



# Web advanced

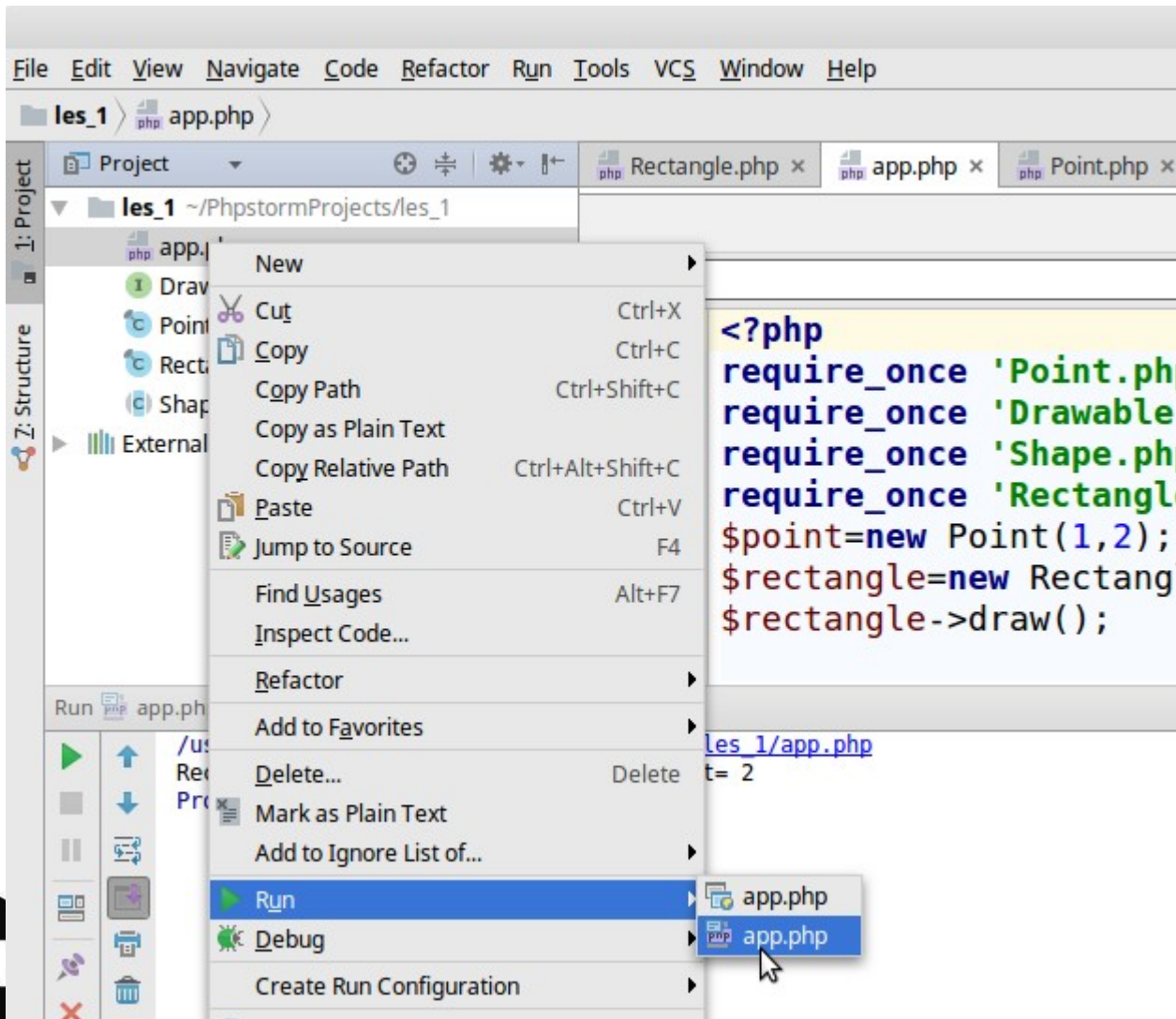
OOP

## 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)



# Software



# Voorbeeld1: instance

```
1 <?php
```

```
2  
3 final class Point
```

```
4 {
```

```
5     private $x;
```

```
6     private $y;
```

```
7  
8     public function __construct($x, $y)
```

```
9 {
```

```
10     $this->x = $x;
```

```
11     $this->y = $y;
```

```
12 }
```

```
13  
14     function __toString()
```

```
15 {
```

```
16     return "($this->x , $this->y)";
```

```
17 }
```

```
18  
19     public function calculateDistance(Point $point)
```

```
20 {
```

```
21     return sqrt( ($this->x-$point->x)*($this->x-$point->x)+  
22                 ($this->y-$point->y)*($this->y-$point->y) );
```

```
23 }
```

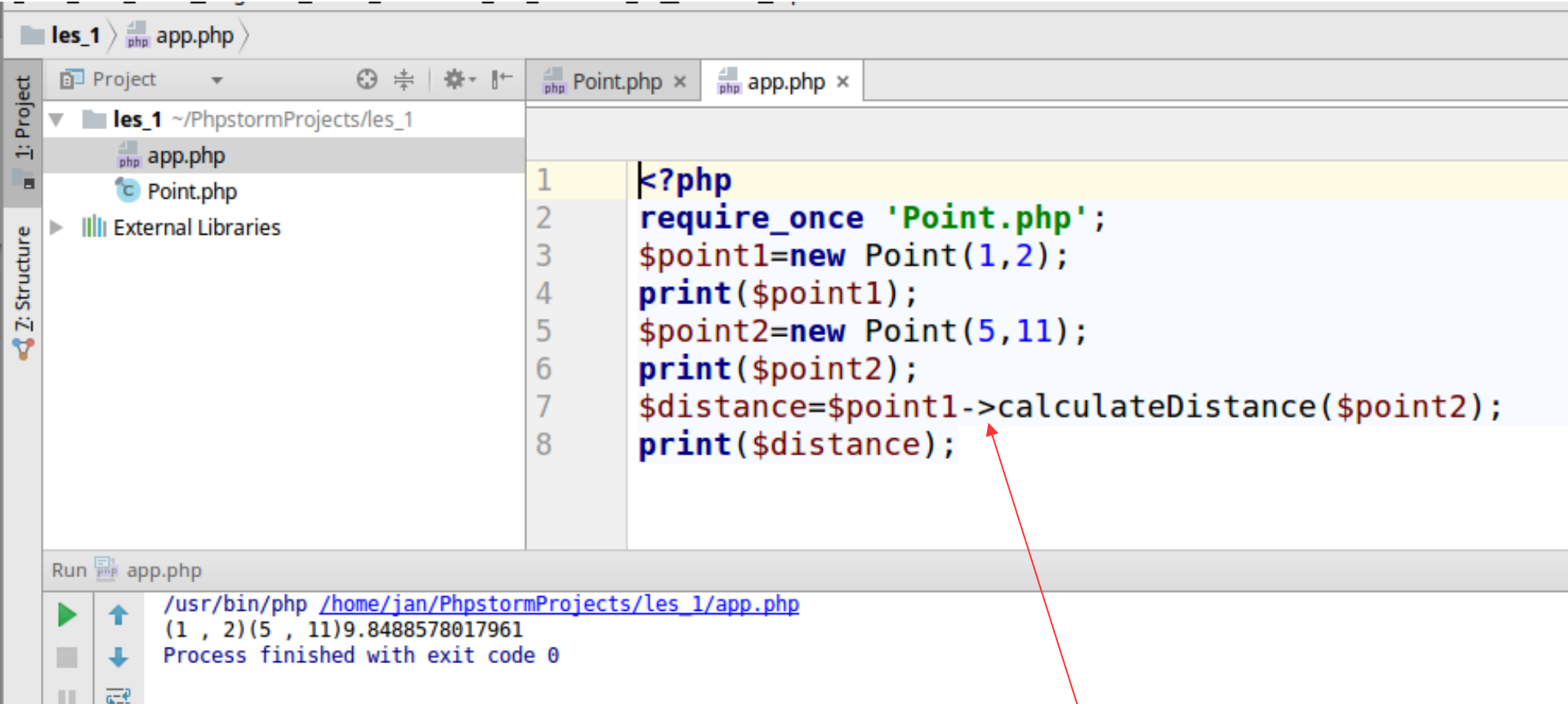
```
24  
25 }
```

2 underscores voor  
construct & toString  
(magic methods)

instance variable: \$this->

type-hinting

# Voorbeeld1: instance



The screenshot shows an IDE with a project named 'les\_1'. The file explorer on the left shows 'app.php' and 'Point.php'. The main editor displays the contents of 'app.php' with the following code:

```
1 <?php
2 require_once 'Point.php';
3 $point1=new Point(1,2);
4 print($point1);
5 $point2=new Point(5,11);
6 print($point2);
7 $distance=$point1->calculateDistance($point2);
8 print($distance);
```

The 'Run' button is highlighted, and the output console at the bottom shows the execution results:

```
Run app.php
/usr/bin/php /home/jan/PhpstormProjects/les_1/app.php
(1 , 2)(5 , 11)9.8488578017961
Process finished with exit code 0
```

A red arrow points from the text 'instance method: \$object->' to the '\$point1->' part of the code on line 7.

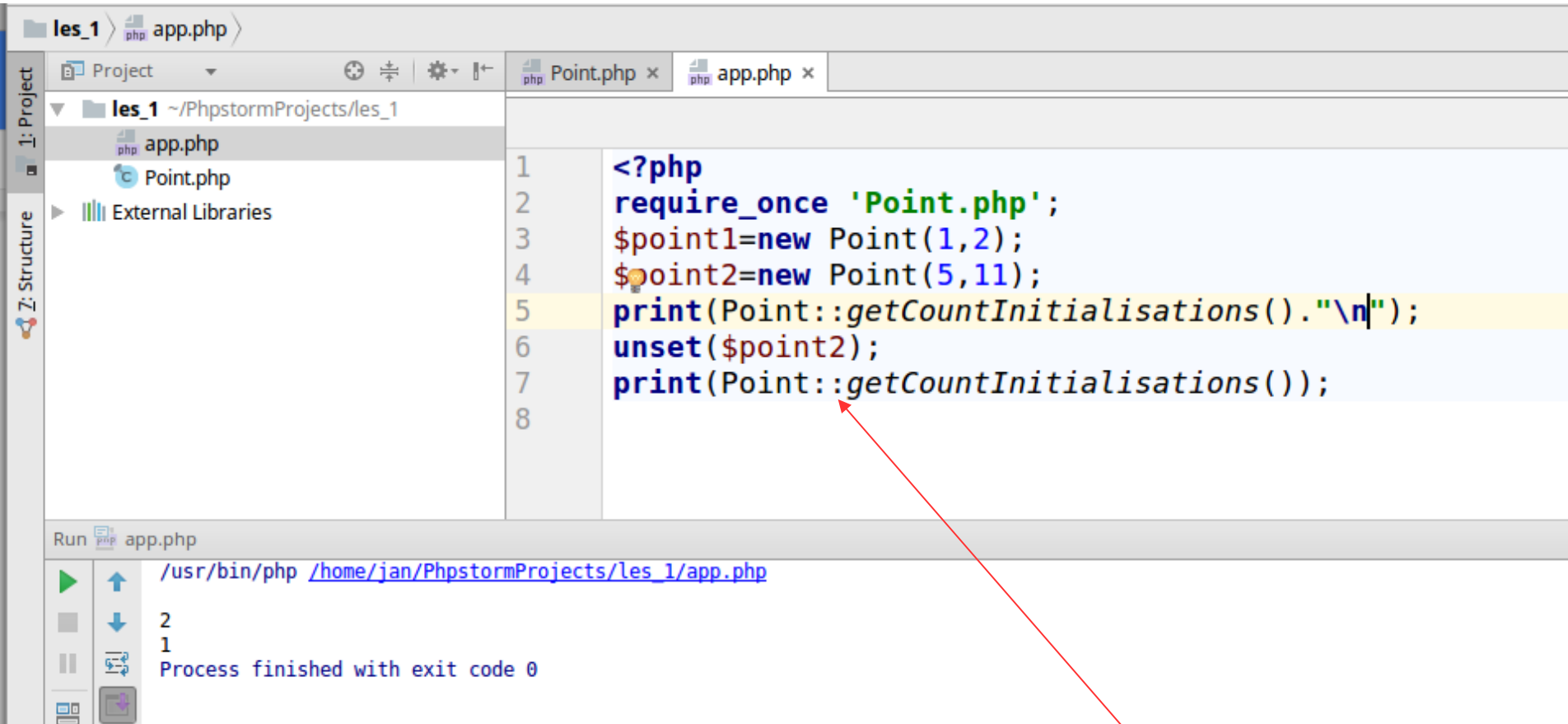
instance method:  
\$object->

# Voorbeeld2: static

```
1  <?php
2  final class Point
3  {
4      private $x,$y;
5      private static $countInitialisations=0;
6
7      public function __construct($x, $y)
8      {
9          $this->x = $x;
10         $this->y = $y;
11         self::$countInitialisations++;
12     }
13     public function __destruct()
14     {
15         self::$countInitialisations--;
16     }
17
18
19     public static function getCountInitialisations()
20     {
21         return self::$countInitialisations;
22     }
23 }
24
```

static variable: self::

# Voorbeeld2: static



The screenshot shows the PHPStorm IDE interface. On the left, the 'Project' and 'Structure' toolbars are visible. The main editor displays a file named 'app.php' with the following PHP code:

```
1 <?php
2 require_once 'Point.php';
3 $point1=new Point(1,2);
4 $point2=new Point(5,11);
5 print(Point::getCountInitialisations()."\n");
6 unset($point2);
7 print(Point::getCountInitialisations());
8
```

The code is executed, and the output is shown in the 'Run' window at the bottom. The output indicates that the process finished with exit code 0.

Run app.php  
/usr/bin/php /home/jan/PhpstormProjects/les\_1/app.php  
2  
1  
Process finished with exit code 0

static method:  
ClassName::

# Abstract class

```
1 <?php
2 abstract class Shape
3 {
4     private $point;
5
6     public function __construct(Point $point)
7     {
8         $this->point=$point;
9     }
10
11     function __toString()
12     {
13         return $this->point->__toString();
14     }
15
16     public abstract function calculatePerimeter();
17
18 }
```

volledig  
uitgewerkt

elke shape  
omtrek  
berekenen  
hoe?



# Abstract class

```
1 <?php
2
3 final class Rectangle extends Shape
4 {
5     private $width, $height;
6     public function __construct(Point $point, $width, $height)
7     {
8         parent::__construct($point);
9         $this->width=$width;
10        $this->height=$height;
11    }
12
13    public function calculatePerimeter()
14    {
15        return 2*$this->width+2*$this->height;
16    }
17
18    public function __toString()
19    {
20        return "Rectangle, Point= ". parent::__toString() ." width=
21            $this->width height= $this->height";
22    }
23 }
```

Rectangle niet  
abstract =>  
calculatePerimeter  
verplicht



# Interface

```
<?php
```

```
interface Drawable
```

```
{
```

```
    public function draw();
```

```
}
```

abstract  
methode  
draw

```
1 <?php
```

```
2 abstract class Shape implements Drawable
```

```
3 {
```

```
4     private $point;
```

```
5
```

```
6     public function __construct(Point $point)
```

```
7     {
```

```
8         $this->point=$point;
```

```
9     }
```

```
10
```

```
11     function __toString()
```

```
12     {
```

```
13         return $this->point->__toString();
```

```
14     }
```

```
15
```

```
16     public abstract function calculatePerimeter();
```

```
17
```

```
18 }
```

```
19
```

abstract  
methode

```
1 <?php
```

```
2  
3 final class Rectangle extends Shape
```

```
4 {
```

```
5     private $width, $height;
```

```
6     public function __construct(Point $point, $width, $height)
```

```
7 {
```

```
8         parent::__construct($point);
```

```
9         $this->width=$width;
```

```
10        $this->height=$height;
```

```
11    }
```

```
12  
13    public function calculatePerimeter()
```

```
14 {
```

```
15     return 2*$this->width+2*$this->height;
```

```
16 }
```

```
17  
18    public function __toString()
```

```
19 {
```

```
20     return "Rectangle, Point= ". parent::__toString() .
```

```
21     " width= $this->width height= $this->height";
```

```
22 }
```

```
23  
24    public function draw()
```

```
25 {
```

```
26     print($this->__toString());
```

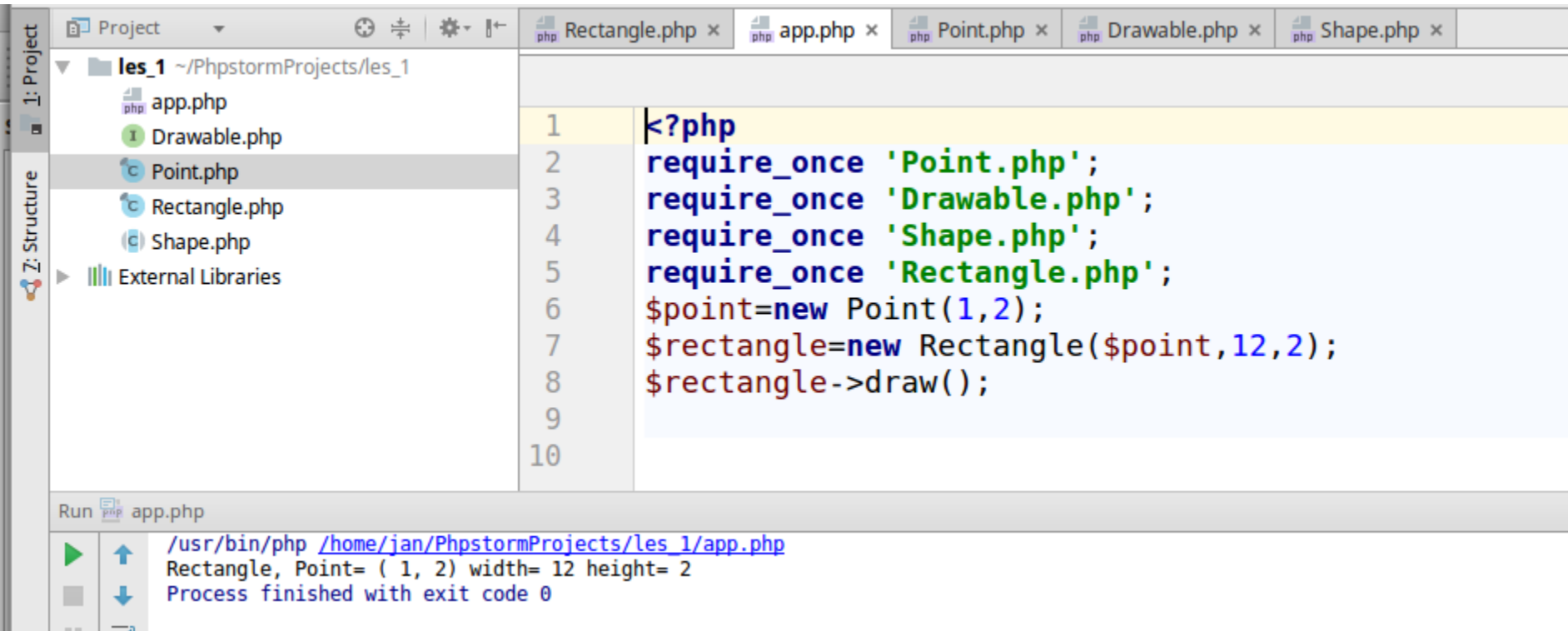
```
27 }
```

```
28 }
```

Afkomstig v.  
Shape

Afkomstig v.  
Drawable

# Interface



The screenshot displays the PhpStorm IDE interface. On the left, the 'Project' and 'Structure' toolbars are visible. The 'Project' toolbar shows a folder icon and a dropdown menu. The 'Structure' toolbar shows a folder icon and a dropdown menu. The main editor area shows a project named 'les\_1' with the following files: 'app.php', 'Drawable.php', 'Point.php', 'Rectangle.php', and 'Shape.php'. The 'app.php' file is selected, and its code is displayed in the editor. The code is as follows:

```
1 <?php
2 require_once 'Point.php';
3 require_once 'Drawable.php';
4 require_once 'Shape.php';
5 require_once 'Rectangle.php';
6 $point=new Point(1,2);
7 $rectangle=new Rectangle($point,12,2);
8 $rectangle->draw();
9
10
```

At the bottom, the 'Run' toolbar is visible, showing a green play button and a blue up arrow. The output window shows the following text:

```
/usr/bin/php /home/jan/PhpstormProjects/les_1/app.php
Rectangle, Point= ( 1, 2) width= 12 height= 2
Process finished with exit code 0
```

# Namespaces

~ Java packages

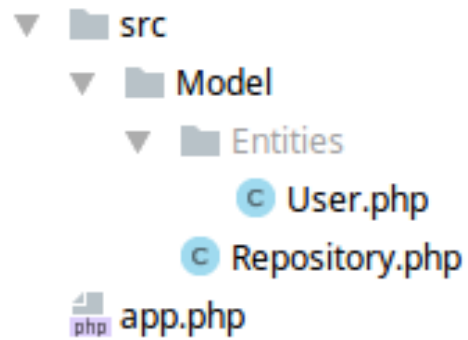
Nut:

- naming collisions vermijden

- unieke identificatie van klassen / interfaces

Conventie:

- namespaces komen overeen met directory-structuur



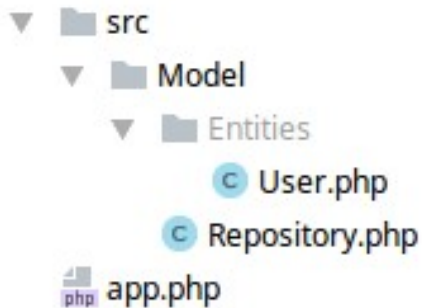
### src/Model/Repository.php

```
<?php namespace Model;  
  
class Repository  
{  
    ...  
}
```

### src/Model/Entities/User.php

```
<?php namespace Model\Entities;  
  
class User  
{  
    ...  
}
```

# Namespaces



In het bestand `app.php` wordt geen namespace gedefinieerd. Alle code in dit bestand hoort in de default-namespace (`\`).

## `app.php`

```
<?php
require_once 'src/Model/Repository.php';
require_once 'src/Model/Entities/User.php';
$user = new Model\Entities\User();
$repository = new Model\Repository();
```

# Namespaces

use: verkorte notatie

app.php

```
<?php
require_once 'src/Model/Repository.php';
require_once 'src/Model/Entities/User.php';
$user = new Model\Entities\User();
$repository = new Model\Repository();
```



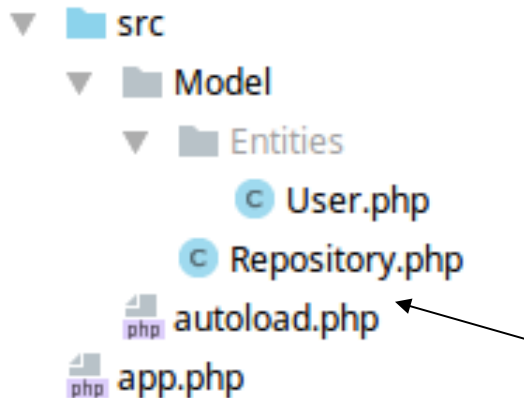
app.php

```
<?php
require_once 'src/Model/Repository.php';
require_once 'src/Model/Entities/User.php';
use Model\Repository;
use Model\Entities\User;
$user = new User();
$repository = new Repository();
```



# Autoloading

Een autoloader (src/autoload.php) vereenvoudigt het ophalen van code:



## app.php

```
<?php
require_once 'src/autoload.php';
use Model\Repository;
use Model\Entities\User;
$user = new User();
$repository = new Repository();
```

# Autoloading

## src/autoload.php

```
<?php
if (!function_exists('classAutoLoader')) {
    function classAutoLoader($className)
    {
        $fileName = 'src/'.
                    str_replace('\\', '/', $className).
                    '.php';
        if(file_exists($fileName)) {
            require_once $fileName;
        }
    }
}
spl_autoload_register('classAutoLoader');
```

Voor elke niet gekende klasse wordt de functie classAutoLoader aangeroepen. In deze functie wordt

Model\Repository

Model\Entities

omgezet naar

omgezet naar

src/Model/Repository.php

src/Model/Entities/User.php

# Autoloading: composer



Dependency Manager for PHP

Dependencies (bv Monolog) downloaden van repository

Maakt ook autoloader voor afgehaalde code (en eigen code).

## composer.json

```
{
    "autoload": {
        "psr-4": {
            "Model\\": "src/Model/",
            "Model\\Entities\\": "src/Model/Entities"
        }
    }
}
```

jan@laptop-jan ~/Desktop/werk/vagrants/vagrant\_webadv

File Edit View Search Terminal Help

vagrant@webadv:/var/www/html\$ composer dump-autoload -o

You are running composer with xdebug enabled. This has a major impact on runtime performance. See <https://getcomposer.org/xdebug>

Generating optimized autoload files

## app.php

```
<?php
require_once 'vendor/autoload.php';
use Model\Repository;
use Model\Entities\User;
$user = new User();
$repository = new Repository();
```

<https://getcomposer.org/doc/01-basic-usage.md#autoloading>