Web Adv

PHP: namespaces

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Namespaces

- Toepassing is samengesteld uit code van verschillende bronnen

- naming collisions, oplossing: namespaces

- gebruik van namespaces: absolute en relatieve notatie
  (cfr. absolute en relatieve paden in directory-structuur)

- use (importeren van klassen)

PXL IT

Namespaces

Definitie van een namespace:

```
namespace Alpha;
```

Definitie van een sub-namespaces:

```
namespace Alpha\Beta;
namespace Alpha\Beta\Gamma;
```

PXL IT

Namespaces

A.php

```php
<?php
namespace Alpha;
class A
{
    public static function fa()
    {
        echo 'a';
    }
}

namespace Alpha\Beta;
class AB
{
    public static function fab()
    {
        echo 'ab';
    }
}
```

namespace Alpha

namespace Alpha\Beta

Namespaces

TestA.php

```
<?php
require('A.php');
Alpha\A::fa();
\Alpha\A::fa();
Alpha\Beta\AB::fab();
\Alpha\Beta\AB::fab();
```

Geen namespace gedefinieerd in TestA.php
Alle code van TestA.php staat in de default namespace (\).

Relatieve notatie (hier relatief tov \)

```
Alpha\A::fa()
Alpha\Beta\AB::fab()
```

Absolute notatie:

```
\Alpha\A::fa()
\Alpha\Beta\AB::fab()
```

PXL IT

Namespaces

TestA2.php

```
<?php
namespace gamma;
require('A.php');
\Alpha\A::fa();
\Alpha\Beta\AB::fab();
```

Namespace gamma gedefinieerd in TestA2.php

Absolute notatie:

```
\Alpha\A::fa()
\Alpha\Beta\AB::fab()
```

PXL IT

Namespaces

TestA3.php

```
<?php
namespace Alpha;
require('A.php');
A::fa();
\Alpha\A::fa();
Beta\AB::fab();
\Alpha\Beta\AB::fab();
```

Namespace Alpha gedefinieerd in TestA3.php

Relatieve notatie (hier relatief tov Alpha)

```
A::fa()
Beta\AB::fab()
```

Absolute notatie:

```
\Alpha\A::fa()
\Alpha\Beta\AB::fab()
```

PXL IT

Namespaces

TestA4.php

```
<?php
namespace Alpha\Beta;
require('A.php');
\Alpha\A::fa();
AB::fab();
\Alpha\Beta\AB::fab();
```

Namespace Alpha\Beta gedefinieerd in TestA4.php

Relatieve notatie (hier relatief tov Alpha\Beta)

```
AB::fab();
```

Absolute notatie:

```
\Alpha\A::fa()
\Alpha\Beta\AB::fab();
```

PXL IT

PrintDate.php

```php
<?php
namespace Delta;
class PrintDate
{
   public static function print()
   {
      $date = new \DateTime('Now',
      new \DateTimeZone("Europe/Brussels"));
      echo $date->format(\DateTime::ISO8601);
   }
}
```

DateTime, DateTimeZone in default namespace!

TestPrintDate.php

```php
<?php
require('PrintDate.php');
\Delta\PrintDate::print();
Delta\PrintDate::print();
```

Namespaces

use: importeer een klasse

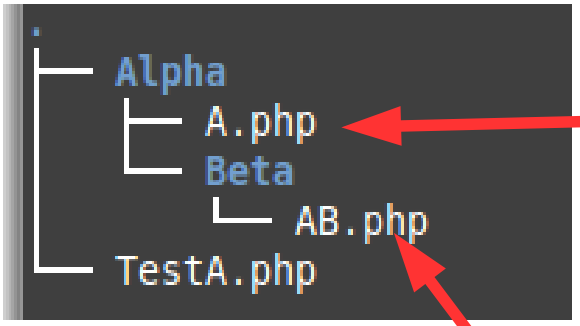
```
<?php
use Alpha\Beta\AB;
use Alpha\A;
require('A.php');
A::fa();
AB::fab();
```

use ... as ... : importeer een klasse en gebruik een andere naam

```
<?php
use \Alpha\Beta\AB as Ok;
require('A.php');

Ok::fab();
```

Namespaces

Conventie: structuur namespaces legt directory-structuur vast (cfr packages in Java)



namespace Alpha ~ directory Alpha

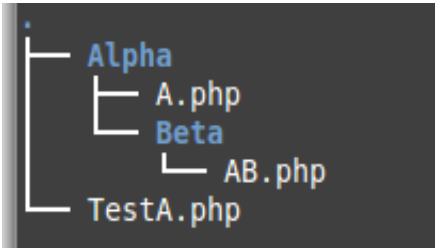
```
<?php
namespace Alpha;
class A
{
```

namespace Alpha\Beta ~ directory Alpha/Beta

```
<?php
namespace Alpha\Beta;
class AB
{
```

PXL IT

Autoloading



- function __autoload wordt aangeroepen als een ongekende klasse aangeroepen wordt.

```
<?php
function __autoload($className)
{
   $fileName =  str_replace('\\',
         DIRECTORY_SEPARATOR,
         $className) . '.php';
   require($fileName);
}
Alpha\A::fa();
Alpha\Beta\AB::fab();
```

PXL IT

Oefening

Vertaal de code op de volgende slides van Java naar PHP.

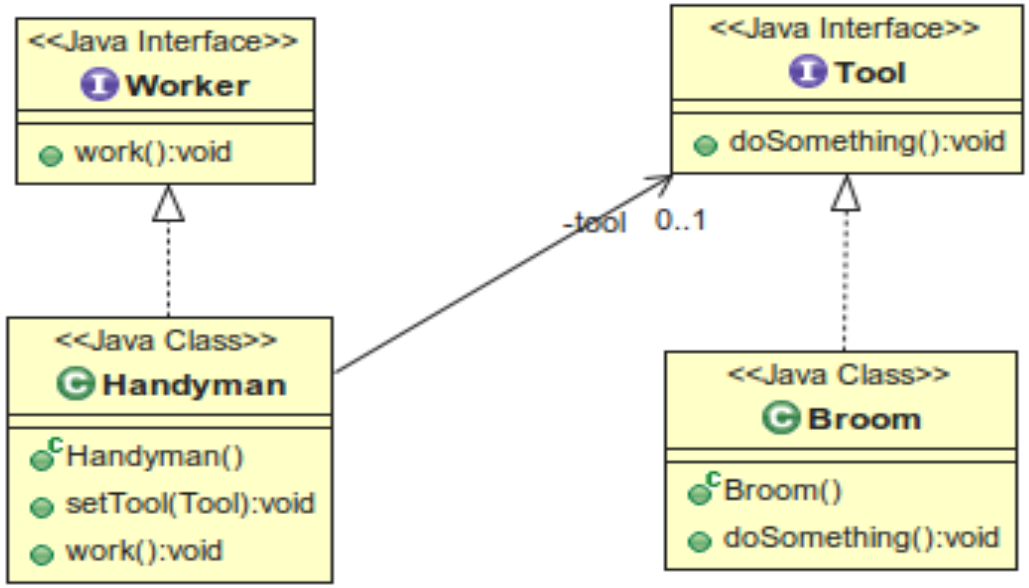
Maak gebruik van namespace Workers en plaats de code voor Handyman, Worker, Broom en Tool in de map Workers. Plaats Test.php in de basismap van je project (de map waar ook Workers in staat).

Maak in Test.php gebruik van autoloading.

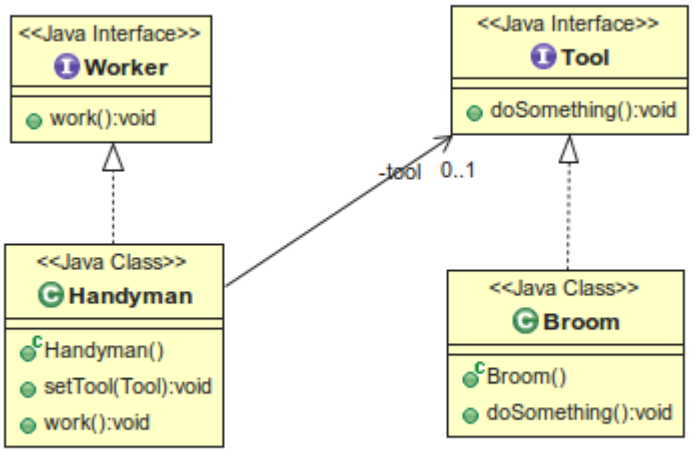
Maak gebruik van type-hinting in de methode setTool

PXL IT

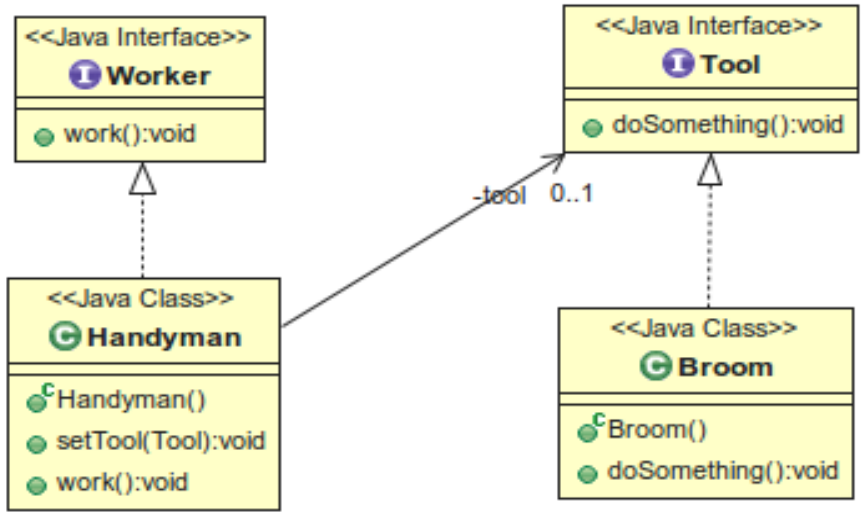
Voorbeeld setter injection



```java
package be.pxl.workers;
public interface Tool {
    public void doSomething();
}

package be.pxl.workers;
public class Broom implements Tool {
    public void doSomething() {
        System.out.println("Sweep");
    }
}

package be.pxl.workers;
public interface Worker {
    public void work();
}

package be.pxl.workers;
public class Handyman implements Worker {
    private Tool tool;

    public void setTool(Tool tool) {
        this.tool = tool;
    }
    public void work() {
        tool.doSomething();
    }
}
```



```
public class Main {
    public static void main(String[] args) {
        Broom b = new Broom();
        Handyman harry = new Handyman();
        harry.setTool(b);

        harry.work();

    }
}
```



Handyman kan met elke Tool werken
Handyman verliest de **controle** over zijn gereedschap aan Main **Inversion of control**

Bronnen

http://php.net/manual/en/language.namespaces.php

http://www.sitepoint.com/php-53-namespaces-basics/