La POO

INSY2S
Septembre 2022

Qu'est ce que la POO ?

- POO : Programmation Orientée Objet

- La programmation orientée objet est une façon différente d'écrire et d'arranger son code autour de ce qu'on appelle des objets. Un objet est une entité qui va pouvoir contenir un ensemble de fonctions et de variables.

Avantages et inconvénients d'une approche objet

- réutilisabilité du code :
  Le serveur web identifie ce fichier dans son système de gestion de fichiers
  Le fichier est transmis au module d'interprétation PHP du serveur
  Le code HTML est généré par l'interpréteur à partir du code PHP
  Le serveur web répond au client
- Une conception de l'algorithme plus claire et organisée. Le programmeur identifie chaque élément de son programme comme un objet ayant son contexte, ses propriétés et des actions qui lui sont propres.
- code modulaire

PLAN

- Introduction à la « programmation orientée objet »
- Notion de classe
- Notion d'instance
- Propriétés et méthodes

Vous avez dit Objet ?

Définition :

Un « objet » est une représentation d'une chose matérielle ou immatérielle du réel à laquelle on associe des propriétés et des actions.
Par exemple : une voiture, une personne, un animal, un nombre ou bien un compte bancaire peuvent être vus comme des objets.

Attributs

Les « attributs » sont les caractères propres à un objet.
Une personne, par exemple, possèdent différents attributs qui lui sont propres comme le nom, le prénom, la couleur des yeux, le sexe, la couleur des cheveux, la taille...

Méthodes

Les « méthodes » sont les actions applicables à un objet.
Un objet personne, par exemple, dispose des actions suivantes : manger, dormir, boire, marcher, courir...

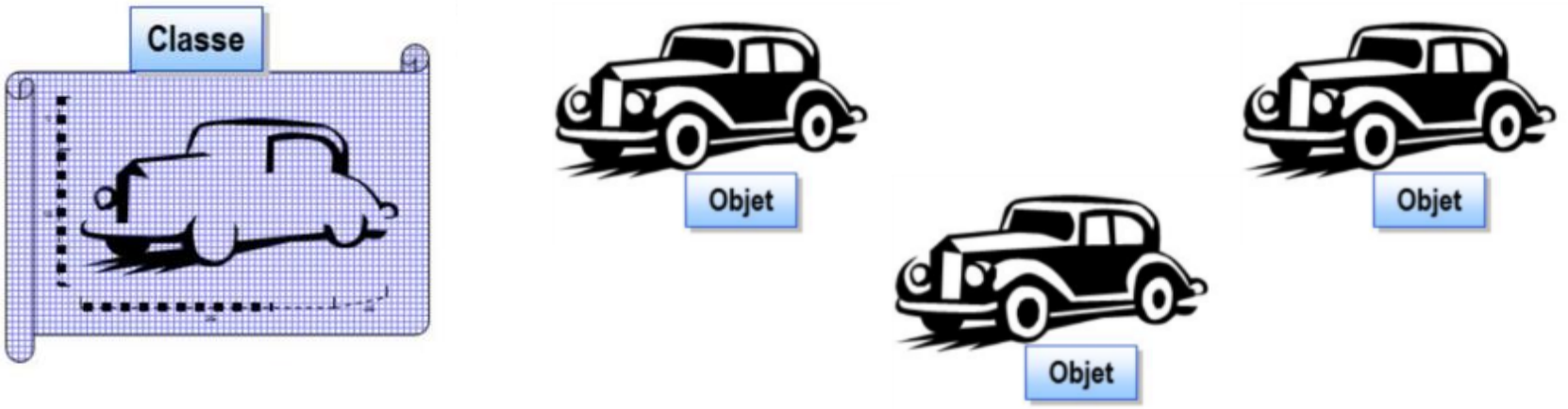
Les classes

- les objets sont bâtis sur des modèles que l'on appelle des classes

percevoir une classe comme un moule grâce auquel nous allons créer autant d'objets de même ***type*** et de même structure qu'on le désire

Instances

- Lorsque l'on crée un objet, on réalise ce que l'on appelle une « instance de la classe ». C'est à dire que du moule, on en extrait un nouvel objet qui dispose de ses attributs et de ses méthodes. L'objet ainsi créé aura pour type le nom de la classe.



Déclaration d'une classe

```php
<?php
class MaClasse
{
// Attributs

// Constantes

// Méthodes
}
?>
```

Remarque :

par convention, on écrit le nom d'une classe en majuscule et en CamelCase»

Exemple de Classe

```php
<?php
class Compte
{
        // Attributs
        public $solde=0;
        public $titulaire;

        // Méthodes
        public function __construct() { }

        public function debiter()
        {
        }
        public function alimenter()
        {
        }
}
```

Attributs

- les attributs sont les caractéristiques propres d'un objet.
- Toute personne possède un nom, un prenom, une date de naissance, un sexe... Tous ces éléments caractérisent un être humain.
- Il existe trois niveaux de visibilité (public, private et protected) qui peuvent être appliqués à un attribut.
- Le mot-clé public permet de rendre l'attribut accessible depuis l'extérieur de la classe.
- En POO, un attribut est qu'une variable, une fonction est une methode
- Deux classes différentes peuvent avoir les même attibuts et methode sans risque de conflit.
- **une constante doit être déclarée et initialisée avec sa valeur en même temps**

Contrôle d'accès

- public : une méthode ou attribut publique est accessible depuis toute l'application.
- private : une méthode ou attribut privé n'est accessible que depuis l'intérieur de la classe.
- protected : une méthode ou attribut protégé est accessible depuis l'intérieur de la classe, et depuis toutes les classes dérivées.

Le constructeur

- Le constructeur est une méthode particulière appelé méthode magique en PHP __construct
- C'est elle qui est appelée implicitement à la création de l'objet (instanciation).

- Le programmeur est libre de définir des paramètres obligatoires à passer au constructeur ainsi qu'un groupe d'instructions à exécuter à l'instanciation de la classe.

```
    function __construct ($titulaire, $depot) {
        $this->titulaire = $titulaire;
        $this->solde = $depot;
    }
}
$monCompte = new Compte("Dupont",150); // instanciation
```

Les méthodes

- Les méthodes sont les actions que l'on peut déclencher sur un objet.
- Il s'agit en fait de fonctions qui peuvent prendre ou non des paramètres et retourner ou non des valeurs / objets.

- Elles se déclarent de la même manière que des fonctions traditionnelles.

- Au même titre que les attributs, on déclare une méthode avec un niveau de visibilité.
- ex : Le mot-clé public
- indique que l'on pourra appeler la méthode en dehors de la classe, c'est à dire sur l'objet lui même.

- Remarque : deux classes différentes peuvent avoir les mêmes méthodes sans risque de conflit.

Instanciation d'une classe

- L'instanciation d'une classe est la phase de création de l’objet.
- Lorsque l'on instancie une classe, on utilise le mot-clé **new** suivant du nom de la classe.
- Cette instruction appelle la méthode constructeur (__construct()) qui construit l'objet et effectue la réservation en mémoire.

```php
$compte = new Compte();
$compte2 = new Compte();
```

Accéder à un attribut

```
class Compte
{
    public $titulaire = "anonyme";
}
$monCompte = new Compte();
echo $monCompte->titulaire;
// affiche anonyme
```

Il est plus prudent de rendre les variables privées et d'y accéder par les accesseurs

Les accesseurs et mutateurs

```php
<?php
class Compte
{
        // Attributs
        private $solde=0;
        private $titulaire;

        // Méthodes
        public function __construct(float $solde, string
        $titulaire) {
                $this->solde = $solde;
                $this->titulaire = $titulaire;
        }
        public function getSolde()
        {
                return $this->solde;
        }
        public function setSolde($newSolde){
                $this->solde = $newSolde ;
        }
}
```

L’héritage en Poo

- Permet de regrouper des parties communes dans une classe dite « mère ».
- Toute classe dérivant de cette classe « mère » est appelée classe « fille ».
- Une classe « fille » possède les attributs et méthodes de la classe « mère » ainsi que des attributs et des méthodes qui lui sont propres.
- Si la classe hérite d'une classe « mère », l'appel du constructeur de la classe « mère » n'est pas implicite. Il faut alors ajouter la ligne suivante pour appeler le constructeur de la classe « mère » : **parent::__construct()**.

L’héritage en Poo

```php
class Vehicule {

    public $marque = "";

    function avance() {
        //code qui fait avancer le véhicule
    }

    function freine() {
        //code qui fait freiner le vehicule
    }
}

class Voiture extends Vehicule {

    function klaxonne() {
        //code qui fait klaxonner la voiture
    }
    // la classe voiture possède un attribut marque, une méthode freine et une méthode avance par héritage
}
```

L'héritage en Poo

- Les méthodes héritées peuvent être réécrites dans la classe « fille ».

```
class Voiture extends Vehicule
{
    public $marque= "peugeot";

    function klaxonne() {
        //code qui fait klaxonner la voiture
    }

    function avance() {
        //code qui fait avancer la voiture
    }

    function freine() {
        //code qui fait freiner la voiture
     }
}
```

L’héritage en Poo

- Les accès aux attributs et méthodes sont redéfinissables dans les classes « filles » pourvu que la directive soit identique ou plus large.
  - Une méthode protégée peut être redéfinie comme protégée ou publique dans une classe « fille ».
  - Une méthode publique ne peut être que publique dans un classe « fille ».
  - Si une méthode privée est redéfinie dans une classe « fille », PHP considèrera qu'il a deux méthodes de même nom simultanément dans la classe « fille ».
    - Si c'est une méthode de la classe « mère » qui y fait appel, elle accèdera à la méthode privée initiale.
    - Si c'est une méthode de la classe « fille » qui y fait appel, elle accèdera à la nouvelle implémentation.

Les classes et méthodes abstraite

- Une classe abstraite est une classe qui ne va pas pouvoir être instanciée directement, c'est-à-dire qu'on ne va pas pouvoir manipuler directement.

- Une méthode abstraite est une méthode dont seule la signature (c'est-à-dire le nom et les paramètres) va pouvoir être déclarée , on ne va pas pouvoir déclarer d'implémentation (c'est-à-dire le code dans la fonction ou ce que fait la fonction).

Les classes Abstraite

- Non instanciable ;
- Toute classe contenant au moins une méthode abstraite doit être déclarée abstraite ;
- Seule la signature d'une méthode abstraite est déclarée (pas d'implémentation) ;
- Les classes dérivées doivent implémenter toutes les méthodes abstraites ;
- Les classes dérivées ne sont pas obligées d'implémenter les méthodes déjà implémentées dans la classe parent, et peuvent posséder leurs propres méthodes.

Les classes Abstraite

```
abstract class Vehicule {

abstract function avancer();

        function tourner($sens) {
                echo "tourne à ".$sens."<br />";
        }
}
```

Héritage d'une classe abstraite

```
class Voiture extends Vehicule {


    function avancer() {
            echo "go!<br />";
    }


    function klaxonner()  {
            echo "tut tut!<br />";
    }


}
```

Les Interfaces

- Non instanciable.
- Seule les signatures des méthodes d'une interface sont déclarées (pas d'implémentation).
- Toutes les méthodes de l'interface doivent être implémentées.
- Les classes peuvent implémenter plus d'une interface en séparant chaque interface par une virgule.

Les Interfaces

```
interface PeutAvancer {

        public function avancer();

        public function freiner();

}

interface PeutTourner {

        public function tournerGauche();

        public function tournerDroite();

}
```

Implémentation d'une interface

```
class voiture implements PeutAvancer, PeutTourner {

    public function avancer() {
        echo "on avance";
    }

    public function freiner() {
        echo "on freine";
    }

    public function tournerGauche() {
        echo "on tourne à gauche";
    }

    public function tournerDroite() {
        echo "on tourne à droite";
    }

    function klaxonner() {
        echo "tut tut!<br />";
    }
}
```

Abstract VS Interface

- Aucun code n'est présent dans une interface :
  - Une interface est donc une classe abstraite qui ne contiendrait que des méthodes abstraites.
  - Une classe ne peut dériver que d'une classe abstraite mais peut implémenter plusieurs interfaces.

Classe et méthode finale

- Ces classes et méthodes ne pourront jamais être héritées.

```
class Voiture extends Vehicule
{
   final function avancer()
   {
            echo "on avance";
   }
}

final class Voiture extends Vehicule
{
   public function avancer()
   {
            echo "on avance";
   }
}
```

Appartenance d'un objet à une classe

```
class Voiture { ... }

$MaVoiture = new Voiture();

if ($MaVoiture instanceof Voiture) {
        echo "cet objet est une voiture";
} else {
        echo "cet objet n'est pas une voiture";
}
```

Obtenir la classe d'un objet

```
class Voiture { ... }

$MaVoiture = new Voiture();

if ($MaVoiture instanceof Voiture) {
        echo "cet objet est une voiture";
} else {
        echo "cet objet n'est pas une voiture";
}
```

Obtenir la classe parente d'un objet

```php
class Vehicule { ... }

class Voiture extends Vehicule { ... }

$MaVoiture = new Voiture();
echo get_class_parent($MaVoiture);
// affiche "Vehicule"

echo get_class_parent('Voiture');
// affiche "Vehicule"
```

Définition de la méthode __toString

- La méthode __toString est automatique appelée à chaque fois que l'objet est utilisé dans un contexte où PHP attend une chaîne (par exemple dans echo).
  - N'a pas de paramètre ;
  - Doit retourner une chaîne.

```
class Compte
{
    function __toString()
    {
        return number_format($this->solde, 2, ',', ' ') . ' &euro;';
    }
}
$monCompte = new Compte("Dupont", 150);
echo $monCompte;
// affiche "150,00 €"
```

Atelier

- Créer une interface Icompte qui contiendra les méthodes suivantes :
  - getSolde()
  - getTitulaire()
  - debiter(montant)
  - Créditer(montant)

- Créer une class abstraite CompteBancaire qui implémentera l'interface Icompte

- Créer deux classes CompteCourant et CompteEpargne qui hériteront de CompteBancaire.

- Surcharger la méthode débiter afin que :
  - Dans la classe CompteCourant, elle ne dépasse pas le montant du découvert autorisé .
  - Dans la classe CompteEpargne, elle ne doit pas être en dessous d'un plancher

- **Créer les pages webs afin de tester la créations de vos comptes**

Aide

- Pour stocker un objet dans une variable session, il faut le sérialiser, puis le « désérialiser » pour le récupérer.

- $_SESSION['compte'] = serialize($cpt);

- $cpt = unserialize($_SESSION['compte']);