

Développer un site web dynamique en PHP

Module 3 : La Programmation Orientée Objet en PHP



Objectifs

- Appréhender la syntaxe de la programmation orientée objet (POO) en PHP
- Comprendre l'utilité et l'apport de la POO en PHP

L'histoire de la POO en PHP

- PHP 3
 - Implémentation « rapide » de la POO
- PHP 4
 - Légère amélioration du modèle objet
- PHP 5
 - Révision complète
 - Introduction d'un modèle objet complet

La Programmation Orientée Objet en PHP

La création d'une classe

Déclaration d'une classe

```
<?php
class Utilisateur {

    private $nom; // nom de l'utilisateur
    private $prenom; // prénom de l'utilisateur

    public function __construct($prenom, $nom) {
        $this->prenom = $prenom;
        $this->nom = $nom;
    }

    public function __destruct() {
        echo '<p><b>Suppression de ' . $this->nom . '</b></p>';
    }

    public function informations() {
        return 'Nom : ' . $this->nom . '<br>Prénom : ' . $this->prenom . '<br>';
    }

    public function __toString() {
        return strtolower($this->prenom) . ' ' . ucwords(strtolower($this->nom));
    }
}
```

| Utilisateur |
|--|
| -nom -prenom |
| + __construct(nom, prenom) + __destruct() + informations() : String + __toString() : String |

Les classes – les règles de nommage

- Règles de nommage
 - Mêmes règles de nommage que pour les fonctions
- Conventions de nommage
 - Commence par une majuscule, le reste en CamelCase
 - Chaque classe est dans un fichier PHP ayant pour nom le nom de la classe suivi de l'extension `.class.php`

La Programmation Orientée Objet en PHP

La création d'une classe

Déclaration des attributs

```
<?php
class Utilisateur {

    private $nom; // nom de l'utilisateur
    private $prenom; // prénom de l'utilisateur

    public function __construct($prenom, $nom) {
        $this->prenom = $prenom;
        $this->nom = $nom;
    }

    public function __destruct() {
        echo '<p><b>Suppression de ' . $this->nom . '</b></p>';
    }

    public function informations() {
        return 'Nom : ' . $this->nom . '<br>Prénom : ' . $this->prenom . '<br>';
    }

    public function __toString() {
        return strtolower($this->prenom) . ' ' . ucwords(strtolower($this->nom));
    }
}
```

| Utilisateur |
|--|
| -nom -prenom |
| +__construct(nom, prenom) +__destruct() +informations() : String +__toString() : String |

Les attributs d'instance


- Visibilité

- `private` : accessible uniquement au sein de la classe
- `protected` : accessible au sein de la classe et des classes qui en héritent

- Nom

- Comme pour une variable

```
private $nom; // nom de l'utilisateur  
private $prenom; // prénom de l'utilisateur
```



En vertu du principe
d'encapsulation,
il ne serait pas correct
d'avoir une visibilité publique

La Programmation Orientée Objet en PHP

La création d'une classe

Déclaration du constructeur

```
<?php
class Utilisateur {

    private $nom; // nom de l'utilisateur
    private $prenom; // prénom de l'utilisateur

    public function __construct($prenom, $nom) {
        $this->prenom = $prenom;
        $this->nom = $nom;
    }

    public function __destruct() {
        echo '<p><b>Suppression de ' . $this->nom . '</b></p>';
    }

    public function informations() {
        return 'Nom : ' . $this->nom . '<br>Prénom : ' . $this->prenom . '<br>';
    }

    public function __toString() {
        return strtolower($this->prenom) . ' ' . ucwords(strtolower($this->nom));
    }
}
```

| Utilisateur |
|--|
| -nom -prenom |
| + __construct(nom, prenom) + __destruct() + informations() : String + __toString() : String |

Les constructeurs

- Visibilité

- `private` : constructeur accessible uniquement au sein de cette classe
- `protected` : constructeur accessible au sein de la classe et des classes qui en héritent
- `public` : constructeur accessible partout

- Mot clé `function`

- Nom `__construct`

- Pas le choix du nom
- Méthode magique
- Pas d'appel direct à cette méthode

- Paramètres possibles

```
public function __construct($prenom, $nom) {  
    $this->prenom = $prenom;  
    $this->nom = $nom;  
}
```

L'accès aux éléments d'instance

- Instance courante
 - Utilisation de la pseudo variable `$this`
- Opérateur de l'objet (flèche) `->`
 - Permet d'accéder aux éléments d'instances
- Obligatoire pour accéder aux attributs d'instance
 - `$this->`

```
public function __construct($prenom, $nom) {  
    $this->prenom = $prenom;  
    $this->nom = $nom;  
}
```

La Programmation Orientée Objet en PHP

La création d'une classe

```
<?php
class Utilisateur {

    private $nom; // nom de l'utilisateur
    private $prenom; // prénom de l'utilisateur

    public function __construct($prenom, $nom) {
        $this->prenom = $prenom;
        $this->nom = $nom;
    }

    public function __destruct() {
        echo '<p><b>Suppression de ' . $this->nom . '</b></p>';
    }

    public function informations() {
        return 'Nom : ' . $this->nom . '<br>Prénom : ' . $this->prenom . '<br>';
    }

    public function __toString() {
        return strtolower($this->prenom) . ' ' . ucwords(strtolower($this->nom));
    }
}
```

| Utilisateur |
|--|
| -nom -prenom |
| + __construct(nom, prenom) + __destruct() + informations() : String + __toString() : String |

Déclaration du destructeur

Le destructeur

- Méthode appelée lors de la destruction de l'instance
- Visibilité
 - `public` : destructeur accessible partout
- Mot clé `function`
- Nom `__destruct`
 - Pas le choix du nom
 - Méthode magique
 - Pas d'appel direct à cette méthode
- Pas de paramètre

```
public function __destruct() {  
    echo '<p><b>Suppression de ' . $this->nom . '</b></p>';  
}
```

La Programmation Orientée Objet en PHP

La création d'une classe

```
<?php
class Utilisateur {

    private $nom; // nom de l'utilisateur
    private $prenom; // prénom de l'utilisateur

    public function __construct($prenom, $nom) {
        $this->prenom = $prenom;
        $this->nom = $nom;
    }

    public function __destruct() {
        echo '<p><b>Suppression de ' . $this->nom . '</b></p>';
    }

    public function informations() {
        return 'Nom : ' . $this->nom . '<br>Prénom : ' . $this->prenom . '<br>';
    }

    public function __toString() {
        return strtolower($this->prenom) . ' ' . ucwords(strtolower($this->nom));
    }
}
```

**Déclaration
d'une méthode
d'instance**

| Utilisateur |
|--|
| -nom -prenom |
| +__construct(nom, prenom) +__destruct() +informations() : String +__toString() : String |

Les méthodes d'instance

- Visibilité

- `private` : méthode accessible uniquement au sein de cette classe
- `protected` : méthode accessible au sein de la classe et des classes qui en héritent
- `public` : méthode accessible partout

- Mot clé `function`

- Nom de la méthode (mêmes règles de nommage que pour les fonctions)

- Paramètres possibles

```
public function informations() {  
    return 'Nom : ' . $this->nom . '<br>Prénom : ' .  
        $this->prenom . '<br>';  
}
```

La Programmation Orientée Objet en PHP

La création d'une classe

```
<?php
class Utilisateur {

    private $nom; // nom de l'utilisateur
    private $prenom; // prénom de l'utilisateur

    public function __construct($prenom, $nom) {
        $this->prenom = $prenom;
        $this->nom = $nom;
    }

    public function __destruct() {
        echo '<p><b>Suppression de ' . $this->nom . '</b></p>';
    }

    public function informations() {
        return 'Nom : ' . $this->nom . '<br>Prénom : ' . $this->prenom . '<br>';
    }

    public function __toString() {
        return strtolower($this->prenom) . ' ' . ucwords(strtolower($this->nom));
    }
}
```

| Utilisateur |
|--|
| -nom -prenom |
| +__construct(nom, prenom) +__destruct() +informations() : String +__toString() : String |

**Déclaration
d'une méthode
« magique »**

Les méthodes magiques

- Méthode d'instance

- Le nom de la méthode commence par deux caractères _

- `__construct()`, `__destruct()`, `__call()`, `__callStatic()`, `__get()`, `__set()`, `__isset()`, `__unset()`, `__sleep()`, `__wakeup()`, `__toString()`, `__invoke()`, `__set_state()`, `__clone()` **et** `__debugInfo()`

- `__toString()`

- La méthode est appelée automatiquement lorsque l'instance est traitée comme une chaîne de caractères

```
public function __toString() {  
    return strtolower($this->prenom) . ' ' . ucwords(strtolower($this->nom));  
}
```


La Programmation Orientée Objet en PHP

L'utilisation de la classe

```
<?php
require_once '../classes/Utilisateur.class.php';

$jean = new Utilisateur('jean', 'de la fontaine');

echo $jean->informations() . '<br>';

echo 'Appel explicite : ' . $jean->__toString() . '<br>';
echo 'Appel implicite : ' . $jean . '<br>';
```

| Utilisateur |
|--|
| -nom -prenom |
| +__construct(nom, prenom) +__destruct() +informations() : String +__toString() : String |

| jean:Utilisateur |
|---|
| -nom = 'de la fontaine' -prenom = 'jean' |



Nom : de la fontaine
Prénom : jean
Appel explicite : jean De La Fontaine
Appel implicite : jean De La Fontaine
Suppression de de la fontaine

La Programmation Orientée Objet en PHP

L'héritage

Déclaration de l'héritage

```
<?php
require_once 'Utilisateur.class.php';

class UtilisateurAvecCouleurs extends Utilisateur {

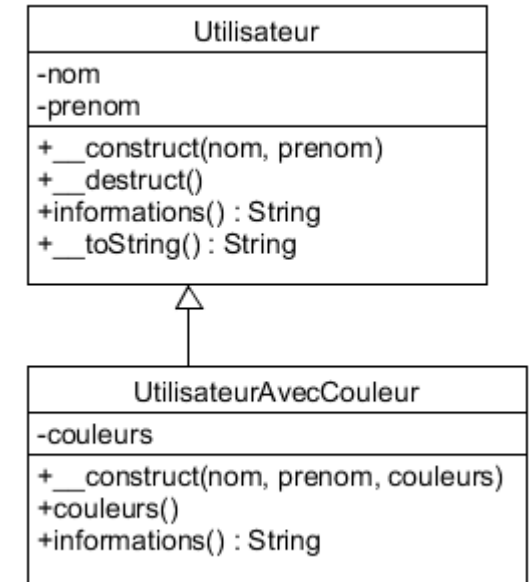
    private $couleurs; // couleurs préférées de l'utilisateur

    public function __construct($prenom, $nom, $couleurs) {
        parent::__construct($prenom, $nom);
        $this->couleurs = explode(',', $couleurs);
    }

    public function couleurs() {
        return implode('-', $this->couleurs);
    }

    public function informations() {
        return parent::informations() . 'Couleurs : ' . $this->couleurs() . '<br>';
    }

}
```



La Programmation Orientée Objet en PHP

L'héritage

Attributs supplémentaires

```
<?php
require_once 'Utilisateur.class.php';

class UtilisateurAvecCouleurs extends Utilisateur {

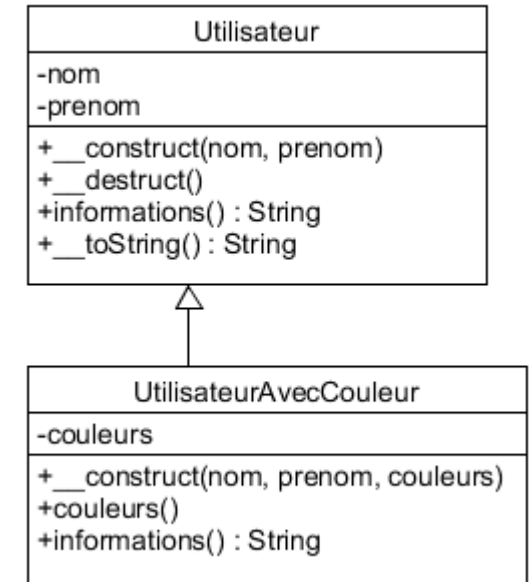
    private $couleurs; // couleurs préférées de l'utilisateur

    public function __construct($prenom, $nom, $couleurs) {
        parent::__construct($prenom, $nom);
        $this->couleurs = explode(',', $couleurs);
    }

    public function couleurs() {
        return implode('-', $this->couleurs);
    }

    public function informations() {
        return parent::informations() . 'Couleurs : ' . $this->couleurs() . '<br>';
    }

}
```



La Programmation Orientée Objet en PHP

L'héritage

```
<?php
require_once 'Utilisateur.class.php';

class UtilisateurAvecCouleurs extends Utilisateur {

    private $couleurs; // couleurs préférées de l'utilisateur

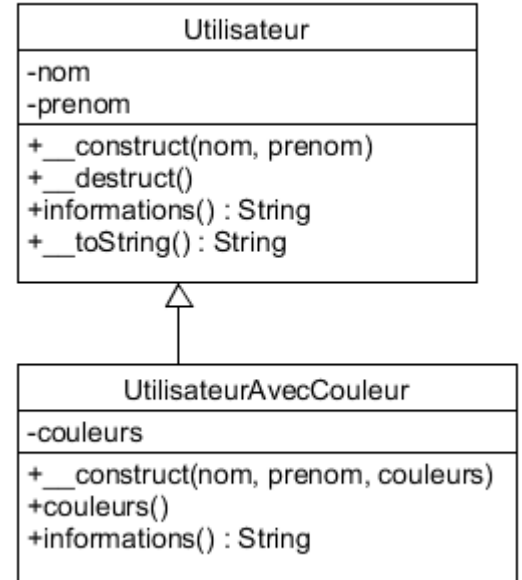
    public function __construct($prenom, $nom, $couleurs) {
        parent::__construct($prenom, $nom);
        $this->couleurs = explode(',', $couleurs);
    }

    public function couleurs() {
        return implode('-', $this->couleurs);
    }

    public function informations() {
        return parent::informations() . 'Couleurs : ' . $this->couleurs() . '<br>';
    }

}
```

**Appel au
constructeur
de la classe parent**



La Programmation Orientée Objet en PHP

L'héritage

```
<?php
require_once 'Utilisateur.class.php';

class UtilisateurAvecCouleurs extends Utilisateur {

    private $couleurs; // couleurs préférées de l'utilisateur

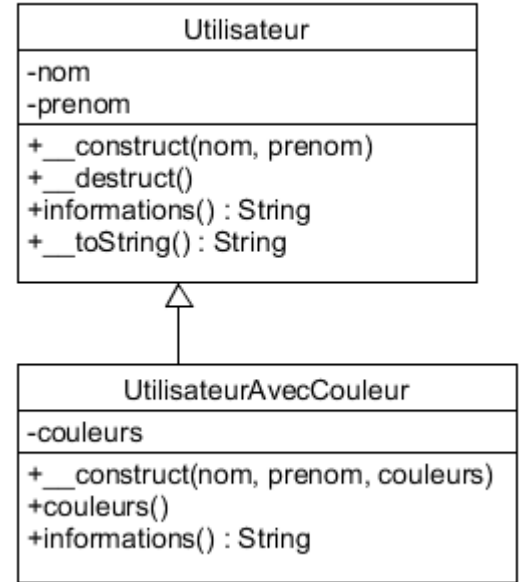
    public function __construct($prenom, $nom, $couleurs) {
        parent::__construct($prenom, $nom);
        $this->couleurs = explode(',', $couleurs);
    }

    public function couleurs() {
        return implode('-', $this->couleurs);
    }

    public function informations() {
        return parent::informations() . 'Couleurs : ' . $this->couleurs() . '<br>';
    }

}
```

**Méthodes
supplémentaires**



La Programmation Orientée Objet en PHP

L'héritage

```
<?php
require_once 'Utilisateur.class.php';

class UtilisateurAvecCouleurs extends Utilisateur {

    private $couleurs; // couleurs préférées de l'utilisateur

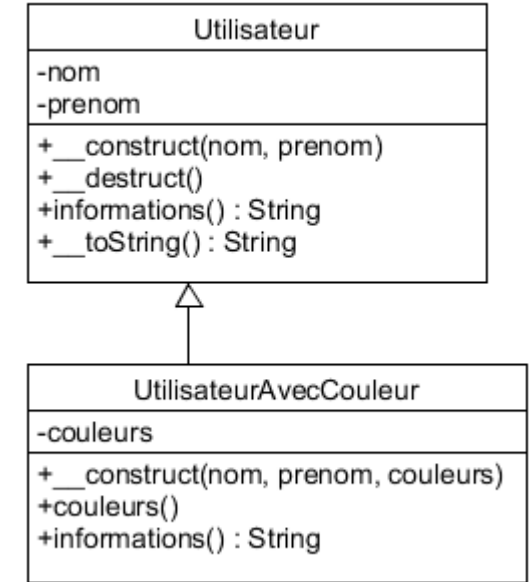
    public function __construct($prenom, $nom, $couleurs) {
        parent::__construct($prenom, $nom);
        $this->couleurs = explode(',', $couleurs);
    }

    public function couleurs() {
        return implode('-', $this->couleurs);
    }

    public function informations() {
        return parent::informations() . 'Couleurs : ' . $this->couleurs() . '<br>';
    }

}
```

**Méthodes
substituées**




La Programmation Orientée Objet en PHP

L'utilisation de la classe fille

```
<?php
require_once '../classes/UtilisateurAvecCouleurs.class.php';

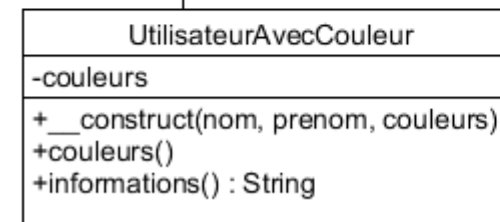
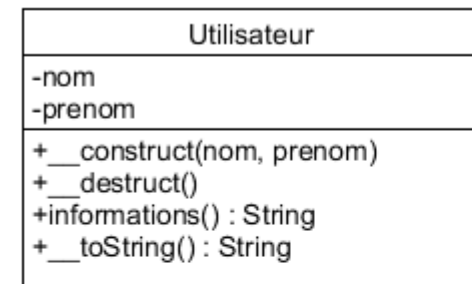
$louis = new UtilisateurAvecCouleurs('louis', 'lumière', 'rouge,vert,bleu');

echo '<p><code>$louis->__toString()</code> :<br>' .
    $louis->__toString() . '<br>'; // existe par héritage
echo '<code>$louis->couleurs()</code> :<br>' .
    $louis->couleurs() . '<br>'; // existe dans la classe
echo '<code>$louis->informations()</code> :<br>' .
    $louis->informations() . '</p>'; // méthode substituée
```



```
$louis->__toString() : louis Lumière
$louis->couleurs() : rouge-vert-bleu
$louis->informations() :
Nom : lumière
Prénom : louis
Couleurs : rouge-vert-bleu
```

Suppression de lumière



| louis:UtilisateurAvecCouleur |
|--|
| -nom = 'lumière' -prenom = 'louis' -couleurs = ['rouge', 'vert', 'bleu'] |

La Programmation Orientée Objet en PHP

Les éléments de classe

```
<?php
require_once 'Utilisateur.class.php';

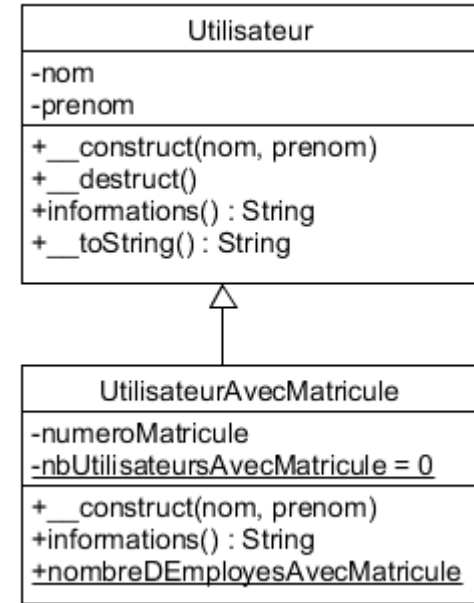
class UtilisateurAvecMatricule extends Utilisateur {
    private static $nbUtilisateursAvecMatricule = 0;
    private $numeroMatricule;

    public function __construct($prenom, $nom) {
        parent::__construct($prenom, $nom);
        static::$nbUtilisateursAvecMatricules++;
        $this->numeroMatricule = static::$nbUtilisateursAvecMatricules;
    }

    public function informations() {
        return parent::informations() .
            'Numéro de matricule : ' . $this->numeroMatricule . '<br>';
    }

    public static function nombreDEmpoyesAvecMatricule() {
        return static::$nbUtilisateursAvecMatricules;
    }
}
```

Déclaration d'un attribut de classe



La Programmation Orientée Objet en PHP

Les éléments de classe

```
<?php
require_once 'Utilisateur.class.php';

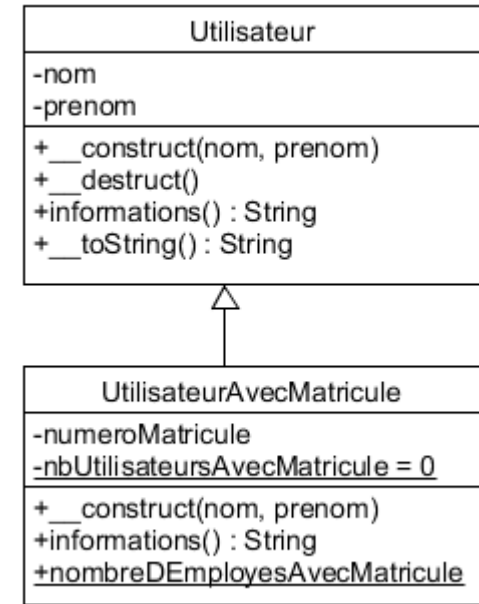
class UtilisateurAvecMatricule extends Utilisateur {
    private static $nbUtilisateursAvecMatricule = 0;
    private $numeroMatricule;

    public function __construct($prenom, $nom) {
        parent::__construct($prenom, $nom);
        static::$nbUtilisateursAvecMatricules++;
        $this->numeroMatricule = static::$nbUtilisateursAvecMatricules;
    }

    public function informations() {
        return parent::informations() .
            'Numéro de matricule : ' . $this->numeroMatricule . '<br>';
    }

    public static function nombreDEmpoyesAvecMatricule() {
        return static::$nbUtilisateursAvecMatricules;
    }
}
```

Accès à un
attribut de classe



La Programmation Orientée Objet en PHP

Les éléments de classe

```
<?php
require_once 'Utilisateur.class.php';

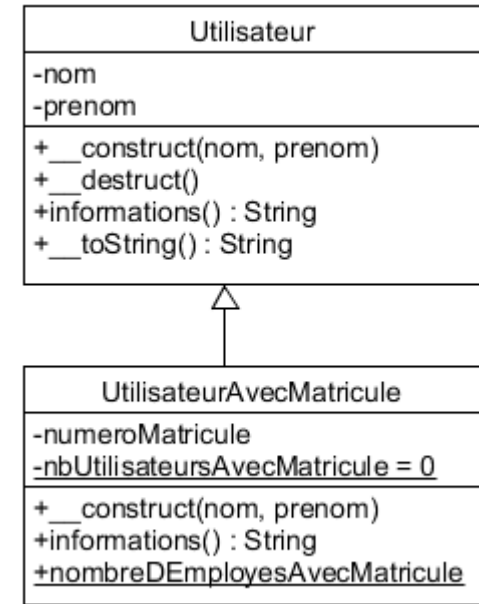
class UtilisateurAvecMatricule extends Utilisateur {
    private static $nbUtilisateursAvecMatricule = 0;
    private $numeroMatricule;

    public function __construct($prenom, $nom) {
        parent::__construct($prenom, $nom);
        static::$nbUtilisateursAvecMatricules++;
        $this->numeroMatricule = static::$nbUtilisateursAvecMatricules;
    }

    public function informations() {
        return parent::informations() .
            'Numéro de matricule : ' . $this->numeroMatricule . '<br>';
    }

    public static function nombreDEmpoyesAvecMatricule() {
        return static::$nbUtilisateursAvecMatricules;
    }
}
```

**Déclaration d'une
méthode de classe**



La Programmation Orientée Objet en PHP

Les éléments de classe

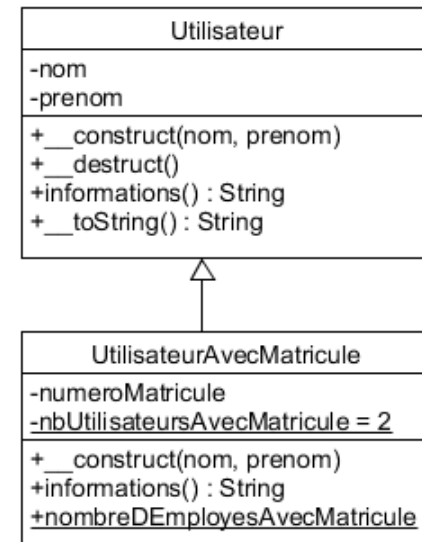
```
<?php
require_once '../classes/UtilisateurAvecMatricule.class.php';
echo 'nombre d\'utilisateurs avec matricule : ' . UtilisateurAvecMatricule::nombreDEmpoyesAvecMatricule() . '<br>';
$antoine = new UtilisateurAvecMatricule('Antoine', 'Lavoisier');
echo $antoine->informations() . '<br>';
echo 'nombre d\'utilisateurs avec matricule : ' . UtilisateurAvecMatricule::nombreDEmpoyesAvecMatricule() . '<br>';
$albert = new UtilisateurAvecMatricule('Albert', 'Schweitzer');
echo $albert->informations() . '<br>';
echo 'nombre d\'utilisateurs avec matricule : ' . UtilisateurAvecMatricule::nombreDEmpoyesAvecMatricule() . '<br>';
```

nombre d'utilisateurs avec matricule : 0
Nom : Lavoisier
Prénom : Antoine
Numéro de matricule : 1



nombre d'utilisateurs avec matricule : 1
Nom : Schweitzer
Prénom : Albert
Numéro de matricule : 2

nombre d'utilisateurs avec matricule : 2



antoine:UtilisateurAvecMatricule
-nom = 'Lavoisier'
-prenom = 'Antoine'
-numeroMatricule = 1

albert:UtilisateurAvecMatricule
-nom = 'Schweitzer'
-prenom = 'Albert'
-numeroMatricule = 2

Les classes abstraites

**Déclaration d'une
classe abstraite**

```
<?php
abstract class Animal {

    private $espece;

    public function __toString() {
        return 'je suis un ' . $this->espece . '<br>';
    }

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

    public abstract function mange();
}
```

| |
|--|
| «Abstract» <i>Animal</i> |
| -espece |
| +__construct(espece) +__toString():String +mange() |

La Programmation Orientée Objet en PHP

Les classes abstraites

```
<?php
abstract class Animal {

    private $espece;

    public function __toString() {
        return 'je suis un ' . $this->espece . ' .<br>';
    }

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

    public abstract function mange();
}
```

**Déclaration d'une
méthode abstraite**

| |
|--|
| «Abstract» <i>Animal</i> |
| -espece |
| +__construct(espece) +__toString():String +mange() |

La Programmation Orientée Objet en PHP

Les classes abstraites

```
<?php
require_once '../classes/Animal.class.php';

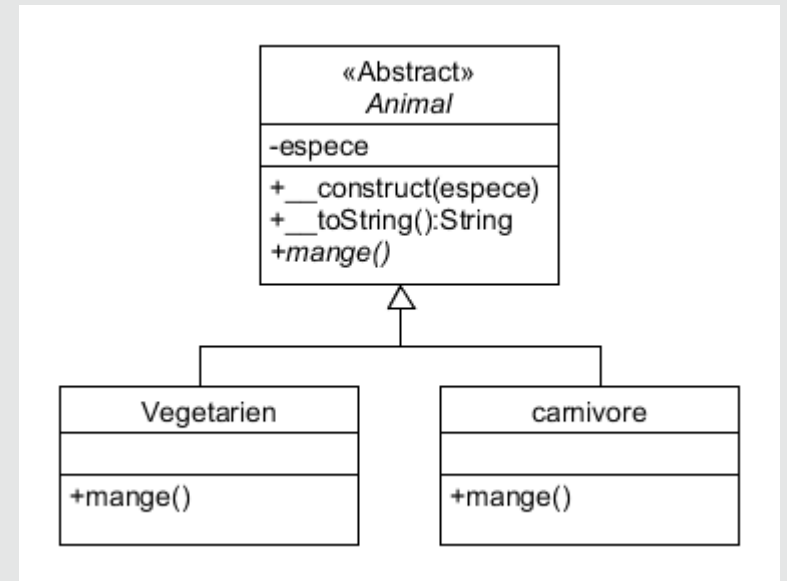
class Vegetarien extends Animal {

    public function manger() {
        echo 'je mange des fruits et des légumes.<br>';
    }
}
```

```
<?php
require_once '../classes/Animal.class.php';

class Carnivore extends Animal {

    public function manger() {
        echo 'je mange de la viande.<br>';
    }
}
```



La Programmation Orientée Objet en PHP

Les classes abstraites

```
<?php
require_once '../classes/Vegetarien.class.php';

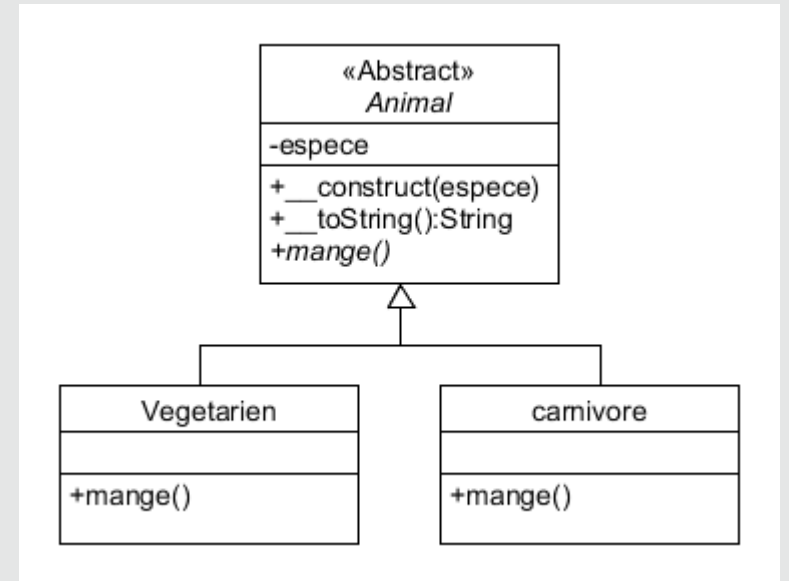
echo '<p>création d\'une instance de lapin<br>';
$lapin = new Vegetarien('Lapin');

echo '<code>$lapin</code> = ' . $lapin . '<br>';
echo 'Appel à <code>$lapin->mange()</code><br>';
$lapin->mange();
echo '</p>';
```



création d'une instance de lapin
`$lapin = je suis un Lapin.`

Appel à `$lapin->mange()`
je mange des fruits et des légumes.

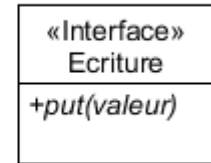
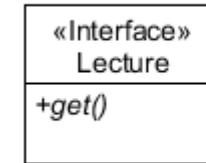


La Programmation Orientée Objet en PHP

Les interfaces

```
<?php  
interface Lecture {  
    function get();  
}
```

```
<?php  
interface Ecriture {  
    function put($valeur);  
}
```



La Programmation Orientée Objet en PHP

Les interfaces

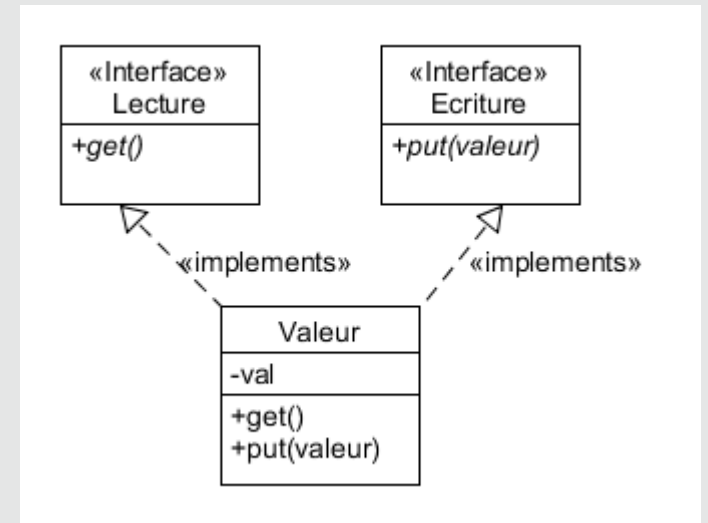
```
<?php
require_once '../interfaces/Lecture.interface.php';
require_once '../interfaces/Ecriture.interface.php';

class Valeur implements Lecture, Ecriture {

    private $val;

    // Implémentation de la méthode de Lecture.
    public function get() {
        return $this->val;
    }

    // Implémentation de la méthode d'Ecriture.
    public function put($valeur) {
        $this->val = $valeur;
    }
}
```



La Programmation Orientée Objet en PHP

Les interfaces

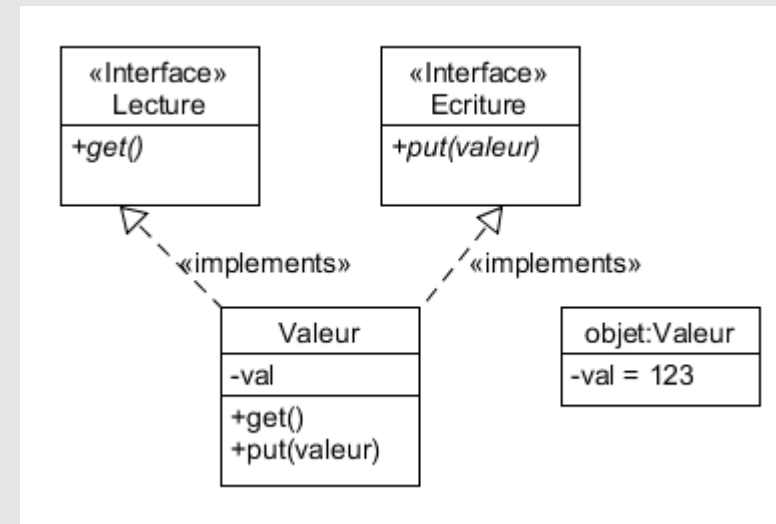
```
<?php
require_once '../classes/Valeur.class.php';

$objet = new Valeur();
echo '<p>Appel à la méthode <code>put()</code><br>';
$objet->put(123);
echo '<p>Appel à la méthode <code>get()</code> : ' .
      $objet->get() . '</p>';
```



Appel à la méthode `put()`

Appel à la méthode `get()` : 123



Les exceptions

```
<?php
class Distance {

    private $dist;

    public function __construct($dist) {
        if ($dist >= 0) {
            $this->dist = $dist;
        } else {
            throw new Exception('Une distance ne peut pas être négative !', 123);
        }
    }
}
```

La Programmation Orientée Objet en PHP

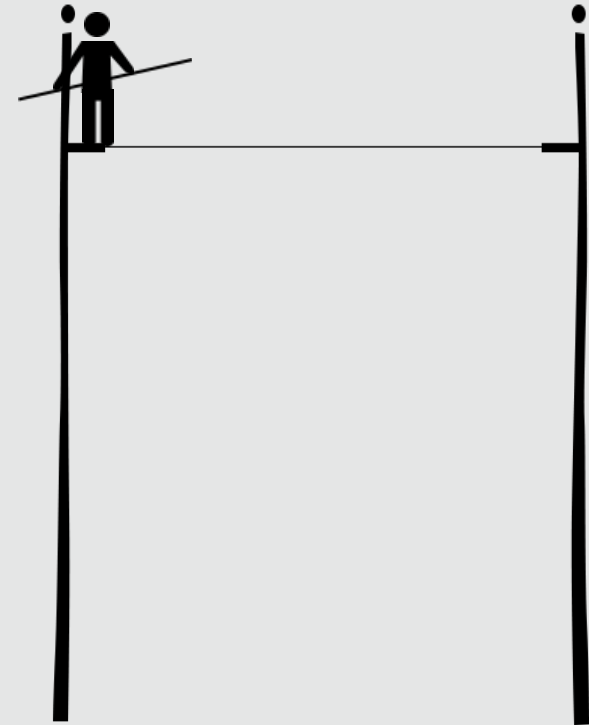
Les exceptions

```
<?php
require_once '../classes/Distance.class.php';

$i = mt_rand(-2, 2);
echo '<p> *** Création d\'une distance de ' . $i . ' ***<br>';
$d1 = new Distance($i);
echo 'La distance a été créée sans qu\'aucune exception n\'ait été levée<br>';
```



*** Création d'une distance de 1 ***
La distance a été créée sans qu'aucune exception n'ait été levée



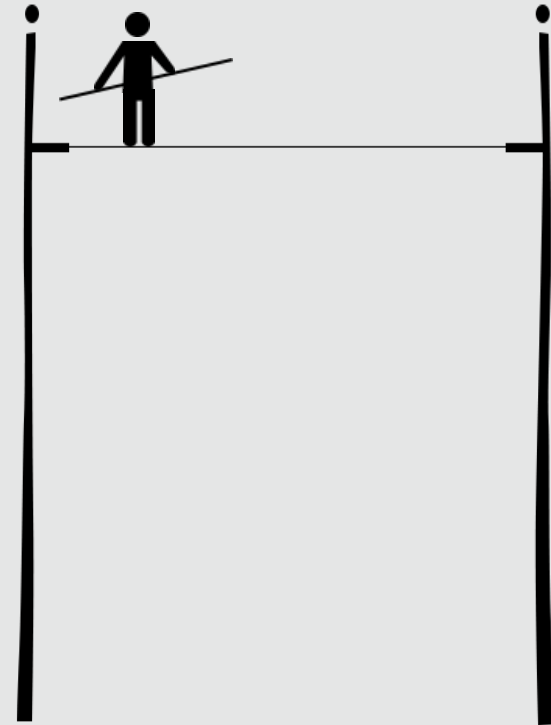
Les exceptions

```
<?php
require_once '../classes/Distance.class.php';

$i = mt_rand(-2, 2);
echo '<p> *** Création d\'une distance de ' . $i . ' ***<br>';
$d1 = new Distance($i);
echo 'La distance a été créée sans qu\'aucune exception n\'ait été levée<br>';
```



*** Création d'une distance de -2 ***
Fatal Error

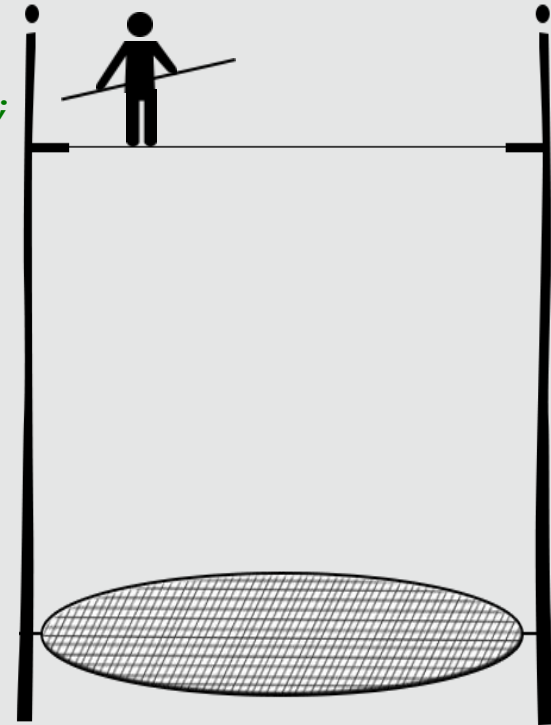


La Programmation Orientée Objet en PHP

Les exceptions

```
<?php
require_once '../classes/Distance.class.php';

for ($i = 0; $i > -3; $i--) {
    try {
        echo '<p> *** Création d\'une distance de ' . $i . ' ***<br>';
        $d1 = new Distance($i);
        echo 'La distance a été créée sans qu\'aucune exception n\'ait été levée<br>';
    } catch (Exception $e) {
        echo 'Une exception a été levée et récupérée :<br>';
        echo 'Numéro de l\'erreur : ' . $e->getCode() . '<br>';
        echo 'Message : ' . $e->getMessage() . '<br>';
    } finally {
        echo 'Qu\'une exception ait été levée ou pas de toutes les façons ce code ' .
            'sera exécuté</p>';
    }
}
```



Les exceptions

*** Création d'une distance de 0 ***

La distance a été créée sans qu'aucune exception n'ait été levée

Qu'une exception ait été levée ou pas de toutes les façons ce code sera exécuté

*** Création d'une distance de -1 ***

Une exception a été levée et récupérée :

Numéro de l'erreur : 123

Message : Une distance ne peut pas être négative !

Qu'une exception ait été levée ou pas de toutes les façons ce code sera exécuté

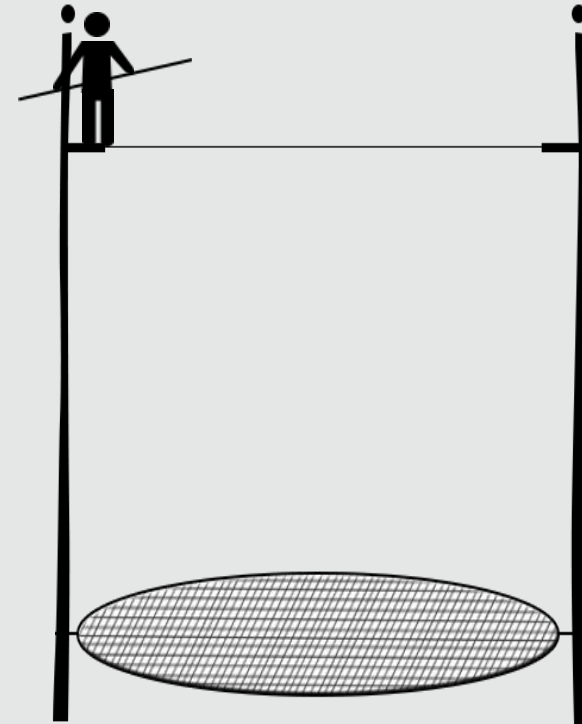
*** Création d'une distance de -2 ***

Une exception a été levée et récupérée :

Numéro de l'erreur : 123

Message : Une distance ne peut pas être négative !

Qu'une exception ait été levée ou pas de toutes les façons ce code sera exécuté



La Programmation Orientée Objet en PHP

Les espaces de nom

```
<?php
namespace monProjet;

class MaClasse {

    public function mafonction() {
        return '<br>Namespace : ' . __NAMESPACE__ . ', class : ' . __CLASS__ .
', fonction : ' . __FUNCTION__;
    }
}
```

```
<?php
namespace maLibrairieSpecifique;

class MaClasse {

    public function mafonction() {
        return '<br>Namespace : ' . __NAMESPACE__ . ', class : ' . __CLASS__ .
', fonction : ' . __FUNCTION__;
    }
}
```


Les espaces de nom

```
<?php
require_once './monProjet_MaClasse.class.php';
require_once './maLibrairieSpecifique_MaClasse.class.php';

// Déclaration d'une instance ma classe "maClasse" se trouvant dans le namespace MonProjet
$objProjet = new MonProjet\maClasse();
echo $objProjet->mafonction();
// Déclaration d'une instance ma classe "maClasse" se trouvant dans le namespace maLibrairieSpecifique
$objMaLivraison = new maLibrairieSpecifique\maClasse();
echo $objMaLivraison->mafonction();
```



Namespace : monProjet, class : monProjet\MaClasse, fonction : mafonction

Namespace : maLibrairieSpecifique, class : maLibrairieSpecifique\MaClasse, fonction : mafonction

La Programmation Orientée Objet en PHP

Les espaces de nom

```
<?php
require_once './monProjet_MaClasse.class.php';

// Création d'un alias sur la classe du namespace monProjet
use monProjet\MaClasse as Mc;

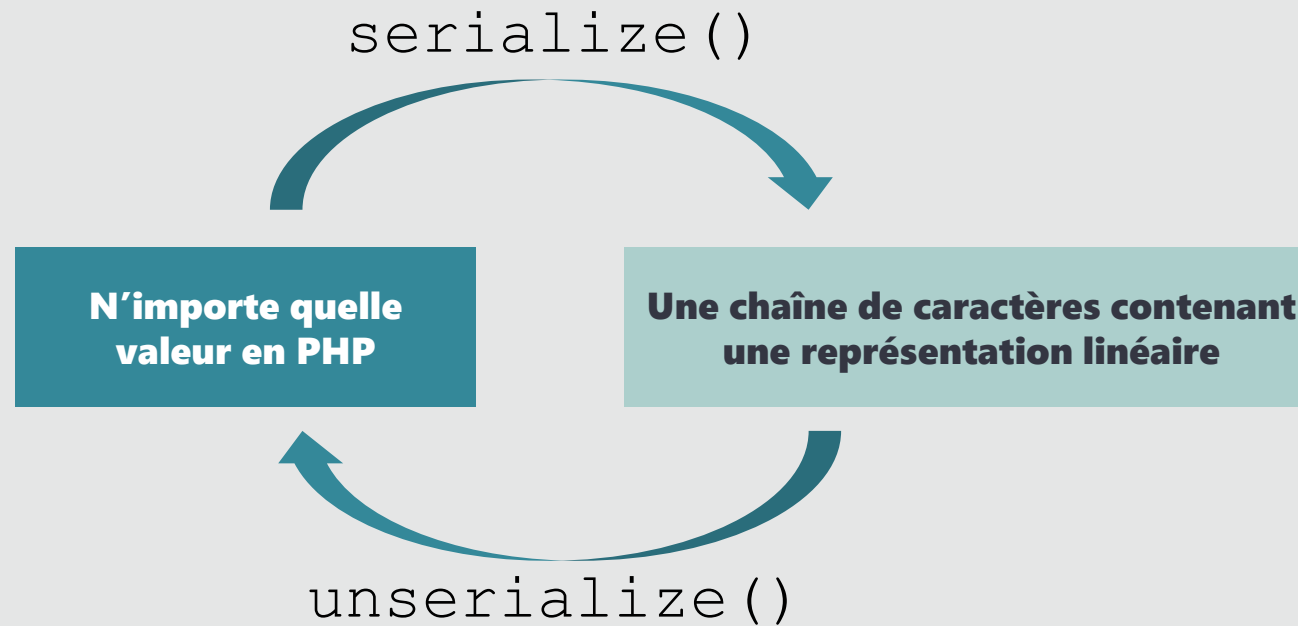
$objAlias = new Mc();
echo $objAlias->mafonction();
```



Namespace : monProjet, class : monProjet\MaClasse, fonction : mafonction

La sérialisation

- Objectif
 - Transformer n'importe quelle valeur en une représentation linéaire (une chaîne de caractères)



La sérialisation

```
<?php
$entier = 7;
$serial = serialize($entier);
var_dump($serial);
var_dump(unserialize($serial));
```



```
string 'i:7;' (length=4)
int 7
```

La sérialisation

```
<?php
$bool = false;
$serial = serialize($bool);
var_dump($serial);
var_dump(unserialize(serialize($bool)));
```



```
string 'b:0;' (length=4)
boolean false
```

La sérialisation

```
<?php
$float = 3.14159265359;
$serial = serialize($float);
var_dump($serial);
var_dump(unserialize($serial));
```



```
string 'd:3.1415926535900001;'
(length=21)
float 3.14159265359
```

La sérialisation

```
<?php
$chaine = 'il dit : "j\'ai faim !"';
$serial = serialize($chaine);
var_dump($serial);
var_dump(unserialize($serial));
```



```
string 's:22:"il dit : "j'ai faim !"' (length=30)
string 'il dit : "j'ai faim !"' (length=22)
```

La sérialisation


```
<?php
$array = array(0, 4, 7, 3);
$serial = serialize($array);
var_dump($serial);
var_dump(unserialize($serial));
```



```
string 'a:4:{i:0;i:0;i:1;i:4;i:2;i:7;i:3;i:3;}' (length=38)
array (size=4)
    0 => int 0
    1 => int 4
    2 => int 7
    3 => int 3
```


La sérialisation

```
<?php
$array = array('truc' => 'machin', 'chose' => 8, 9 => 'yeah !');
$serial = serialize($array);
var_dump($serial);
var_dump(unserialize($serial));
```



```
string 'a:3:{s:4:"truc";s:6:"machin";s:5:"chose";i:8;i:9;s:6:"yeah !";}' (length=63)
array (size=3)
    'truc' => string 'machin' (length=6)
    'chose' => int 8
    9 => string 'yeah !' (length=6)
```

La Programmation Orientée Objet en PHP

La sérialisation

```
<?php
class ClasseSimple {
    private $entier;
    private $chaine;

    public function __construct($a, $b) {
        $this->entier = $a;
        $this->chaine = $b;
    }

    public function __toString() {
        return 'Je suis une instance de la classe "ClasseSimple" et j\'ai comme attributs "$entier" qui vaut "' .
            $this->entier . '" et "$chaine" qui vaut "' . $this->chaine . '"<br>';
    }
}
```

```
<?php
$instance1 = new ClasseSimple(17, 'bonjour');
echo $instance1;
$serial = serialize($instance1);
var_dump($serial);
var_dump(unserialize($serial));
echo unserialize($serial);
```



Je suis une instance de la classe "ClasseSimple" et j'ai comme attributs "\$entier" qui vaut "17" et "\$chaine" qui vaut "bonjour"

string

'O:12:"ClasseSimple":2:{s:20:"?ClasseSimple?entier";i:17;s:20:"?ClasseSimple?chaine";s:7:"bonjour";}' (length=99)

object (ClasseSimple) [2]

private 'entier' => int 17

private 'chaine' => string 'bonjour' (length=7)

Je suis une instance de la classe "ClasseSimple" et j'ai comme attributs "\$entier" qui vaut "17" et "\$chaine" qui vaut "bonjour"

La sérialisation

```
<?php
class ClasseComplexe {

    private $tab;
    private $inst;

    function __construct() {
        $this->tab =
            ['test' => [17, 24, 56, 'bonjour'], 'autre' => ['tutu' => 45, 23, 42]];
        $this->inst = new ClasseSimple(17, 'bonjour');
    }
}

$instance2 = new ClasseComplexe();
$serial = serialize($instance2);
var_dump($serial);
var_dump(unserialize($serial));
```

La sérialisation

```
string 'O:14:"ClasseComplexe":2:
{s:19:"ClasseComplexe"tab";a:2:{s:4:"test";a:4:{i:0;i:17;i:1;i:24;i:2;i:56;i:3;s:7:"bonjour";}
s:5:"autre";a:3:{s:4:"tutu";i:45;i:0;i:23;i:1;i:42;}}
s:20:"ClasseComplexeinst";O:12:"ClasseSimple":2:{s:20:"ClasseSimpleentier";i:17;
s:20:"ClasseSimplechaine";s:7:"bonjour";}}' (length=300)
object (ClasseComplexe) [3]
    private 'tab' => array (size=2)
        'test' => array (size=4)
            0 => int 17
            1 => int 24
            2 => int 56
            3 => string 'bonjour' (length=7)
        'autre' => array (size=3)
            'tutu' => int 45
            0 => int 23
            1 => int 42
    private 'inst' => object (ClasseSimple) [4]
        private 'entier' => int 17
        private 'chaine' => string 'bonjour' (length=7)
```



La Programmation Orientée Objet en PHP

La sérialisation

```
<?php
class ClasseAvecCallback {
    private $att1;
    private $att2;
    private $att;

    function __construct() {
        $this->att1 = 'a';
        $this->att2 = mt_rand(1, 24);
        $this->att = "construction";
    }

    function __sleep() {
        $this->att2 += 5;
        $this->att2 *= 25;
        return array('att1', 'att2');
    }

    function __wakeup() {
        $this->att = "delinéarisation";
        $this->att2 /= 25;
        $this->att2 -= 5;
    }
}
```

```
<?php
$instance2 = new ClasseAvecCallback();
var_dump($instance2);
$serial = serialize($instance2);
var_dump($serial);
var_dump(unserialize($serial));
```

La sérialisation



```
object(ClasseAvecCallback) [1]
    private 'att1' => string 'a' (length=1)
    private 'att2' => int 14
    private 'att' => string 'construction' (length=12)
string 'O:18:"ClasseAvecCallback":2:{s:24:"?ClasseAvecCallback?att1";s:1:"a";
s:24:"?ClasseAvecCallback?att2";i:475;}' (length=108)
object(ClasseAvecCallback) [2]
    private 'att1' => string 'a' (length=1)
    private 'att2' => int 14
    private 'att' => string 'delinéarisation' (length=16)
```

Le chargement automatique des classes

```
<?php
// fonction qui réalise l'inclusion du fichier correspondant
// au nom de la classe
function chargementAutomatique($nomClasse) {
    include_once '../classes/' . $nomClasse . '.class.php';
}

// Enregistrer la fonction comme fonction d'autochargement.
// PHP l'appellera si on fait appel à une classe qui n'est pas
// incluse.
spl_autoload_register('chargementAutomatique');
```

Le chargement automatique des classes

- Dans le dossier classes

```
<?php
class Chaîne {

    private $x;

    public function getX() {
        return $this->x;
    }

    public function setX($x) {
        $this->x = $x;
    }

    function __construct($x) {
        $this->x = $x;
    }
}
```

```
<?php
require_once '../autoload.php';
// Appel à la classe Chaîne
//-> la fonction chargementAuto réalise l'include de
../classes/Chaîne.class.php au moment de l'instanciation.
$b = new Chaîne('Je suis une chaîne de caractères !');
echo 'instance Chaîne créée : ' . $b->getX();
```



instance Chaîne créée : Je suis une chaîne de caractères !

Le clonage d'instance

- Copie d'un type référence
 - Copie de la référence
- Clone d'un type référence
 - Copie de l'élément référencé

La Programmation Orientée Objet en PHP

Le clonage d'instance

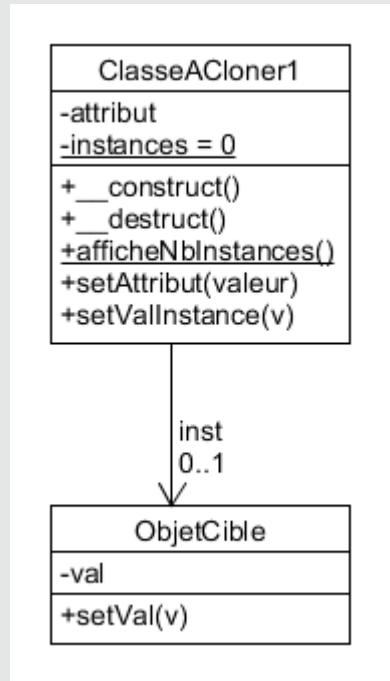
```
<?php
class ClasseACloner1 {
    private $attribut;
    private static $instances = 0;
    private $inst;

    public function __construct() {
        static::$instances++;
        $this->attribut = 'test';
        $this->inst = new ObjetCible();
    }

    public function __destruct() {
        static::$instances--;
        echo 'Suppression d\'une instance de ' . __CLASS__ . '<br>';
    }

    public static function afficheNbInstances() {
        echo 'il y a ' . static::$instances . ' instance' . (static::$instances > 1 ? 's' : '') .
            ' de la classe ' . __CLASS__ . '<br>';
    }

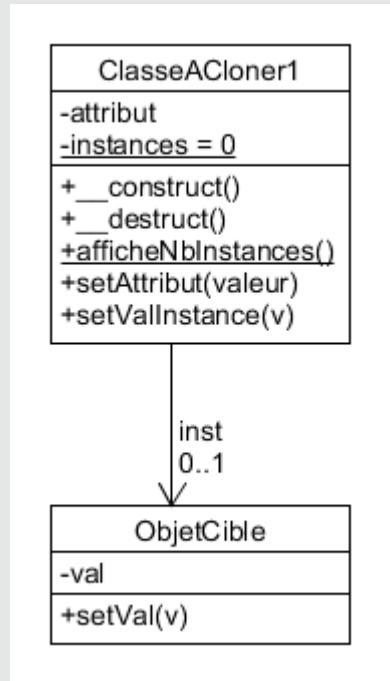
    public function setAttribut($valeur) { $this->attribut = $valeur; }
    public function setValInstance($v) { $this->inst->setVal($v); }
}
```



Le clonage d'instance

```
<?php
class ObjetCible {
    private $val = 12;

    public function setVal($v) {
        $this->val = $v;
    }
}
```



La Programmation Orientée Objet en PHP

Le clonage d'instance

```
<?php
require_once '../classes/ClasseACloner1.class.php';
require_once '../classes/ObjetCible.class.php';

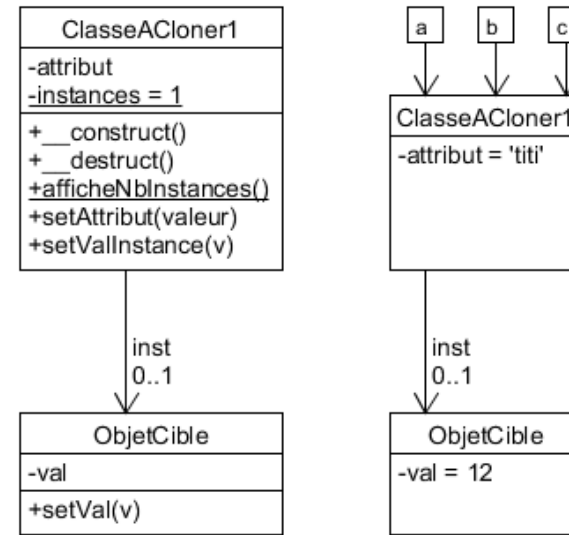
$a = new ClasseACloner1();

$b = $a;
$c = &$a;

$c->setAttribut('titi');

var_dump($a);
var_dump($b);
var_dump($c);

ClasseACloner1::afficheNbInstances();
```



```
object (ClasseACloner1) [1]
    private 'attribut' => string 'titi' (length=4)
    private 'inst' =>
        object (ObjetCible) [2]
            private 'val' => int 12

object (ClasseACloner1) [1]
    private 'attribut' => string 'titi' (length=4)
    private 'inst' =>
        object (ObjetCible) [2]
            private 'val' => int 12

object (ClasseACloner1) [1]
    private 'attribut' => string 'titi' (length=4)
    private 'inst' =>
        object (ObjetCible) [2]
            private 'val' => int 12
```

Il y a 1 instance de la classe ClasseACloner1
Suppression d'une instance de ClasseACloner1

La Programmation Orientée Objet en PHP

Le clonage d'instance

```
<?php
require_once '../classes/ClasseACloner1.class.php';
require_once '../classes/ObjetCible.class.php';

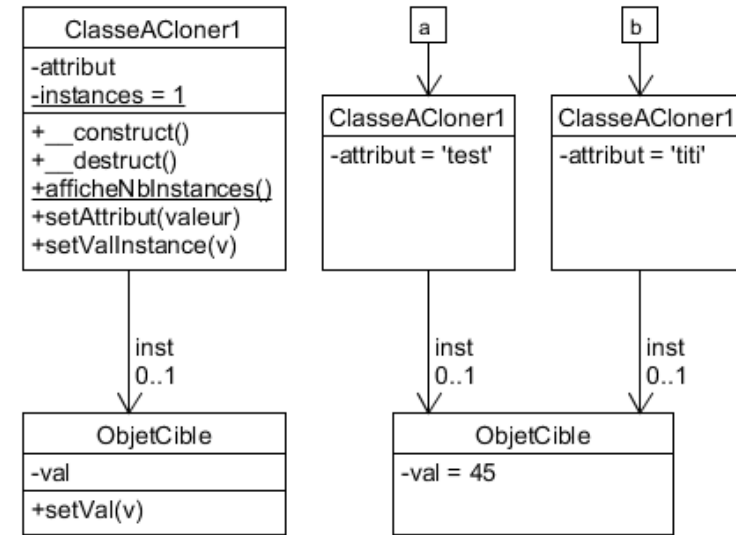
$a = new ClasseACloner1();

$b = clone $a;

$b->setAttribut('titi');
$b->setValInstance(45);

var_dump($a);
var_dump($b);

ClasseACloner1::afficheNbInstances();
```



```
object (ClasseACloner1) [1]
  private 'attribut' => string 'test' (length=4)
  private 'inst' =>
    object (ObjetCible) [2]
      private 'val' => int 45
object (ClasseACloner1) [3]
  private 'attribut' => string 'titi' (length=4)
  private 'inst' =>
    object (ObjetCible) [2]
      private 'val' => int 45
```

il y a 1 instance de la classe ClasseACloner1

Suppression d'une instance de ClasseACloner1

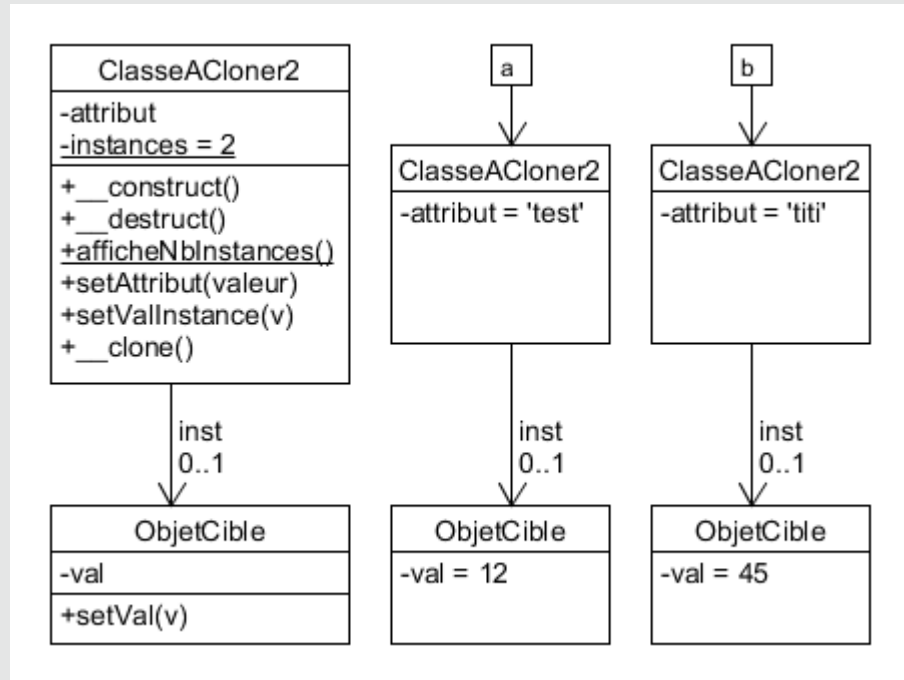
Suppression d'une instance de ClasseACloner1

La Programmation Orientée Objet en PHP

Le clonage d'instance

- Utilisation de la méthode magique `__clone()`

```
public function __clone() {  
    static::$instances++;  
    if ($this->inst != null) {  
        $this->inst = clone $this->inst;  
    }  
}
```



La Programmation Orientée Objet en PHP

Le clonage d'instance

```
<?php
require_once '../classes/ClasseACloner2.class.php';
require_once '../classes/ObjetCible.class.php';

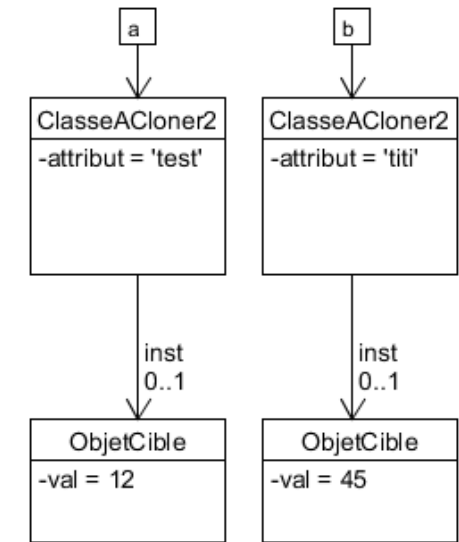
$a = new ClasseACloner2();

$b = clone $a;

$b->setAttribut('titi');
$b->setValInstance(45);

var_dump($a);
var_dump($b);

ClasseACloner2::afficheNbInstances();
```



```
object (ClasseACloner2) [3]
    private 'attribut' => string 'test' (length=4)
    private 'inst' =>
        object (ObjetCible) [2]
            private 'val' => int 12
object (ClasseACloner2) [1]
    private 'attribut' => string 'titi' (length=4)
    private 'inst' =>
        object (ObjetCible) [4]
            private 'val' => int 45
```

Il y a 2 instances de la classe ClasseACloner2