Chapitre 4 : notions élémentaires sur le modèle objet

Notion d'objet en PHP

- Un objet est une entité logicielle qui possède :
  - des attributs
  - des méthodes
- Attributs :
  - Son "savoir"
  - L'état de l'objet
  - Comme un ensemble de 'variables'
  - Identique techniquement à un tableau PHP
- Méthode
  - Son "savoir faire"
  - Action sur l'objet pour changer son état
  - Comme un ensemble de 'fonctions' mais sur l'objet

Manipulation des objets

- Les objets appartiennent à une classe
- La classe regroupe le "savoir faire"
  - Les méthodes
  - Attention : en PHP la classe **ne définit pas tous les attributs**
    - Chaque objet d'une même classe peux avoir des attributs différents !
- Objet dans une variable :
  - stocke non pas l'objet, mais un identifiant (i.e. adresse constante) !
- Création d'objets (identique Java et C++):

  $a = new classe();

  Possibilité de passer des valeurs à la création
- Accéder aux attributs et déclencher des méthodes
  - Notation flèche comme en C++ (le point déjà utilisé pour la concaténation)

  $a->attribut = 5;

  $a->run();

Modèle à base d'objets : "personification"

- Voir les objets comme des 'entités' autonomes d'une même famille (leur classe)
- Voir les échanges entre objets comme des demandes entre objets qui s'envoient des **messages**

- Messages simples : "donnes moi ton nom"
  - utilise les attributs
  - des "getters", des "setters"
  - possible simplification en accédant directement à l'attribut
- Messages calculés : "bonjour paragraphe, combien as-tu de mots ?"
  - Nécessite un calcul
- Messages complexes : "bonjour document, combien as-tu de mots ?"
  - Nécessite d'envoyer d'autres messages à d'autres objets, par exemple si le document est composé de paragraphes.

Affectation d'objets

```
class MaClasse {
        public $attribut;
    }
$a = new MaClasse();
$a->attribut = 10;
$b = $a;

var_dump($a);var_dump($b);
```

- Une variable manipulant un objet : en fait uniquement sa **référence**
- ... donc l'affectation copie l'idf de l'objet (pas de nouvel objet !)

```
object(MaClasse)#1 (1) {
  ["attribut"] => int(10)
}
object(MaClasse)#1 (1) {
  ["attribut"] => int(10)
}
```

Passage d'un objet en paramètre

```
function plus (MaClasse $x) {
        $x->attribut++;
}
class MaClasse {
        public $attribut;
}

$a = new MaClasse();
$a->attribut = 10;

plus($a);
```

- Identique à l'affectation d'un variable : copie l'idf de l'objet
- Conséquence : l'objet passé en paramètre est modifiable!
  - Résultat : l'attribut de $a vaut 11
  - NB: identique à Java

Se désigner soi-même

```
class Paragraphe {
        public $contenu;
        function nombreDeMots(): int {
            return str_word_count($this->contenu,0,'àé!');
        }
    }

    $p = new Paragraphe();
    $p->contenu = "Bonjour à tous les étudiants en AS !";
    print($p->nombreDeMots()."\n");
```

- Utiliser la variable contextuelle : $this
- Identique C++

Modifier ses attributs

```
class Voiture {
        public $marque;
        public $modele;
        function __construct(string $ma, string $mo) {
                $this->marque = $ma;
                $this->modele = $mo;
         }
}

$v = new Voiture('Renault','Kangoo');
$v->couleur = 'rouge';
```

- Ajout / suppression d'attributs de manière dynamique

Déclarer ses attributs

```
class Voiture {
        function __construct(string $ma, string $mo) {
                $this->marque = $ma;
                $this->modele = $mo;
        }
}

$v = new Voiture('Renault','Kangoo');
$v->couleur = 'rouge';
```

- Déclaration d'attributs dans la classe est facultative
  - ... mais **conseillée**
- Les attributs déclarés sont toujours présents avec une valeur par défaut (sauf si on les supprime ...)

Parcourir ses attributs

```
$v = new Voiture('Renault','Kangoo');
$v->couleur = 'rouge';
var_dump($v);

foreach ($v as $attrib => $value) {
        echo "$attrib : $value </br>";
}
```

- Parcours les attributs **visibles**

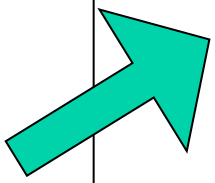
```
object(Voiture)#1 (3) {
 ["marque"]=> string(7) "Renault"
 ["modele"]=> string(6) "Kangoo"
 ["couleur"]=> string(5) "rouge" }
 marque : Renaul t
 modele : Kangoo
 couleur : rouge
```

Accès dynamiques aux attributs

```
$v = new Voiture('Renault','Kangoo');

$attribut = 'couleur'

$v->$attribut = 'rouge';
```



- Un nom d'attribut dans une variable !
- Attention à l'erreur du $ dans le nom d'attribut lors d'un accès

Constructeur avec paramètre tableau

```
class Voiture {
    function __construct(array $attributs) {
      foreach ($attributs as $key => $value) {
        $this->$key = $value;
      }
    }
  };
$v = new Voiture(
        array('marque' => 'Renault','modèle' => 'Kangoo')
        );
    var_dump($v);
```

- Passer en paramètre la liste des attributs et leurs valeurs dans un tableau
- Permet de simuler la surcharge avec nombre de paramètres variable

Constructeur avec paramètre tableau

```
class Voiture {
  function __construct(array $attributs=null) {
    if ($attributs == null) return;
    foreach ($attributs as $key => $value) {
      $this->$key = $value;
    }
  }
};

$w = new Voiture();
```

- Rajouter l'attribut par défaut null pour accepter un constructeur vide.
- NB: en PHP 7.1, il y a un nouveau type de passage de paramètre pour indiquer que le paramètre est possiblement null

  ?type $param : function __construct(?array $attributs)

Modèle objet en PHP : a retenir

- Inspiré du modèle de Java
  - typage des paramètres des méthodes mais pas des attributs
- Objets manipulables automatiquement par référence
  - passage des paramètres par copie de la référence
  - rendu par copie de la référence
  - identique à Java
- Attributs : définit dynamiquement, comme les tableaux associatifs
  - possibilité de parcourir les attributs publiques
  - déclaration dans la classe non nécessaire mais conseillée
- Constructeur facultatif
  - mais pas de surcharge : un seul constructeur
- Méthodes
  - pas de surcharge