Développer en Orienté Objet avec le langage PHP

L'orienté objet

La programmation orientée objet (POO) est un paradigme informatique consistant à définir et à faire interagir des **objets**.

en réalité il s'agit d’une manière d’organiser, de penser son code dans le but de rendre un programme plus maintenable et plus modulaire.

C'est un concept qui a été introduit avec le langage de programmation Simula dans les années 1960 dans le but de faciliter la programmation de logiciels de simulation.Popularisée à partir de 1980 avec l'arrivée du C++, objective C en 1964 et d’autres langages.

Qu’est ce qu’un Objet ?

Un **objet** est un conteneur symbolique qui englobe des caractéristiques (**propriétés**) et des comportements (**méthodes**) et qui par métaphore, représente quelque chose du monde réel.

En programmation orientée objet, un objet est créé à partir d'un modèle appelé **classe** à partir de laquelle il hérite ses comportements et ses caractéristiques. Ces comportements et caractéristiques sont typiquement basés sur celles propres aux choses qui ont inspiré l'**objet.**

Qu’est ce qu’une Classe ?

Une classe en POO regroupe l’ensemble des membres (**propriétés** et **méthodes**) commun à un groupe d’objets. Une classe peut être vue comme un modèle qui sert à décrire un **objet**.

Exemple :

les gâteaux et leur moule. Le moule, il est unique mais Il peut produire une quantité infinie de gâteaux.

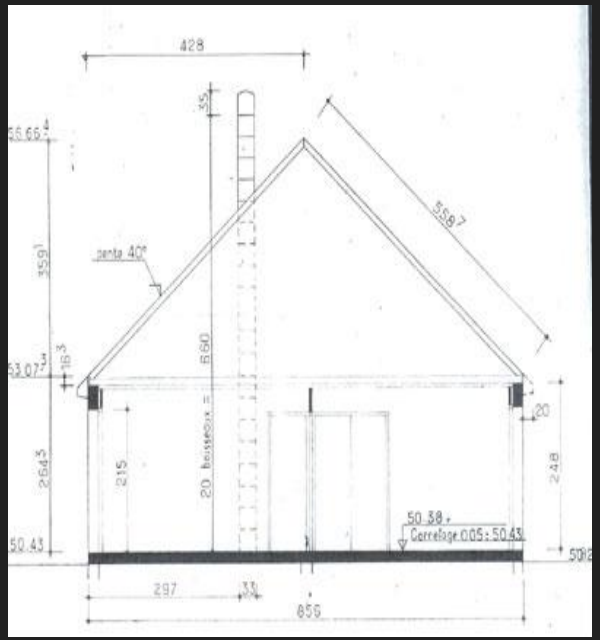
Dans ce cas, les gâteaux sont les objets et le moule est la classe.

Classe / Objet

- La **classe** est un plan, une description de l'objet, ensembles des **attributs** et **méthodes**. Par exemple le plan de construction d’une maison.

- L'**objet** est une application concrète du plan. Dans notre exemple un maçon peut se servir du même plan pour construire autant de maison qu’il souhaite.

Classe / Objet



Définition d'une classe

1. Déclarez la classe à l’aide du mot-clé class suivi du nom que vous souhaitez lui attribuer.
2. Ouvrez un bloc de code à l’aide d’une accolade ouvrante contenant l’intégralité de la définition de la classe.
3. Déclarez les variables propres de la classe comme des variables ordinaires, avec les mêmes règles de nommage.

Définition d'une classe

Par convention les noms de classe commencent par une majuscule et respect le camelCase s'il sont composés.

```
class NomDeClasse{

    // corps de la classe

    // ensemble des attributs et méthodes

}
```

le mot clé class signifie à PHP qu'on est entrain de créer une classe. Entre les accolades se trouvera la définition de la classe.

Définition d'une classe (Ajouter des attributs)

Les **attributs** sont les **propriétés** d’une classe c’est des variables qui appartiennent à la classe.

La déclaration d’un attribut est précédé d’un des mots clés suivant :

- public
- private
- protected

Définition d'une classe (Ajouter des attributs)

```php
class NomDeClasse{
    // déclaration des attribut de la classe
    public $attribut_1;
    public $attribut_2;
    public $attribut_3;
}
```

Définition d'une classe (Ajouter des méthodes)

Une méthode n’est rien d’autre qu’une **fonction** appartenant à la classe.

Comme les attributs les méthodes aussi sont précédés d’un des trois mots clés : public, private ou protected suivi du mot clé **function** et du nom que l’on souhaite lui donner.

Le nom d’une méthode ne doit pas commencer par deux **underscore** (__) cette notation étant réservée à certaines fonctions particulières.

Définition d'une classe (Ajouter des méthodes)

```php
class NomDeClasse{
        // déclaration des attribut de la classe
        public $attribut_1;
        public $attribut_2;
        public attribut_3;
        // déclaration de méthodes de la classe
        public function nomMethode(){
                // instructions...
        }
}
```

La notion de portée

La déclaration d'un attribut et d'une méthode est précéder par un des mots clés : public, private ou protected.

Ces mots clés définissent la portée de la **propriété** ou la **méthode** c'est à dire à partir d'où on peut avoir accès à ces **méthodes** ou **propriétés**.

La portée publique (public)

Le mot clé public quand il précède une déclaration de propriété ou de méthode rend celle-ci accessible à partir de partout dans le programme. Ce qui veut dire que cette méthode ou propriété pourra être invoqué à tout endroit du programme (dans le programme principale comme dans d'autre classes).

Une propriété déclarée avec le mot clé **var** ou une méthode déclarée sans un des trois mots clé aura par défaut une portée public.

La portée privée (private)

L'accès aux éléments privés est uniquement réservé à la classe qui les a définis.

Donc un attribut ou une méthode déclarer avec le mot private ne sera accessible qu'à l’intérieur de la classe qui l’a définie.

c’est une protection de la propriétés au cas il doit respecter certaine règle ce qui permettra de ne pas pouvoir la modifier à partir de n’importe ou dans le programme. Les propriétés déclarées en privée seront manipuler à partir de méthodes définies dans la classe et ayant une portée public.

Instancier un objet

Pour créer un **objet** à partir d'une **classe** on doit utiliser le mot clé new suivi du nom de la classe et des parenthèses. le fait de créer l'objet à partir de la classe s'appelle instancer.

```
$objet1 = new nomClasse();

$objet2 = new nomClasse();

$objet3 = new nomClasse();
```

Accéder aux attributs et méthodes d’un objet

Pour accéder aux **méthodes** et aux **attributs** d’un **objet** on on utilise l’opérateur suivant : **->** une flèche composée du tiret du milieu et du signe supérieur.

exemple:

```
// créer une instance de la classe

$obj = new maClass();

// afficher sa vitesse

echo $obj -> attribut;
```

La pseudo-variable $this

$this est une variable réservée qui désigne l'objet en cours. Elle est utilisée exclusivement à l'intérieur de la classe pour appeler une méthode ou un attribut de l'objet. Elle est accompagner de l'opérateur « **->** ».

```
class maClass{
        public $attribut = 19;
        public function afficherAttribut(){
                echo $this -> attribut;
        }
}
```

Les accesseurs et mutateurs

les accesseurs sont des méthodes qui permettent de récupérer la valeur des attributs.

Les mutateurs sont des méthodes qui permettent de modifier la valeur d'attribut en en effectuant les vérifications nécessaires.

les accesseur sont aussi appeler getter et le mutateur setter.

Les accesseurs

Si un attributs d’une classe est déclarée avec le mot clé « private » cette attribut ne sera pas accessible depuis l'extérieur de la claas qui l’encapsule.

pour rendre cette attribut utilisable depuis l'extérieur de la classe il faut créer une méthode ayant une visibilité public donc un accesseur. par convention les noms des accesseurs commence par get.

```
public function getNomAttribut(){

    // instructions

}
```

Les mutateurs

Si un attributs d’une classe est déclarée avec le mot clé « private » cette attribut ne sera pas accessible depuis l'extérieur de la claas qui l’encapsule.

pour rendre cette attribut modifiable depuis l'extérieur de la classe il faut créer une méthode ayant une visibilité public mutateur. par convention les noms des mutateur commence par set.

```
public function setNomAttribut(){

    //instructions

}
```

Constructeur

Dans ce qui précède, vous avez créé des objets en instanciant la classe action puis avez défini les propriétés des objets ainsi créés. Cette méthode est un peu lourde, car elle implique de définir les propriétés une par une. Il existe une façon plus élégante et plus rapide de créer des objets et de définir leurs propriétés en une seule opération. Elle consiste à créer un constructeur d'objet, qui n'est rien d'autre qu'une fonction spéciale de la classe, dont les paramètres sont les valeurs que vous voulez attribuer aux propriétés de l'objet.

Constructeur

```
class MaClasse{
        private $a;
        private $b;
        public function __construct($a, $b){
                $this -> a = $a;
                $this -> b = $b;
        }
}
```

instanciation d’un objet pour une class avec constructeur

```
$obj1 = new MaClasse(2, 7);

$obj2 = new MaClasse(8, 4);

$obj3 = new MaClasse(1, 0);

$obj4 = new MaClasse(9, 10);
```