GALIMATECH
L'EXPERIENCE DIGITALE

Php Orienté Objet

***

Plan du cours

- Introduction
- Les classes
- Les Méthodes
- Le constructeur
- L'héritage
- Visibilité des propriétés
- Les accesseurs
- Les mutateurs
- Propriété et méthode statiques

La programmation orientée objet consiste à introduire un nouveau type de données qui est l'objet. Ce type de données offre plus de possibilités de contrôle et d'interactions entre les différents composants de notre application.

Php est connu comme un langage procédural. Mais depuis sa version 4 il a introduit la possibilité de programmer en orienté objet. Cette tentative d'immersion dans le monde de la programmation orientée objet a été mieux réussis à partir de la version 5.6.
Nous allons voir ensemble dans ce support, les bases de la POO avec PHP.

Les class

Les classes permettent de définir un nouveau type de données avec des règles et un comportement spécifiques. Pour déclarer une classe en php nous utilisons la syntaxe suivante:

```
class NomClasse
{
    //corps de la classe
}
```

Après la création d'une classe il faut l'instancier.
Instancier consiste à créer une variable qui va avoir comme type de données la classe que nous avons définie. `$nomVariable = new NomClasse();`

Nous pouvons donc dire que la variable `$nomVariable` est un objet ou une instance de la classe `NomClasse`.

Les méthodes

Les méthodes sont des fonctions déclarées dans une classe. Les méthodes d'une classe ne sont accessibles que par les objets de la classe ou elles sont définies et par les objets enfant de cette classe. Pour déclarer une méthode nous allons faire dans notre classe:

```php
public function maMethode(){

    //Instructions...
}
```

Pour utiliser une méthode il faut d'abord créer un objet de la classe dans laquelle elle est déclarée:

```php
$monObjet = new NomClasse();
```

Nous pouvons ensuite utiliser la méthode à travers l'objet `$monObjet->maMethode();`

La méthode __construct(); le constructeur

Le constructeur d'une classe permet de définir les paramètres à renseigner au moment de l'instanciation. En effet, il est possible de déclarer une classe qui pour s'instancier a besoin que l'on définisse la valeur de ses attributs.
Pour un objet personne, par exemple, nous allons avoir besoin des attributs nom et prénom. En définissant la classe Personne nous utilisons alors la méthode __construct pour ajouter ces attributs:

```
class Personne
{
    public $nom;
    public $prenom;

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

Lors de l'instanciation nous devons donc préciser les attributs nom et prénom comme suite

```
$objet_personne = new Personne($nom, $prenom);
```

L'heritage

Les classes qui ont des attributs similaires peuvent être regroupés dans une relation hiérarchique. C'est-à-dire une relation de parent à enfant appelé héritage.

Dans cette relation nous avons une classe parente qui définit les propriétés et les méthodes. Et les classes enfants vont hériter tout ou partie de ses attributs et méthodes. Pour définir un héritage on utilise le mot clé 'extends' lors de la création de la classe.

Reprenons notre exemple de la classe personne. Cette classe peut avoir comme héritier une class professeur.

Cette dernière aura les mêmes attributs que la classe Personne et un attribut matière.

```
class Profs extends Personne
{
    private $matiere;
    public function   construct($matiere){
        $this->matiere = $matiere;
    }
}
```

Les instances de cette classe peuvent alors utiliser les attributs nom et prénom de la classe Personne et les méthodes définies dans cette dernière. Elles peuvent aussi appeler les attributs et méthodes déclarées dans la classe Profs.

La visibilité des propriétés et des méthodes

La visibilité des propriétés et des méthodes définit à quel niveau une propriété ou une méthode est accessible. Il existe trois types de visibilités:

- public: c'est la visibilité par défaut. L'élément avec cette visibilité sera visible dans tout le programme.
- private: les éléments avec cette visibilité ne sont accessibles qu'à l'intérieur de la classe ou ils sont définis.
- protected: les éléments avec cette visibilité ne sont accessibles qu'à l'intérieur de leur propre classe ou à l'intérieur des classes enfants de cette classe.

Il est conseillé de déclarer les attributs et méthodes d'une classe en protected.

Nous pouvons alors modifier la classe Personne comme suite:

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

Les accesseurs

Les accesseurs, aussi appelés getters, permettent d'accéder aux propriétés protected d'une classe. En effet une fois que nous avons défini les attributs de la classe Personnes en protected, la class Profs ne peut plus accéder à ces attributs. Il faut alors définir une méthode accesseur pour accéder aux attributs de la classe parent.
Pour définir un getter il faut créer une méthode dont le nom commence par get suivit du nom l'attribut.

```
//dans la classe Personne

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

Un objet de la class Profs peut accéder à l'attribut nom comme suite:

```
echo $prof->getNom();
```

Les mutateurs

Les mutateurs aussi appelés setters jouent le rôle de définir les valeurs des attributs protected.

Dans notre classe Personne nous pouvons créer un setter pour permettre de définir l'attribut nom d'un objet Profs.

```php
//dans la classe Personne
public function  setNom($value){
   $this->nom = $value;
}
```

```php
$prof->setNom("Famara");
```

Les setters sont essentiels pour sécuriser une classe. Ils permettent de contrôler la manière dont les attributs sont définis. Nous pourrions par exemple nous assurer que la valeur passée au setter est une chaine de caractère.

Propriété et méthode static

Les propriétés et les méthodes définies comme static sont accessibles sans instancier la classe.
Pour déclarer une méthode ou une propriété en statique on utilise le mot-clé static.

```
public static function Bonjour(){
    return "Bonjour vous allez bien";
}
```

Nous pouvons utiliser cette fonction sans instancier la classe Personne.

```
echo Personne::Bonjour();
```

Liens utiles

- https://openclassrooms.com/fr/courses/1665806-programmez-en-oriente-objet-en-php/1665911-introduction-a-la-poo
- https://www.php.net/manual/en/language.oop5.php