PHP Orientée Objet

Qu’est ce que la Programmation Orientée Objet ?

La programmation Orientée Objet est une façon de concevoir son code.

Elle consiste en la création d’une partie appelée Objets. Un Objet représente un concept, une entité voire une idée.

Exemple : une voiture, une personne, ou une page d’un document national.

Objectif

L' objectif de l'orienté objet est de mettre les valeurs de certaines variable dans des “objets”, ce qui aura pour but de faciliter l'incrémentation, la modification et la suppression d’information dans la BDD.

Un objet contient : des attributs, un constructeur et des méthodes.

Exemples

Nous allons prendre deux exemples : une voiture et une personne.

Pour créer un Objet, nous devons faire ce que l'on appelle une classe : le nom de la classe sera le nom de l’objet

```
class Personne {
    //le code de l'objet
}

class Voiture {
    //le code de l'objet
}
```

Exemples

```php
class Personne {
    //variables
    public $nom;
    public $prenom;
    public $age;
    public $nbre_de_pied;
}

class Voiture {
    //variables
    public $modele;
    public $plaqueimmatriculation;
    public $couleur;
}
```

Après avoir créé les Objets, nous allons leur donner des variables qui permettront d'identifier cet objet.

Exemples

Ensuite, il faudrait pouvoir construire/créer cet objet. Il faut donc créer un constructeur qui aura pour mission de créer l'objet que l'on souhaite.

```php
class Personne {
    //variables
    public $nom;
    public $prenom;
    public $age;
    public $nbre_de_pied;

    //constructeur
    public function __construct($N,$P,$A){
        $this->nom=$N;
        $this->prenom=$P;
        $this->age=$A;
        $this->nbre_de_pied=2;

    }
}
```

```php
class Voiture {
    //variables
    public $modele;
    public $plaqueimmatriculation;
    public $couleur;


    //constructeur
    public function __construct($M,$P,$C){
        $this->modele=$M;
        $this->plaqueimmatriculation=$P;
        $this->couleur=$A;
    }
}
```

Méthodes

Enfin, il nous faut implémenter les méthodes. Les méthodes sont des fonctions qui ont pour but d'appeler, modifier et supprimer des variables de la classe.

Deux méthodes très souvent utilisées dans les classes sont les getter et les setter.

Getter

```php
class Personne {
    //variables
    public $nom;
    public $prenom;
    public $age;
    public $nbre_de_pied;

    //constructeur
    public function __construct($N,$P,$A){
        $this->nom=$N;
        $this->prenom=$P;
        $this->age=$A;
        $this->nbre_de_pied=2;

    }
    public function getNom() {
        return $this->nom;
    }
}
```

Les getter permette d'appeler une ou des variables de la classe correspondantes

Getter pour la class Voiture

```php
class Voiture {
    //variables
    public $modele;
    public $plaqueimmatriculation;
    public $couleur;


    //constructeur
    public function __construct($M,$P,$C){
        $this->modele=$M;
        $this->plaqueimmatriculation=$P;
        $this->couleur=$A;
    }
    public function getModele() {
        return $this->model;
    }
}
```

Voici ce que cela donnerait pour le getter de la class Voiture sur le modèle

Setter

Les setter permettent de modifier une variables de la classe avec une autre valeur (cette nouvelle valeur est mise en paramètre de la fonction)

```php
class Personne {
    //variables
    public $nom;
    public $prenom;
    public $age;
    public $nbre_de_pied;

    //constructeur
    public function __construct($N,$P,$A){
        $this->nom=$N;
        $this->prenom=$P;
        $this->age=$A;
        $this->nbre_de_pied=2;

    }
    public function getNom() {
        return $this->nom;
    }
    public function setNom($NewNom) {
        $this->nom = $NewNom;
    }
}
```

Setter pour la class Voiture

Voici ce que cela donnerait pour le setter de la class Voiture sur le modèle

```php
class Voiture {
    //variables
    public $modele;
    public $plaqueimmatriculation;
    public $couleur;

    //constructeur
    public function __construct($M,$P,$C){
        $this->modele=$M;
        $this->plaqueimmatriculation=$P;
        $this->couleur=$A;
    }
    public function getModele() {
        return $this->model;
    }
    public function setModele($NewModele) {
        $this->modele = $NewModele;
    }
}
```

Classe

Voilà, nos classes sont terminées. Il nous reste juste à les mettres en action, pour cela on crée une variable qui va stocker les valeurs de l'objet que l'on souhaite créer, puis on utilise les méthodes :

```php
$Personne = new Personne("Pierre", "Henry", 39);
$Personne->getNom();
$Personne->setNom("AutreNom");
```

L’Héritage

- **Définition :** L'héritage est un concept fondamental de la Programmation Orientée Objet (POO) qui permet à une classe d'hériter des propriétés et des méthodes d'une autre classe.
- **Objectif :** Favoriser la réutilisabilité du code et permettre une structuration hiérarchique des objets.

```php
<?php
class Animal {
    public function manger() {
        echo "L'animal mange.";
    }
}

class Chien extends Animal {
    public function aboyer() {
        echo "Le chien aboie.";
    }
}

// Instanciation et utilisation
$chien = new Chien();
$chien->manger(); // Hérité de la classe parente
$chien->aboie();  // Méthode spécifique à la classe Chien
?>
```

- Points Clés :

1. La classe Chien hérite de la classe Animal.
2. Le chien peut utiliser la méthode manger de la classe parente.
3. La classe Chien peut avoir ses propres méthodes, ici la méthode aboie.

- Avantages de l'héritage

1. Réutilisabilité du Code : Les fonctionnalités communes peuvent être définies dans une classe parente, évitant ainsi la duplication de code dans les classes filles.
2. Modularité : Les classes peuvent être organisées de manière hiérarchique, facilitant la compréhension et la maintenance du code.
3. Évolutivité : Changements dans la classe parente sont automatiquement répercutés dans les classes filles.

- Bonnes Pratiques

1. Utilisation Judicieuse : L'héritage doit être utilisé de manière judicieuse pour éviter une hiérarchie trop complexe.
2. Penser à l'Is-A Relation : La relation entre la classe fille et la classe parente doit respecter le principe "is-a" (par exemple, un Chien est un Animal). $
3. Éviter l'Héritage Multiple : Préférez l'utilisation d'interfaces et de traits pour éviter les problèmes liés à l'héritage multiple.