PHP

17 Classe et objet :

Dans ce chapitre nous voir php sous un nouvel angle, en utilisant la **programmation orientée objet**.

17.1 Une classe des objets c'est quoi ?

La **programmation orienté objet** nous permet de modéliser dans notre code des éléments de la vie réelle :

Un véhicule, un animal, un bâtiment etc…

Une **classe** fonctionne comme une recette de cuisine (ou un plan de construction) qui va nous permettre de créer des **objets** :

Gardons l'image d'une recette de cuisine c'est le plan (**classe**) qui va nous permettre de réaliser le plats, la recette a besoin d'ingrédients (que nous appellerons **attributs ou propriétés)**, d'étapes (que nous nommerons **méthodes** ou **fonctions**). A la fin de la recette nous allons obtenir un plat (que nous appellerons **objet**). Chaque fois que l'on réalisera la recette nous obtiendrons un nouveau plat qui sera unique et donc un **nouvel objet**.

En programmation orienté objet cela sera la même chose. Nous créerons une **classe** qui sera notre recette, elle contiendra des **attributs** (ou **propriétés**) que l'on peut voir comme nos ingrédients et nous aurons des **méthodes** ou **fonctions** pour effectuer les différentes étapes de réalisation du plat qui sera notre **objet**.

17.2 Créer une classe en PHP :

Pour créer une classe en PHP nous allons créer un nouveau fichier avec la syntaxe suivante :

Ex classe véhicule :

Cette classe va nous permettre de créer des véhicules, elle contiendra des attributs et des méthodes ou fonctions. Le nom d'une classe commence toujours par une majuscule :

```
<?php
        class Vehicule{
        }
?>
```


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

17.3 Instancier un objet :

Pour créer un nouveau véhicule depuis la classe **Vehicule** nous utiliserons la syntaxe suivante :

```
<?php
	//import du fichier class.php qui contient la classe Vehicule
	require './class.php'
	//création d'un nouveau véhicule depuis la classe Vehicule
	$voiture = new Vehicule();
?>
```

Ce code va nous permettre de créer un nouvel **objet** voiture depuis la **classe Vehicule**.

17.4 Ajouter des attributs :

Afin de personnaliser notre classe nous allons créer des attributs (variables), cela va nous permettre d'ajouter des propriétés dans nos objets.

```
< ?php
class Vehicule
{
	//Attributs :
	public $nomVehicule ;
	public $nbrRoue;
	public $vitesse ;
}
?>
```

Dans l'exemple ci-dessus nous avons ajouté des **attributs** à notre véhicule pour définir son nom, son nombre de roues sa vitesse.

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

17.5 Affecter une valeur à un attribut d'un objet :

Pour affecter une valeur à un attribut d'un objet (*la valeur ne sera pas affectée à la classe mais à l'instance de notre objet*) on utilise la syntaxe suivante :

```php
<?php
    //appel du fichier class.php qui contient la classe Vehicule
    //require est équivalent à include
    require './class.php';
    //création d'un nouveau véhicule depuis la classe Vehicule
    $voiture = new Vehicule();
    //ajout de valeur aux attributs de la classe Vehicule
    $voiture->nomVehicule = "Audi A3";
    $voiture->nbrRoue = 4;
    $voiture->vitesse = 250;
?>
```

Le code ci-dessus affecte des valeurs aux **attributs** de **l'objet** voiture (*nomVehicule = Audi, nbrRoue = 4 et vitesse = 250*)

NB : Cette syntaxe est valide uniquement quand les attributs sont en **public**. Nous Verrons plus tard qu'il est conseillé de passer les attributs en **private** ou **protected (pour l'héritage entre classe)**.

17.6 Créer et appeler des méthodes :

Dans nos classes nous avons la possibilité de créer des **fonctions** (*méthodes*) qui serons utilisables par nos objets.

Nous allons créer dans notre classe Vehicule plusieurs méthodes que nos objets pourront utiliser.

17.6.1 Exemple création d'une méthode démarrer :

Dans le fichier **classe Vehicule** (*vehicule.php*) nous allons créer une fonction qui va démarrer le véhicule elle va afficher dans une page html un paragraphe avec comme contenu :

```
"Démarrage de : "nom du véhicule » Vrooom !!!!"
```


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Pour ce faire nous allons utiliser la syntaxe ci-dessous :

```
<?php
class Vehicule{
	/*-----------------------------------------------------
			Attributs :
	-----------------------------------------------------*/
	public $nomVehicule ;
	public $nbrRoue;
	public $vitesse ;
	/*-----------------------------------------------------
			Fonctions :
	-----------------------------------------------------*/
	//fonction démarrer le véhicule
	public function demarrer(){
		echo "<p>Démarrage de la $this->nomVehicule Vrooom !!!!</p>";
	}
}
?>
```

La variable **$this** correspond à **l'instance courante** de notre **objet**.

17.6.2 Appel d'une méthode :

Pour utiliser cette méthode sur un objet nous utiliserons la syntaxe suivante :

```
<?php
    //appel du fichier class.php qui contient la classe Vehicule
    //require est équivalent à include
    require './class.php';
    //création d'un nouveau véhicule depuis la classe Vehicule
    $voiture = new Vehicule();
    //ajout de valeur aux attributs de la classe Vehicule
    $voiture->nomVehicule = "Audi A3";
    $voiture->nbrRoue = 4;
    //utilisation de la méthode démarrer
    $voiture->demarrer();
?>
```

Pour ce faire nous utilisons l'opérateur **->** puis le nom de la fonction suivi de parenthèses.

NB : Pour afficher le détail d'un objet nous utilisons la fonction php ***var_dump($objet)***.

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

PHP

17.7 Constructeur

Pour pouvoir affecter des valeurs à un objet directement au moment de son instanciation nous devons ajouter une méthode à la classe. Celle-ci se nomme constructeur (c'est la méthode qui est appelée au moment du new Classe()).

Par défaut un constructeur vide est disponible dans la classe (même si la méthode n'est pas écrite dans la classe).

Pour créer un constructeur personnalisé (qui va imposer de donner des valeurs à différents attributs) nous utiliserons la syntaxe suivante :

```php
<?php
    class Classe{
        //attributs
        public $nom;
        public $taille;
        //constructeur
        public function __construct(string $nom, ?int $taille){
            $this->nom = $nom;
            $this->taille = $taille;
        }
    }
?>
```

Quand nous instancierons des objets depuis cette classe nous devrons **obligatoirement** renseigner les valeurs définies en paramètre :

```php
//la classe oblige à renseigner une valeur pour l'attribut nom et taille
$classe = new Classe('nom', 20);
```

17.8 Méthode toString :

Si l'on ***echo*** un ***objet*** on se retrouve avec une erreur de type :
`Object of class` ***NomClasse could not be converted to string***. Pour corriger cette erreur nous allons devoir recréer la ***magic method*** **__toString** avec la syntaxe suivante :

```php
public function __toString(){
      return $this->attribut;
      //ici on remplace la valeur par l'attribut de la classe que l'on
      souhaite afficher (par ex le //nom)
}
```

17.9 Exercices :

Exercice 1 :

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

PHP

Créer un fichier **test_objet.php** qui va nous servir de fichier d'exécution,

Créer une nouvelle classe Maison **Maison.php** qui va contenir les attributs suivants :

-nom, longueur, largeur.

Instancier une nouvelle maison dans le fichier **test_objet.php** avec les valeurs de votre choix (**nom**, **longueur** et **largeur**),

-Créer une méthode **surface** qui calcule et affiche la superficie de la maison (**longueur * largeur**) dans la **classe** Maison.

-Appeler la méthode **surface** et **afficher** sous la forme suivante le résultat :

```
"<p>la surface de nomMaison est égale à : x m2</p>".
```

Bonus

Ajouter un attribut **nbrEtage** à la classe Maison,

Modifier la méthode **surface** pour qu'elle prenne en compte le paramètre **nbrEtage**.

Auteur : Mathieu MITHRIDATE
Relu, validé & visé par :
☒ Jérôme CHRETIENNE
☒ Sophie POULAKOS
☒ Mathieu PARIS

Date création : 08 / 12 / 2022
Date révision : xx / xx / 20xx

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

PHP

Exercice 2 :

-Créer un fichier **vehicule.php** qui va contenir la classe,

-Dans ce fichier recréer la classe Vehicule comme dans le cours (**attributs** et **méthodes**),

-Créer un fichier **test_objet.php** au même niveau que vehicule.php,

-Appeler avec **require()** ou **include()** le fichier de la classe **Vehicule**,

-Instancier 2 nouveaux véhicules dans le fichier **test_objet.php** avec les paramètres suivants :

-Objet **voiture** (nomVehicule = « Mercedes CLK », nbrRoue = 4, vitesse 250),

-Objet **moto** (nomVehicule = « Honda CBR », nbrRoue = 2, vitesse = 280),

-Créer une fonction **detect()** qui détecte si le véhicule est une moto ou une voiture (la méthode retourne une **string** ***moto*** ou ***voiture*** avec **return**) dans le fichier de classe **vehicule.php**,

-Exécuter la méthode **detect()** sur les 2 objets voiture et moto dans le fichier **test_objet.php**.

-Afficher le type de véhicule dans le fichier **test_objet.php**,

-Créer une méthode **boost** qui ajoute 50 à la vitesse d'un objet dans le fichier de classe **vehicule.php**,

-Appliquer la méthode **boost** a la voiture dans le fichier **test_objet.php**,

-Afficher la nouvelle vitesse de la voiture dans le fichier **test_objet.php**.

Bonus :

-Créer une méthode **plusRapide()** dans le fichier **vehicule.php** qui compare la vitesse des différents véhicules (*moto* et *voiture*) et retourne le véhicule le plus rapide des 2 avec un **return**.

-Exécuter la méthode **plusRapide()** dans le fichier **test_objet.php**.

-Afficher le véhicule le plus rapide dans le fichier **test_objet.php.**

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.