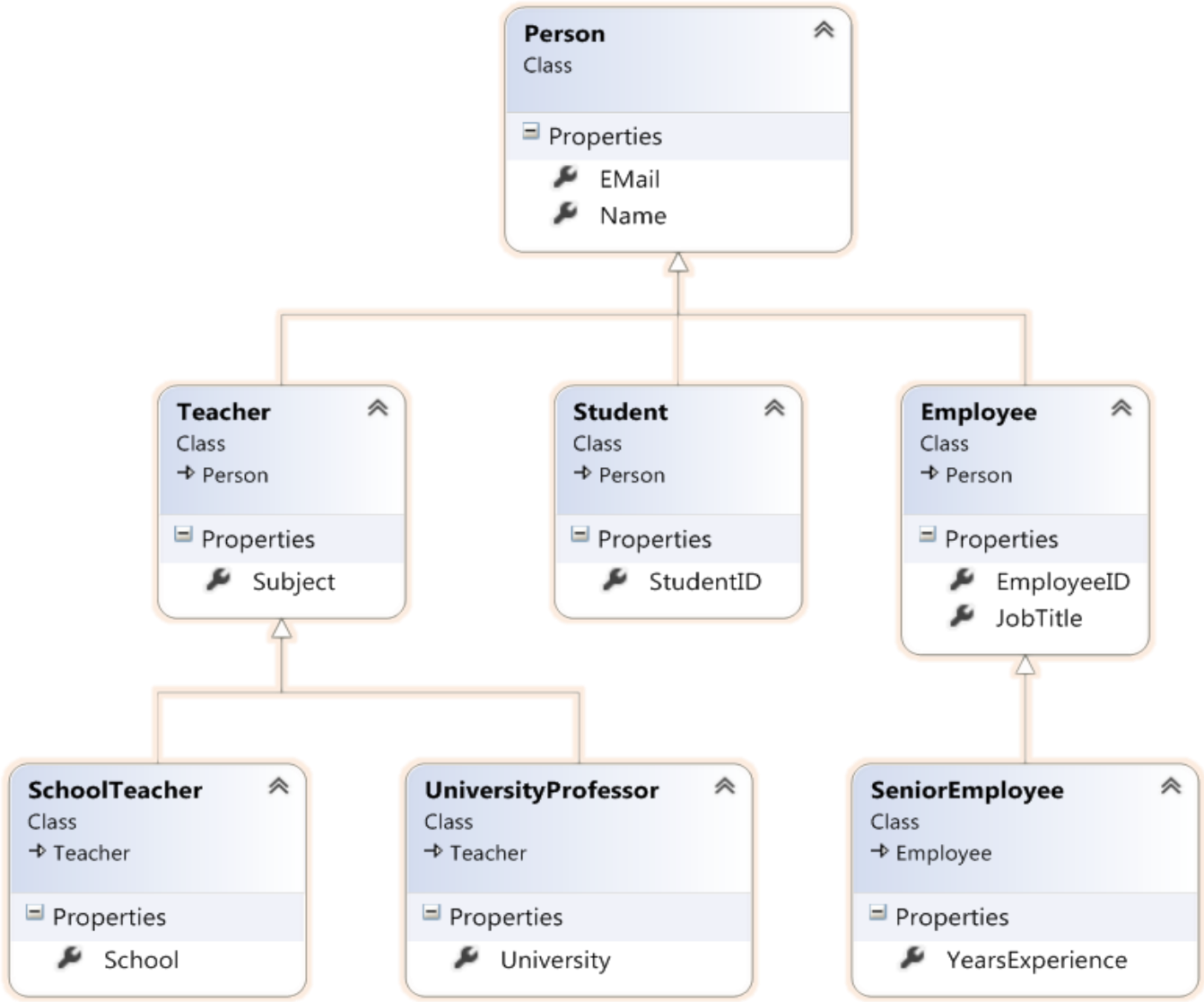
- Data fields ‘name’ en ‘age’ zijn verborgen
- Constructor “Person() ”en accessors Name en Age zijn gedefinieerd (getters and setters)

Person
-name : string -age : TimeSpan
+Person(string name, int age) +Name : string { get; set; } +Age : TimeSpan { get; set; }

Encapsulation in .NET

- Fields worden steeds `private` gedeclareerd
  - Worden aangesproken d.m.v properties in read-only of read-write mode
- Constructors worden meestal `public` gedeclareerd
- Interface Methods zijn steeds `public`
- Non-interface methods worden `private` / `protected` gedeclareerd

Inheritance

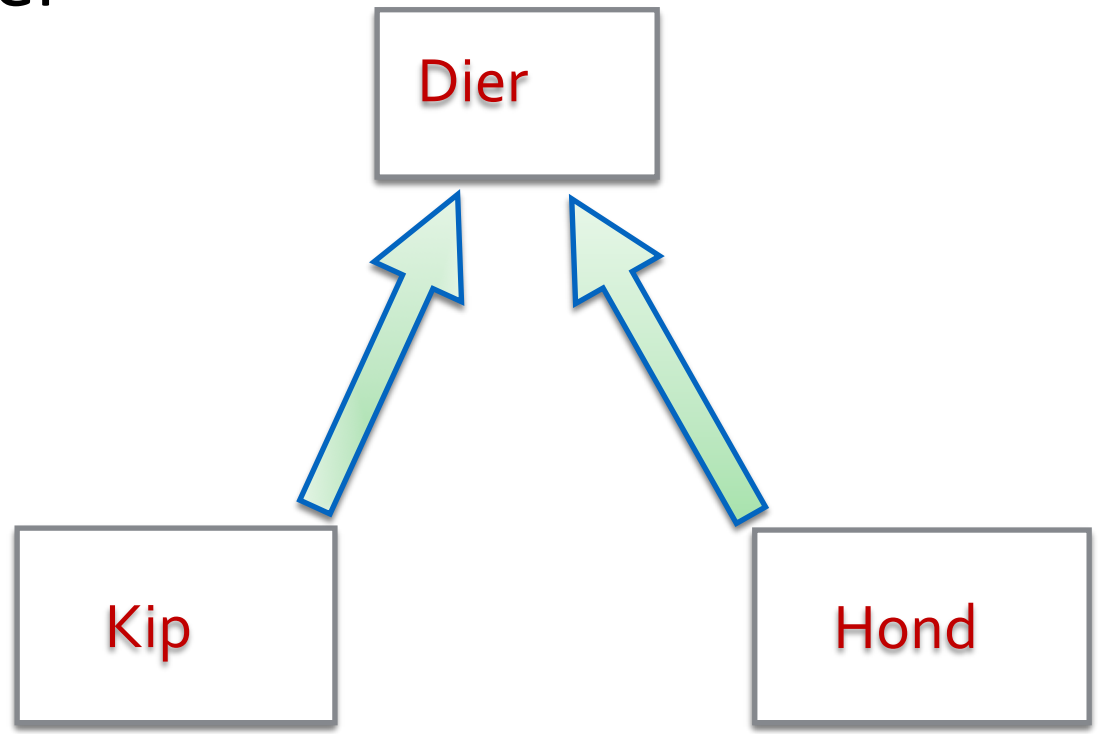


Inheritance (Overerving)

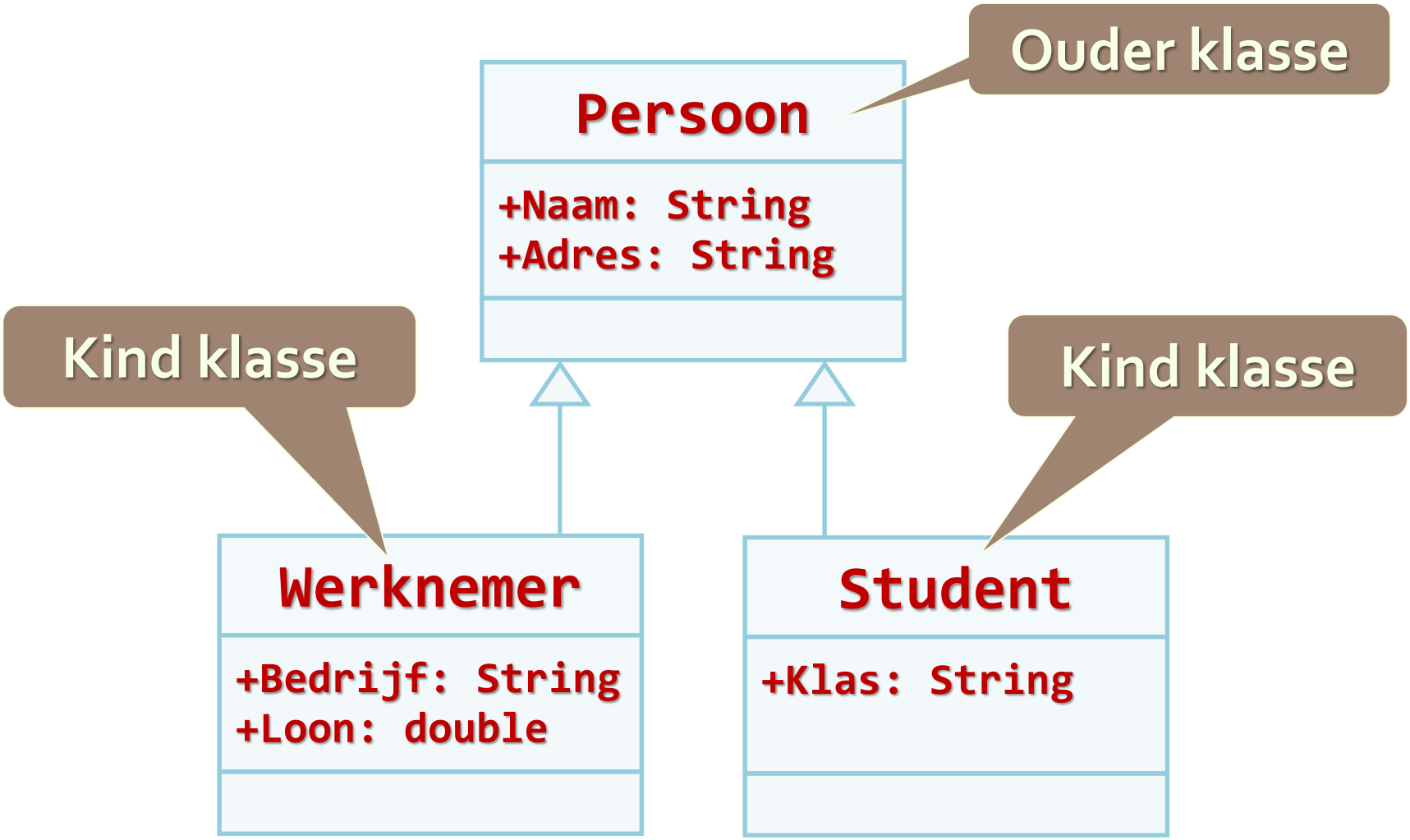
- Inheritance Laat toe dat kind- klassen de kenmerken van de ouder (base) class kunnen overerven:
  - Attributen (fields en properties)
  - Operaties (methoden)
- Kind (Child) klassen kunnen een extensie geven aan de Ouder (parent) class
  - extra fields, properties en methoden kunnen worden toegevoegd
  - Herdefiniëren van bestaande methoden (veranderen van bestaand gedrag)
- Een class kan een interface implementeren door een implementatie voor alle methods te voorzien

Inheritance – Voorbeeld Dieren

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



Inheritance – Voorbeeld Personen



Polymorfisme

Polymorphism (polymorfisme)

- Polymorphism = de mogelijkheid om meer dan 1 vorm aan te nemen
  (objecten hebben meer dan 1 type)

  Voorbeeld:

  – Kirby is zowel een hond als een dier

- Polymorfisme laat toe om gedrag te herdefiniëren
  – Voorbeeld:
  – `Een dier kan spreken, maar doen dit op zijn specifieke manier:`
    - `Hond Kirby -> Kirby.Speak() -> “Woef!”`
    - `Kip Louise -> Louise.Speak() -> “Pok poook!”`

Polymorfisme - Voorbeeld

Dier
kleur: string
Spreek() Eet()

Hond
ras: string kleur: string leeftijd: int
Spreek() Zit() Eet()

Samenvatting basispricipes OOP

- Abstraction (abstractie)
  - Definiëren en gebruiken van abstracte concepten
- Encapsulation (inkapseling)
  - Verbergen van het inwendige, de implementatie
- Inheritance (overerving)
  - Overerven van ouder (parent) class
- Polymorphism (polymorfisme)
  - Het kunnen aannemen van verschillende vormen

leren. durven. doen.

SYNTRA WEST

Vragen?

west-vlaanderen POM

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