



# Web Adv

## PHP: Object oriented programming

### DE HOGESCHOOL MET HET NETWERK

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt  
[www.pxl.be](http://www.pxl.be) - [www.pxl.be/facebook](http://www.pxl.be/facebook)



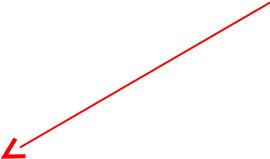
# PHP: OOP

Vergelijkbaar met OOP in Java

**Voorbeeld 1:**  
**klasse Auto**

## Klasse Auto, deel 1

Auto heeft niet-static eigenschappen  
\$nummerplaat en \$maxSnelheid



```
<?php
class Auto {
    // eigenschappen
    private $nummerplaat;
    private $maxSnelheid;
    private static $aantalAutos = 0;
    const landCode = 'B';

    //constructor
    function __construct ($nummerplaat, $maxSnelheid){
        $this -> nummerplaat = $nummerplaat;
        $this -> maxSnelheid = $maxSnelheid;
        self::$aantalAutos++;
        print("<br/>\n---- constructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
    //destructor
    function __destruct (){
        self::$aantalAutos--;
        print("<br/>\n---- destructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
}
```

## Klasse Auto, deel 1

Auto heeft static eigenschap  
\$aantalAutos

```
<?php
class Auto {
    // eigenschappen
    private $nummerplaat;
    private $maxSnelheid;
    private static $aantalAutos = 0;
    const landCode = 'B';

    //constructor
    function __construct ($nummerplaat, $maxSnelheid){
        $this->nummerplaat = $nummerplaat;
        $this->maxSnelheid = $maxSnelheid;
        self::$aantalAutos++;
        print("<br/>\n---- constructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
    //destructor
    function __destruct (){
        self::$aantalAutos--;
        print("<br/>\n---- destructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
}
```

## Klasse Auto, deel 1

Auto heeft een constante eigenschap  
landCode  
PHP: const ~ public static final  
(in java)

```
<?php
class Auto {
    // eigenschappen
    private $nummerplaat;
    private $maxSnelheid;
    private static $aantalAutos = 0;
    const landCode = 'B';

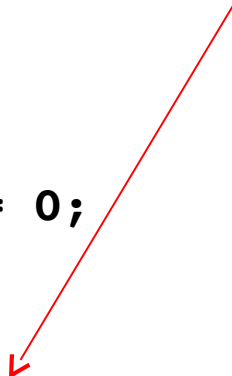
    //constructor
    function __construct ($nummerplaat, $maxSnelheid){
        $this->nummerplaat = $nummerplaat;
        $this->maxSnelheid = $maxSnelheid;
        self::$aantalAutos++;
        print("<br/>\n---- constructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
    //destructor
    function __destruct (){
        self::$aantalAutos--;
        print("<br/>\n---- destructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
}
```

# Klasse Auto, deel 1

Constructor:  
functie `__construct`

```
<?php
class Auto {
    // eigenschappen
    private $nummerplaat;
    private $maxSnelheid;
    private static $aantalAutos = 0;
    const landCode = 'B';

    //constructor
    function __construct ($nummerplaat, $maxSnelheid){
        $this -> nummerplaat = $nummerplaat;
        $this -> maxSnelheid = $maxSnelheid;
        self::$aantalAutos++;
        print("<br/>\n---- constructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
    //destructor
    function __destruct (){
        self::$aantalAutos--;
        print("<br/>\n---- destructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
}
```



# Klasse Auto, deel 1

Constructor:  
functie `__construct`

```
<?php
class Auto {
    // eigenschappen
    private $nummerplaat;
    private $maxSnelheid;
    private static $aantalAutos = 0;
    const landCode = 'B';

    //constructor
    function __construct ($nummerplaat, $maxSnelheid){
        $this -> nummerplaat = $nummerplaat;
        $this -> maxSnelheid = $maxSnelheid;
        self::$aantalAutos++;
        print("<br/>\n---- constructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }

    //destructor
    function __destruct (){
        self::$aantalAutos--;
        print("<br/>\n---- destructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
}
```

niet static eigenschappen krijgen een  
waarde via  
`$this -> variabelenaam`

# Klasse Auto, deel 1

```
<?php
class Auto {
    // eigenschappen
    private $nummerplaat;
    private $maxSnelheid;
    private static $aantalAutos = 0;
    const landCode = 'B';

    //constructor
    function __construct ($nummerplaat, $maxSnelheid){
        $this -> nummerplaat = $nummerplaat;
        $this -> maxSnelheid = $maxSnelheid;
        self::$aantalAutos++;
        print("<br/>\n---- constructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }

    //destructor
    function __destruct (){
        self::$aantalAutos--;
        print("<br/>\n---- destructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
}
```

Constructor:

functie `__construct`

static eigenschap wordt gewijzigd via

`self::$variabelenaam`



## Klasse Auto, deel 1

```
<?php
class Auto {
    // eigenschappen
    private $nummerplaat;
    private $maxSnelheid;
    private static $aantalAutos = 0;
    const landCode = 'B';

    //constructor
    function __construct ($nummerplaat, $maxSnelheid){
        $this->nummerplaat = $nummerplaat;
        $this->maxSnelheid = $maxSnelheid;
        self::$aantalAutos++;
        print("<br/>\n---- constructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
    //destructor
    function __destruct (){
        self::$aantalAutos--;
        print("<br/>\n---- destructor, aantal = " .
            self::$aantalAutos."<br/>\n");
    }
}
```

destructor (wordt aangeroepen als  
een object vernietigd wordt)  
functie \_\_destruct

## Klasse Auto, deel 2

```
// toString
function __toString(){
    return "Auto, code = ". self::landCode. ",
           nummerplaat = " . $this->nummerplaat .
           ", maxSnelheid = " . $this->maxSnelheid .
           ", aantalAutos = " . self::$aantalAutos;
}
```

functie \_\_toString  
wordt (automatisch)aangeropen  
telkens een stringrepresentatie  
nodig is  
e.g. print(\$auto1);

```
//setters en getters
public function setNummerPlaat($nummerplaat){
    $this -> nummerplaat = $nummerplaat;
}

public function setMaxSnelheid($maxSnelheid){
    $this -> maxSnelheid = $maxSnelheid;
}
```

## Klasse Auto, deel 2

setters

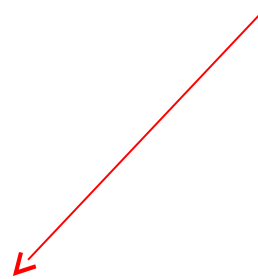
```
// toString
function __toString(){
    return "Auto, code = " . self::landCode. ",
           nummerplaat = " . $this->nummerplaat .
           ", maxSnelheid = " . $this->maxSnelheid .
           ", aantalAutos = " . self::$aantalAutos;
}

//setters en getters
public function setNummerPlaat($nummerplaat){
    $this -> nummerplaat = $nummerplaat;
}

public function setMaxSnelheid($maxSnelheid){
    $this -> maxSnelheid = $maxSnelheid;
}
```

## Klasse Auto, deel 3

getters voor eigenschappen  
\$nummerplaat en \$maxSnelheid  
niet-static functies



```
public function getNummerPlaat(){  
    return $this -> nummerplaat;  
}
```

```
public function getMaxSnelheid(){  
    return $this -> maxSnelheid;  
}
```

```
public static function getAantalAutos(){  
    return self::$aantalAutos;  
}  
}
```

## Klasse Auto, deel 3

getters voor static eigenschap  
\$aantalAutos  
static functie

```
public function getNummerPlaat(){  
    return $this -> nummerplaat;  
}
```

```
public function getMaxSnelheid(){  
    return $this -> maxSnelheid;  
}
```

```
public static function getAantalAutos(){  
    return self::$aantalAutos;  
}
```

```
}
```

# TestAuto.php

Bestand Auto.php wordt  
geladen



```
<?php
include (__DIR__ . ' /Auto.php');

$auto1 = new Auto('aqk-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid(). "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ');
```

# TestAuto.php

constructor wordt aangeroepen om een object van de klasse Auto te maken

object wordt in de variabele \$auto1 geplaatst

```
<?php
include (__DIR__ . ' /Auto.php' );

$auto1 = new Auto('aqk-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid(). "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ' );
```

## TestAuto.php

functie setMaxSnelheid wordt  
aangeropen

```
<?php
include (__DIR__ . ' /Auto.php');

$auto1 = new Auto('aqk-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid(). "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ');
```



## TestAuto.php

var\_dump van \$auto1



```
<?php
include (__DIR__ . ' /Auto.php');

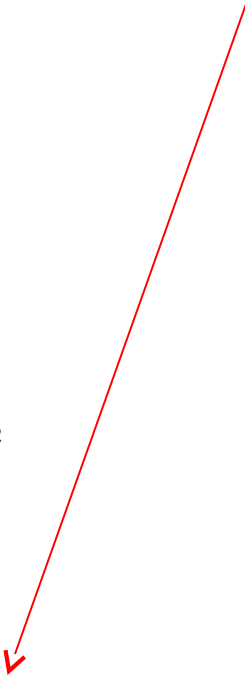
$auto1 = new Auto('aqk-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid(). "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ');
```

## TestAuto.php

getMaxSnelheid wordt  
aangeropen  
op \$auto1



```
<?php
include (__DIR__ . ' /Auto.php' );

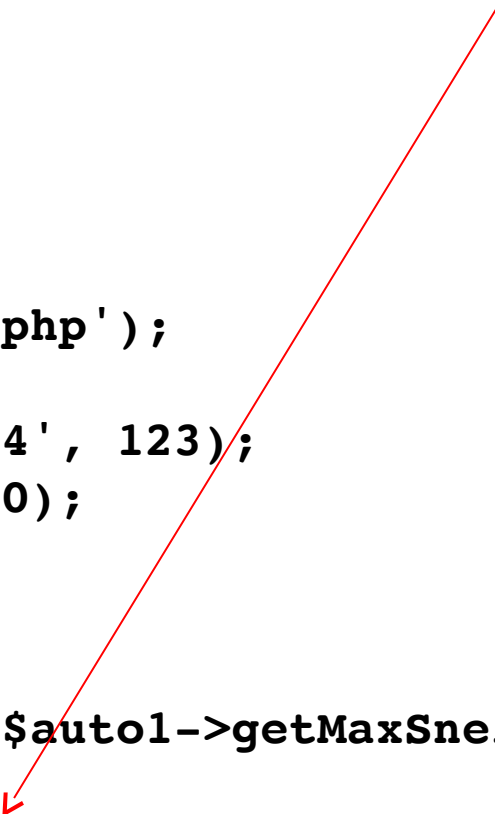
$auto1 = new Auto('aqr-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid() . "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ');
```

## TestAuto.php

nieuw object van de klasse Auto  
in \$auto2



```
<?php
include (__DIR__ . ' /Auto.php');

$auto1 = new Auto('aqk-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid(). "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ');
```

## TestAuto.php

static functie getAantalAutos wordt aangeroepen



```
<?php
include (__DIR__ . ' /Auto.php' );

$auto1 = new Auto('aqk-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid(). "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ' );
```

# TestAuto.php

vernietig \$auto2

```
<?php
include (__DIR__ . ' /Auto.php');

$auto1 = new Auto('aqk-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid(). "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ');
```

## TestAuto.php

static functie getAantalAutos  
wordt aangeroepen

```
<?php
include (__DIR__ . ' /Auto.php');

$auto1 = new Auto('aqr-834', 123);
$auto1->setMaxSnelheid(150);
print ('<pre>');
var_dump($auto1);
print ('</pre>');
print("maxSnelheid = " . $auto1->getMaxSnelheid(). "<br/>\n");

$auto2 = new Auto('lsf-132', 129);
print( ' <br/>aantalAutos = ' . Auto::getAantalAutos());
unset ($auto2);

print( ' <br/>aantalAutos = ');
print( Auto::getAantalAutos());
print( ' <br/> ');

print($auto1);
print ("stop<br/>\n");
```

Variabelen kunnen naar een object van een klasse refereren.

## TestAuto2.php

```
<?php
include ( __DIR__ . ' /Auto.php' );

$auto1 = new Auto( 'aqk-834' , 150 );           //stap 1
$auto2  = $auto1;                               //stap 2
$auto1->setMaxSnelheid(123);                     //stap 3
print ( '<pre>' );
var_dump($auto1);
var_dump($auto2);
print ( "</pre>\n" );
?>
```

Variabelen kunnen naar een object van een klasse refereren.

## TestAuto2.php

```
$auto1 = new Auto('aqk-834', 150);
```

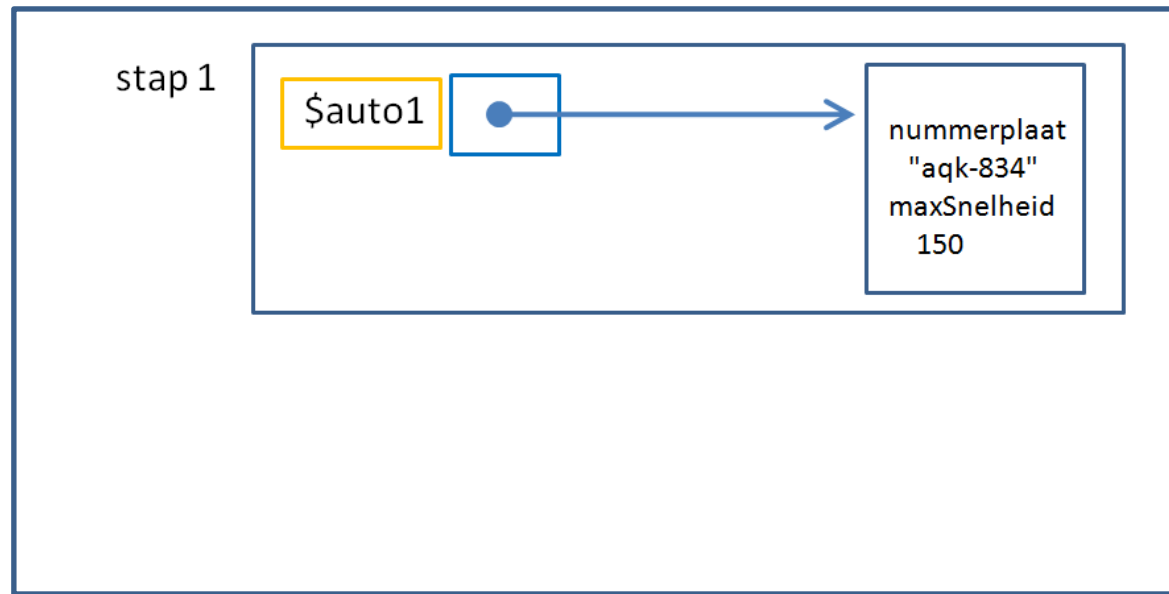
**//stap 1**

```
$auto2 = $auto1;
```

**//stap 2**

```
$auto1->setMaxSnelheid(123);
```

**//stap 3**





Variabelen kunnen naar een object van een klasse refereren.

## TestAuto2.php

```
$auto1 = new Auto('aqk-834', 150);
```

//stap 1

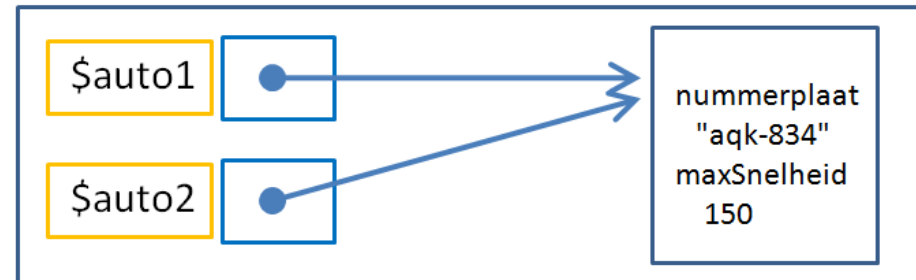
```
$auto2 = $auto1;
```

**//stap 2**

```
$auto1->setMaxSnelheid(123);
```

//stap 3

stap 2



Variabelen kunnen naar een object van een klasse refereren.

## TestAuto2.php

```
$auto1 = new Auto('aak-834', 150);
```

//stap 1

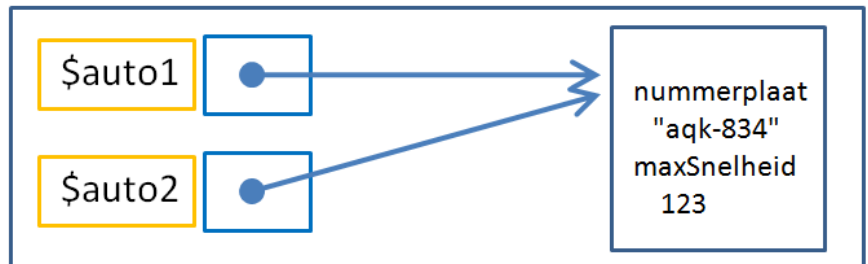
```
$auto2 = $auto1;
```

//stap 2

```
$auto1->setMaxSnelheid(123);
```

//stap 3

stap 3



wijziging in `$auto1` wordt gemerkt in `$auto2`

# 2 Inheritance

- keyword extends

```
class A extends B{  
  
}
```

- Geen multiple inheritance:

```
class A extends B, C{  
  
}
```

Voorbeeld: Vrachtwagen wordt afgeleid van Auto

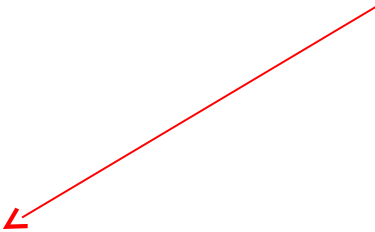
# Vrachtwagen.php, deel 1

Vrachtwagen wordt afgeleid  
van Auto

≈ Vrachtwagen is een Auto

```
<?php
```

```
class Vrachtwagen extends Auto {  
    // eigenschappen  
    private $laadvermogen;  
  
    //constructor  
    function __construct ($nummerplaat,$maxSnelheid,$lvm){  
        parent::__construct ($nummerplaat, $maxSnelheid);  
        $this->laadvermogen = $lvm;  
    }  
  
    //destructor  
    function __destruct () {  
        parent::__destruct ();  
    }  
}
```

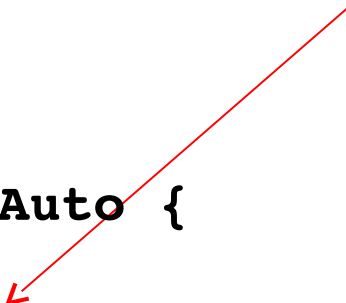


# Vrachtwagen.php, deel 1

Vrachtwagen heeft  
extra eigenschap:  
\$laadvermogen

```
<?php
```

```
class Vrachtwagen extends Auto {  
    // eigenschappen  
    private $laadvermogen;  
  
    //constructor  
    function __construct ($nummerplaat,$maxSnelheid,$lvm){  
        parent::__construct ($nummerplaat, $maxSnelheid);  
        $this->laadvermogen = $lvm;  
    }  
  
    //destructor  
    function __destruct () {  
        parent::__destruct ();  
    }  
}
```



# Vrachtwagen.php, deel 1

constructor: `__construct`



```
<?php
```

```
class Vrachtwagen extends Auto {  
    // eigenschappen  
    private $laadvermogen;  
  
    //constructor  
    function __construct ($nummerplaat,$maxSnelheid,$lvm){  
        parent::__construct ($nummerplaat, $maxSnelheid);  
        $this->laadvermogen = $lvm;  
    }  
  
    //destructor  
    function __destruct () {  
        parent::__destruct ();  
    }  
}
```

# Vrachtwagen.php, deel 1

```
<?php
```

```
class Vrachtwagen extends Auto {  
    // eigenschappen  
    private $laadvermogen;
```

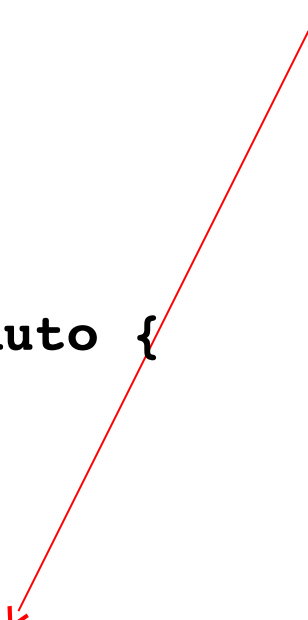
```
    //constructor
```

```
    function __construct ($nummerplaat, $maxSnelheid, $lvm) {  
        parent::__construct ($nummerplaat, $maxSnelheid);  
        $this->laadvermogen = $lvm;  
    }
```

```
    //destructor
```

```
    function __destruct () {  
        parent::__destruct ();  
    }
```

constructor: \_\_construct  
constructor van Auto wordt  
aangeroepen via  
parent::\_\_construct

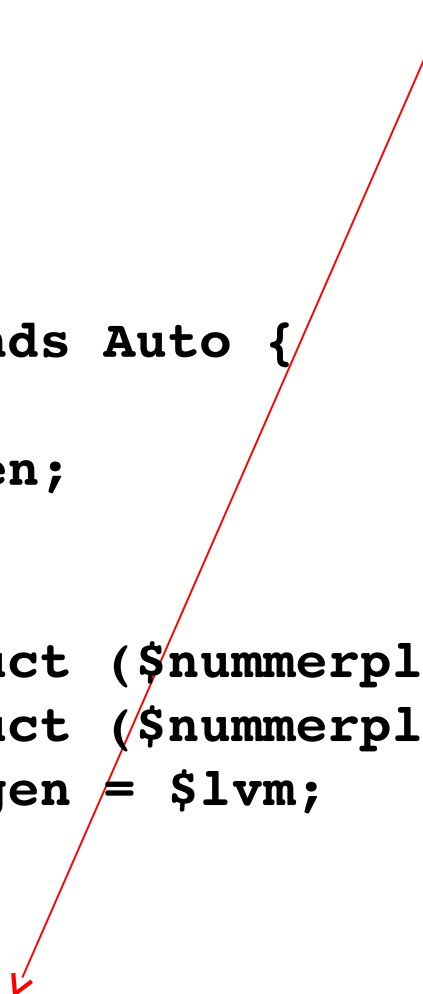


# Vrachtwagen.php, deel 1

```
<?php
```

```
class Vrachtwagen extends Auto {  
    // eigenschappen  
    private $laadvermogen;  
  
    //constructor  
    function __construct ($nummerplaat,$maxSnelheid,$lvm){  
        parent::__construct ($nummerplaat, $maxSnelheid);  
        $this->laadvermogen = $lvm;  
    }  
  
    //destructor  
    function __destruct (){  
        parent::__destruct ();  
    }  
}
```

destructor: \_\_destruct  
destructor van Auto wordt  
aangeroepen via  
parent::\_\_destruct






## Vrachtwagen.php, deel 2

functie `__toString`  
(method overriding)

`__toString` van Auto wordt  
aangeropen via  
`parent::__toString()`

```
// toString
function __toString(){
    return "Vrachtwagen is een ". parent::__toString()
        . " laadvermogen = ".$this->laadvermogen;
}
```

A red arrow points from the `parent::__toString()` call in the code to the text "functie \_\_toString (method overriding)". A blue arrow points from the same call to the text "\_\_toString van Auto wordt aangeropen via parent::\_\_toString()".

```
//setters en getters
public function setLaadvermogen($laadvermogen){
    $this -> laadvermogen = $laadvermogen;
}
```

```
public function getLaadvermogen(){
    return $this -> laadvermogen;
}
```

```
}
```

## Vrachtwagen.php, deel 2

setters en getters

```
// toString
function __toString(){
    return "Vrachtwagen is een ". parent::__toString()
        . " laadvermogen = ".$this->laadvermogen;
}
```

```
//setters en getters
public function setLaadvermogen($laadvermogen){
    $this -> laadvermogen = $laadvermogen;
}
```

```
public function getLaadvermogen(){
    return $this -> laadvermogen;
}
```

```
}
```

# 3 Access modifiers

Eigenschappen en functies in een klasse kunnen private, protected en public gedefinieerd worden.

In PHP geldt

private        =    de eigenschap of functie is enkel  
                      beschikbaar in de klasse zelf

protected      =    de eigenschap of functie is beschikbaar  
                      in de klasse en in elke klasse afgeleid van  
                      deze klasse

public           =    de eigenschap of functie is overal  
                      beschikbaar

# 4 Static & niet static

## Niet-static eigenschappen

- kunnen enkel gebruikt worden als er een object van deze klasse gemaakt is.
- elk object van de klasse heeft zijn eigen waarde voor deze eigenschappen

niet-static \$variabeleNaam uit de klasse A kan gebruikt worden als

**\$this** -> variabeleNaam                      in A zelf

**\$this** -> variabeleNaam                      in een klasse afgeleid van A

NB \$super bestaat niet (!!!!)

**\$obj**->variabeleNaam              erbuiten  
(\$obj bevat een object van klasse A)

## static eigenschap

- kan gebruikt worden zonder dat er een object van de klasse aangemaakt is.
- static variabelen worden bewaard op het niveau van de klasse. Elk object van deze klasse heeft dus toegang tot dezelfde waarde.

Static eigenschappen uit de klasse A worden gebruikt als

`self::$variabeleNaam`                      in A zelf  
`static::$variabeleNaam`

`parent::$variabeleNaam`              in een klasse afgeleid van A

`A::$variabeleNaam`      erbuiten

## niet-static functie

in de uitwerking :

- mogen zowel static als niet-static eigenschappen gebruikt worden
- mogen zowel static als niet-static functies aangeroepen worden
- \$this mag gebruikt worden

## static functie

in de uitwerking:

- mogen enkel static eigenschappen gebruikt worden
- mogen enkel static functies aangeroepen worden
- \$this mag niet gebruikt worden

De functie functie(...) uit de klasse A kan gebruikt worden als

self::functie(...)  
static::functie(...)  
\$this->functie(...)

in de klasse zelf

parent:: functie(...)  
\$this->functie(...)

in een klasse afgeleid van deze klasse

\$obj->functie(...)

erbuiten

*(\$obj bevat een object van A)*

A::functie(...)

erbuiten

# Opm. self vs static

self	~ huidige klasse
static	~ 'calling' class

Voorbeeld.

klasse A met methode getClassname, methode whoami

klasse B getClassname wordt overschreven

```
<?php
```

```
class A{
    public static function getClassname(){
        return "A: klasse= ".__CLASS__;
    }
    public static function whoami(){
        print("<self:> ". self::getClassname() .
            " <static:> ". static::getClassname(). "\n");
    }
}
```

```
class B extends A{
    public static function getClassname(){
        return "B: klasse= ".__CLASS__;
    }
}
```

```
A::whoami();
```

```
B::whoami();
```

```
$ php a.php
<self:> A: klasse= A <static:> A: klasse= A
<self:> A: klasse= A <static:> B: klasse= B
$
```



# 5 Type hinting

- type kan vermeld worden bij argument van een functie als type een klasse is

type hinting en  
default waarde

Bekijk vb3/Persoon.php

```
...  
public function __construct ($naam, Auto $a =NULL) {  
    $this -> naam = $naam;  
    $this -> auto = $a;  
}
```

```
...  
public function setAuto(Auto $a) {  
    $this -> auto = $a;  
}
```

type hinting

## Autoloading:

Wanneer een klasse gebruikt wordt, wordt de methode `__autoload` aangeropen.

In deze methode wordt het bestand via `include` toegevoegd.

vb3/TestPersoon.php

```
<?php  
function __autoload($class_name) {  
    include __DIR__ . '/' . $class_name . '.php';  
}  
  
$auto1 = new Auto('aqk-834', 123);
```

Type-hinting kan ook gebruikt worden voor arrays,

vb4/TestRij.php

```
<?php
function printRij (array $rij) {
    print_r($rij);
}
$a = array('1', 2, 'ff');
printRij($a);
$b = '12';
printRij($b); // leidt tot een exception !!
?>
```

# 6 Abstract class

- Geen object van abstracte klasse
- Wel klasse afgeleid van abstracte klasse

In abstracte klasse:

- aantal eigenschappen vastgelegd
- aantal methodes, constructor, destructor  
→ worden volledig uitgewerkt

maar ook:

- abstracte methodes: enkel definitie geen uitwerking

In afgeleide klasse moet abstracte methode concreet uitgewerkt worden.

2DObject is abstract

```
<?php
```

```
abstract class TweedimensionaalObject{  
    private $beschrijving;  
  
    public function __construct($beschrijving){  
        $this->beschrijving = $beschrijving;  
    }  
  
    public function __toString(){  
        return $this->getBeschrijving();  
    }  
  
    public function getBeschrijving(){  
        return $this->beschrijving;  
    }  
  
    public abstract function  
        berekenOppervlak();  
}
```

wel concreet:

- eig \$beschrijving

- constructor

- toString

- getBeschrijving

abstracte methode

berekenOpp

voor elk 2DObject opp berekene  
niet geweten hoe ???

```
<?php
class Cirkel extends TweedimensionaalObject{
    private $straal;

    public function __construct($straal){
        $beschrijving = "Cirkel r = $straal";
        $this->straal = $straal;
        parent::__construct ($beschrijving);
    }

    public function __toString(){
        return parent::__toString();
    }

    public function berekenOppervlak(){
        return M_PI * pow ($this->straal, 2);
    }
}
```

Cirkel is afgeleid van  
2DObject

extra eigenschap  
\$straal

constructor wordt  
overschreven

toString wordt  
overschreven

berekenOpp  
moet hier uitgewerkt  
worden

```
<?php
class Vierkant extends TweedimensionaalObject{
    private $zijde;

    public function __construct($zijde){
        $beschrijving = "Vierkant zijde = $zijde";
        $this->zijde = $zijde;
        parent::__construct ($beschrijving);
    }

    public function berekenOppervlak(){
        return pow ($this->zijde, 2);
    }
}
```

# 7 Interface

Interface ~ abstracte klasse

- met - enkel abstracte methodes
- (ook constantes)

Nut: 'contract'

functionaliteit wordt vastgelegd

klasse die interface implementeert  
moet abstracte methodes concreet uitwerken

in andere stukken code kan de klasse gebruikt  
worden (zonder de concrete uitwerking te moeten  
kennen)

in ander stuk code is 'interface' ('wat er mee gedaan  
kan worden') gekend



```
<?php
interface OppervlakteBerekenbaar{
    const pi = 3.1415927653;
    public function berekenOppervlak();
}
?>
```

abstracte methode



```
<?php
class Cirkel implements OppervlakteBerekenbaar{
    private $straal;

    public function __construct($straal){
        $this->straal = $straal;
    }

    public function __toString(){
        return "Een cirkel met straal " . $this->straal;
    }

    public function berekenOppervlak(){
        return self::pi * pow ($this->straal, 2);
    }
}
?>
```

## Iterator: voorgedefinieerde interface

vb7/TraversableString.php

```
class TraversableString implements Iterator {
```

TraversableString moet methodes next, current, key rewind uit interface Iterator aanbieden

Deze methodes worden gebruikt in de foreach structuur

Dus met een foreach kan door een TraversableString gegaan worden

```
<?php
class TraversableString implements Iterator {
    private $positie;
    private $inhoud;

    public function __construct() {
        $this->inhoud = "";
        $this->positie = 0;
    }

    public function setInhoud($inhoud){
        if(is_string($inhoud)){
            $this->inhoud=$inhoud;
        }
    }

    public function current() {
        return $this->inhoud[$this->positie];
    }
}
```

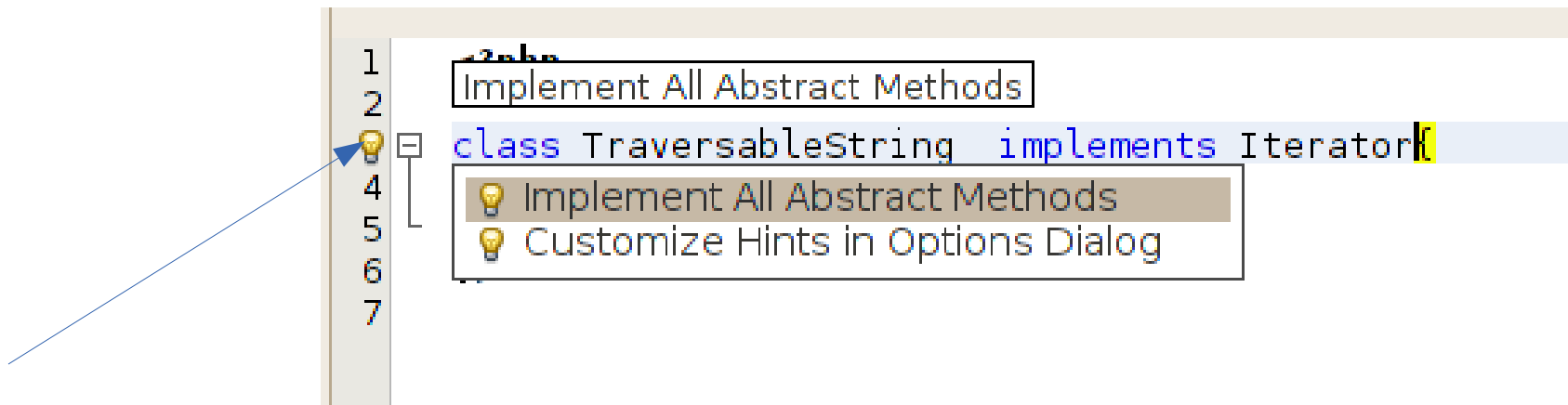
```
public function key() {  
    return $this->positie;  
}  
public function next() {  
    $this->positie++;  
}  
public function rewind() {  
    $this->positie=0;  
}  
public function valid() {  
    return strlen($this->inhoud) > $this->positie;  
}  
}
```

?>

Bekijk - vb7/TraversableString.php  
- vb7/Test.php

vb7/TraversableString.php

class TraversableString implements **Iterator** {



# 8 Final

Van een final klasse kan geen klasse afgeleid worden en een final functie kan niet overschreven worden in een afgeleide klasse.

<http://www.php.net/manual/en/language.oop5.final.php>

# Oefening 1

Creëer een klasse Datum met als kenmerken: dag, maand en jaartal

Voorzie een constructor om een datum op volgende manieren te maken:

zonder argumenten: datum = 1/1/2008

met enkel een argument voor dag en maand: datum = d/m/2008

met 3 argumenten

Maak gebruik van default argumenten of van func\_get\_args.

De datum moet in het formaat 1/1/2008 afgedrukt kunnen worden via de methode toon().

Voorzie een extra methode toonMaand() die de datum als volgt afdruckt: 1/januari/2008.

Maak hiervoor gebruik van een private static array \$maanden.

Voorzie setters en getters voor dag, maand en jaar.

Maak in invoer.html een formulier om dag, maand en jaar in te geven.

In verwerk.php wordt een Datum-object aangemaakt en via de methode toonMaand afgedrukt.



## Oefening 2

Maak een klasse Voertuig met als eigenschappen: merk, type, bouwjaar. Maak deze kenmerken protected.

Een auto is een speciaal type voertuig: een bijkomend kenmerk is het aantal deuren.

Een vrachtwagen is eveneens een speciaal type voertuig: een bijkomend kenmerk is het laadvermogen.

Voorzie een gepaste constructor, illustreer het gebruik van protected, voorzie een print-methode. Voeg nadien een static variabele aan de klasse Voertuig toe om het aantal gecreëerde voertuigen te kunnen bepalen.

# Bronnen



<http://php.net/manual/en/language.oop5.php>



<http://phpro.org/tutorials/Object-Oriented-Programming-with-PHP.html>