ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR

Membre de HONORIS UNITED UNIVERSITIES

Langage de Script PHP

Chapitre 3

Pr. Zainab OUFQIR

Z.Oufkir@emsi.ma

Programmation orientée objet

- Une classe peut être vue comme une structure de données qui contient:
  - des attributs ou propriétés internes
  - des méthodes : des fonctions définies à l'intérieur de la classe.
- Chaque propriété/méthode se voit attribuer un mode permettant de définir l'acessibilité de la classe:
  - public : accessible à partir de n'importe quel objet de la classe, et même dans les classes et objets dérivés
  - private : accessible uniquement à l'intérieur de la classe. Aucun objet ne peut y accéder
  - protected : accessible dans la classe et dans les classes dérivées

Programmation orientée objet

- Une classe se déclare de la manière suivante :

```php
<?php

class NomDeLaClasse
{
}
```

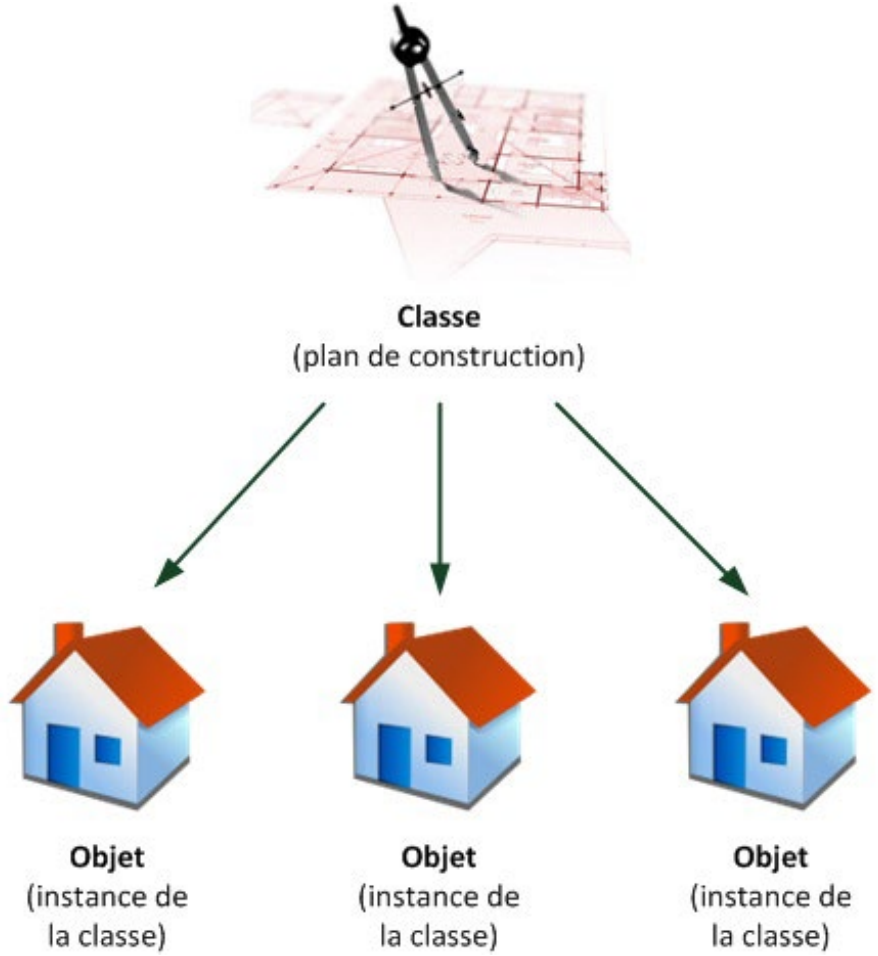
- Le mot clé class suivi d'un nom en PascalCase, un retour à la ligne, suivi d'une paire d'accolades.

  Le code de notre classe se trouvera entre les accolades.

Programmation orientée objet

- Pour obtenir un objet, il faut demander au langage de le créer et de nous le donner pour qu'on puisse le manipuler, c'est la qu'intervient une classe.
- La classe est un plan, une description de l'objet. Imaginez qu'il s'agit par exemple des plans de construction d'une maison.
- L'objet est une instance de la classe, c'est-à-dire une application concrète du plan. Pour reprendre l'exemple précédent, l'objet est la maison. On peut créer plusieurs maisons basées sur un plan de construction. On peut donc créer plusieurs objets à partir d'une classe.

Programmation orientée objet



Programmation orientée objet

- Ajoutons une propriété définissant la longueur de notre pont :

```php
<?php

declare(strict_types=1);

class Pont
{
    public float $longueur = 0;
}

$pont = new Pont;
$pont->longueur = 263.0;

var_dump($pont);
```

Programmation orientée objet

- En ligne 3 se trouve une instruction demandant à PHP d'être exigeant avec le typage. Nous utilisons pour cela le mot clé declare.
- En ligne 7 se trouve la déclaration de la propriété. Elle se compose :
  - du mot clé public, afin de définir l'accessibilité de la propriété;
  - du type de la propriété, ici float (ce n'est pas obligatoire mais c'est une très bonne pratique) ;
  - et enfin, du nom de la propriété préfixé du symbole $.
- En ligne 11 nous assignons une valeur pour notre instance $pont.

Programmation orientée objet

- Une classe peut définir des fonctions qu'elle seule sera en capacité d'exécuter:

```php
<?php

declare(strict_types=1);

class Pont
{
    public float $longueur;
    public float $largeur;

    public function getSurface(): float
    {
        return $this->longueur * $this->largeur;
    }
}
```

Programmation orientée objet

- Appelons maintenant la méthode getSurface() pour obtenir la surface :
  - le nom de la variable contenant l'instance,
  - suivi d'une flèche -> ainsi que le nom de la méthode,
  - suffixé d'une paire de parenthèses vide.

```php
// …

$pont = new Pont;
$pont->longueur = 286.0;
$pont->largeur = 15.0;

$surface = $pont->getSurface();

var_dump($surface);
```

Programmation orientée objet

- L'usage de $this nous permet de faire références aux valeurs portées par chacune des instances.
- Dans notre exemple, peu importe le pont créé, getSurface() doit renvoyer la surface du pont en question, à partir de sa propre largeur et de sa propre longueur. Nous n'aurons pas les mêmes tailles pour tous les ponts. Pour cela, on utilise le mot clé $this

- Exemple:

```php
class Pont
{
    public float $longueur;
    public float $largeur;

    public function getSurface(): float
    {
        return $this->longueur * $this->largeur;
    }
}

$towerBridge = new Pont;
$towerBridge->longueur = 286.0;
$towerBridge->largeur = 15.0;

$manhattanBridge = new Pont;
$manhattanBridge->longueur = 2089.0;
$manhattanBridge->largeur = 36.6;

$towerBridgeSurface = $towerBridge->getSurface();
$manhattanBridgeSurface = $manhattanBridge->getSurface();

var_dump($towerBridgeSurface);
var_dump($manhattanBridgeSurface);
```

Programmation orientée objet

- Pour utiliser une méthode sans instance, elle doit être déclarée statique . Sa valeur sera partagée pour toutes les instances.

```php
class Pont
{
    public static function validerTaille(float $taille): bool
    {
        if ($taille < 50.0) {
            trigger_error(
                'La longueur est trop courte. (min 50m)',
                E_USER_ERROR
            );
        }

        return true;
    }
}

var_dump(Pont::validerTaille(150.0));
var_dump(Pont::validerTaille(20.0));
```

Programmation orientée objet

- Pourquoi on n'a pas utilisé une flèche pour accéder à validerTaille() ?
- Pont fait référence à la classe or, rappelez-vous, -> permet d'accéder aux éléments d'un objet, c'est-à-dire d'une instance.
- Pour dire à PHP que nous souhaitons faire référence à un élément de la classe, il faut utiliser :: à la place, comme pour les constantes.

Programmation orientée objet

- On appelle la méthode __construct un constructeur. Il est appelée automatiquement par PHP lorsque nous créons une instance à l'aide du mot clé new.

- Le constructeur nous sert à initialiser des données de départ pour notre objet.

```php
<?php

declare(strict_types=1);

class Pont
{
    private float $longueur;
    private float $largeur;

    public function __construct(float $longueur, float $largeur)
    {
        $this->longueur = $longueur;
        $this->largeur = $largeur;
    }
}

$towerBridge = new Pont(286.0, 62.0);
```

Programmation orientée objet

- L'héritage permet à des classes (mères) de transférer des propriétés et méthodes à d'autres classes (filles).
- L'héritage se fait à l'aide du mot-clé extends. Il faut que la classe mère soit d'abord définie. Le mot-clé extends peut se lire par "est un" : un administrateur est un utilisateur

Programmation orientée objet

- Classe mère:

```php
<?php

declare(strict_types=1);

class User
{
    private string $status  ;

    public function setStatus(string $status): void
    {
        $this->status = $status;
    }

    public function getStatus(): string
    {
        return $this->status;
    }
}
```

Programmation orientée objet

- Classe fille:

```php
class Admin extends User
{
    private array $roles  ;
    // Méthode d'ajout d'un rôle, puis on supprime les doublons avec array_filter.
    public function addRole(string $role): void
    {
        $this->roles[] = $role;
    }

    // Méthode de renvoie des rôles, dans lequel on définit le rôle ADMIN par défaut.
    public function getRoles(): array
    {
        $roles = $this->roles;

        return $roles;
    }

    public function setRoles(array $roles): void
    {
        $this->roles = $roles;
    }
}
```