

## Table des matières

1.	Les classes .....	2
1.1	Exemple de classe.....	3
1.2	Utilisation d'une classe.....	4
1.3	Mot clé « static » .....	4

## 1. Les classes

- Une **classe** représente un objet, c'est-à-dire une représentation en langage de programmation d'un objet
- Une classe permet d'enrichir votre application et ainsi ajouter un nouveau **type** utilisable (en plus des types de bases, int, string etc)
- On peut voir une classe comme un tableau associatif
- Par exemple, une voiture peut se représenter par :
  - Sa couleur, string
  - Sa marque, string
  - Son modèle, string
  - Ses chevaux, int
  - Etc...
- Ces informations sont appelées des **attributs** et décrivent votre objet
- Une classe a besoin d'un constructeur pour l'instancier, c'est-à-dire permettre la création d'objet de ce type.
- Une classe peut aussi voir des **fonctions** (ou **méthodes**), elles représentent souvent un comportement de la classe, de notre objet. Par exemple, on peut imaginer qu'une voiture ai une fonction « rouler() », qui représente le fait qu'elle avance.
- Les **attributs**, et les **fonctions** d'une classe ont ce que l'on appelle un indice de « **visibilité** », c'est-à-dire un mot clé pour indiquer qui à le droit d'accéder à l'attribut ou la méthode.
- Il existe 3 indices de visibilité :
  - Public : accessible depuis partout, aussi bien dans la classe courante que dans les autres
  - Protected : accessible uniquement par la classe courante et ses éventuelles classes filles (**héritage**)
  - Private : accessible uniquement par la classe courante
- Il est recommandé que les attributs d'une classe soient toujours « **private** » ou « **protected** » dans le cas d'une classe mère (héritage). Car c'est le rôle de la classe de modifier ses attributs.
- Ainsi, pour modifier les attributs d'une classe, on utilise ce que l'on appelle des « **getter** » et « **setter** », ils ont là pour permettre de modifier (setter) et récupérer (getter) les attributs de notre objet.
- Au sein d'une classe, on utilise le mot clé « **\$this** » pour accéder aux fonctions ou attributs de celle-ci.
  - Ce mot clé représente « cette classe », « cet objet »

## 1.1 Exemple de classe

```
class Voiture {

    private string $model;

    private string $brand ;

    private string $color ;

    private int $power ;

    /*
     * Constructeur de la voiture
     * @param string $color, la couleur de la voiture
     */
    public function __construct(string $color) {
        // On utilise le mot clé $this pour accéder à l'attribut color de la voiture
        // Attention à ne pas confondre : $color représente le paramètre du
        // constructeur, et $this->color l'attribut de la voiture !
        $this->color = $color ;
    }

    /*
     * Getter de l'attribut couleur
     * @return l'attribut couleur (de type string)
     */
    public getColor() : string {
        // On utilise le mot clé $this pour accéder à l'attribut color de la voiture
        return $this->color ;
    }

    /*
     * Setter de l'attribut couleur
     * @param $color, la couleur à set à l'attribut
     * @return l'attribut couleur (de type string)
     */
    public setColor(string $color) : void {
        // On utilise le mot clé $this pour accéder à l'attribut color de la voiture
        // Attention à ne pas confondre : $color représente le paramètre du
        // constructeur, et $this->color l'attribut de la voiture !
        $this->color = $color ;
    }
}
```

## 1.2 Utilisation d'une classe

- On instancie un objet, c'est-à-dire que l'on crée une variable qui sera du type de notre objet, via le mot clé « **new** »
- Une fois l'objet créé on peut accéder aux attributs et méthode de notre objet, comme on le faisait avec le \$this : \$this->nomFonction

```
// Créer un objet de type voiture que l'on assigne à la variable voiture
$voiture = new Voiture('bleue') ;
// Affiche le retour de la fonction getColor de notre objet
echo $voiture->getColor() ;
// Appelle la fonction setBrand de la classe
$voiture->setBrand('Renault') ;
```

## 1.3 Mot clé « static »

- Suit les mêmes règles de visibilité habituelles
- **Représente un attribut ou méthode commune à toutes les instances de la classe**
- On accède aux attributs static via le mot clé « **self** », en dehors de la classe on y accède via : LeNomDeLaClasse ::NomAttributMethodeStatic
- Exemple :

```
class Voiture {

    public static int $nbRoues = 4 ;

    public static function getNbRoues() : int {
        echo self::$type ;
    }

}
```