Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

Chapitre 3: POO en PHP

ING 3

S. NOUFEL

Plan

- Introduction
- Rappel POO (définitions)
- Classe
- Constructeur
- Attributs et scoop
- Accesseurs et modificateurs
- Classe abstraite
- Interface

Introduction

- Pour des langages de programmation JAVA/C++ … le paradigme de la POO est trivial dès leurs apparition
- L'intégration de la P.O.O en PHP n'était pas une tâche simple (Many years)
- Première tentatives ont été modeste (ratées) dans la version PHP4/PHP5

- **La spécification des scoop public, private, protected, abstract pour les attributs et les méthodes n'est pas respectée**
- **Standardisation du nommage des constructeurs**
- **Interfaces**
- **Clonage des objets**

Introduction

- PHP a intégré les principes de la POO sur plusieurs étapes
- L'approche classique de la programmation en PHP consiste en une séparation complète entre les données et les traitements à exécuter
- La modification des données impose la modification de ces procédures qui les utilises
- Objectif est de facilité la maintenance des applications en se basant sur les avantages du paradigme Orientée objet.

Remarque en PHP combine les deux approches classique et orientées objets

Introduction

On peut faire un programme en PHP qui se compose

- Un ensemble d'objets
- Un objet encapsule les données et les traitements (attributs et méthodes)
- La communication entre les objets se fait par échange de messages

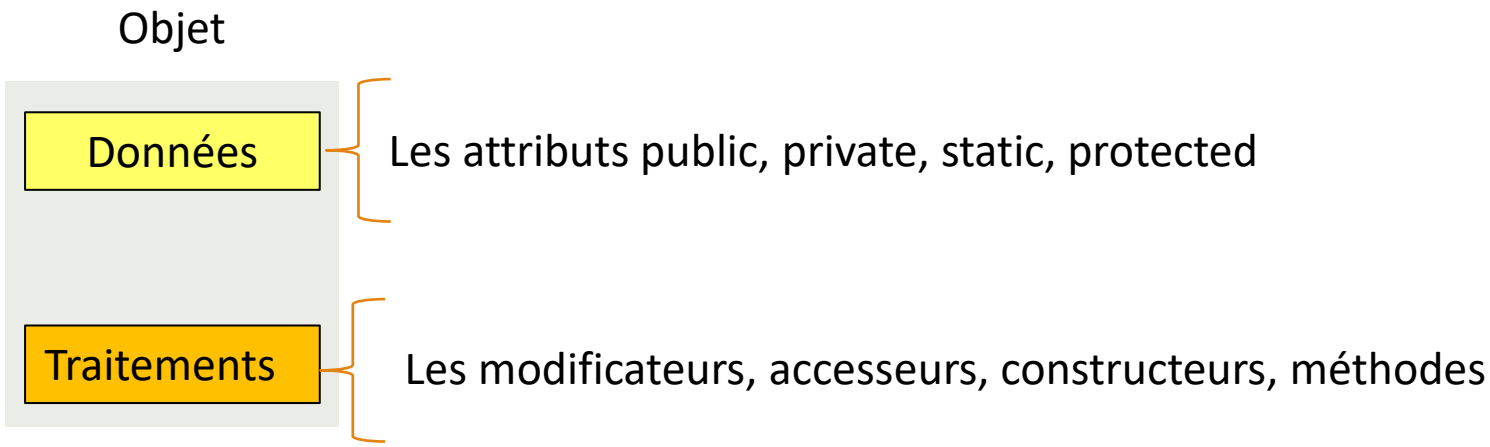
Rappel POO

La POO repose sur trois concepts de base:

- Structuration de l'application en utilisant les classes et les objets
- Communication entre les objets par l'envoie des messages
- Construction de l'application par affinage avec héritage

Objet

En programmation orientée objets, l'objet encapsule les données et les traitements



La création des objets se fait par une classe, l'objet est considéré comme une instance

La classe

- La classe permet de définir le modèle des objets qui ont la même structure et le même comportement
- La classe est le moule qui permet d'instancier un type d'objet
- La création des objets se fait par l'instanciation

Héritage/Généralisation

- Permet de factoriser le code partagé entre plusieurs classes dans la superclasse
- On raffine la spécification de la super classe dans les classes filles
- Les classes filles héritent les attributs et les méthodes de la superclasse
- Les classes filles rajoutent leurs propres attributs
- Les classes filles peuvent définir de nouvelles méthodes et redéfinir les méthodes définit dans la superclasse

Les classes en PHP

- La base de la Programmation Orientée en PHP est la création des classes
- Une classe permet de représenter une entité du monde réel et de spécifier son comportement
- La déclaration d'une classe se fait de la manière suivante

```php
// class produit
class Produit{

}
```

Remarque en PHP on ne spécifie pas le scoop pour la classe (par défaut la classe est considérée publique)

Les classes en PHP

Dans une classe en spécifies la partie statique (données) d'un objet et sa partie dynamique (méthodes)

Une classe permet de déclarer

Les attributs

- Privés, publiques, statique etc.

Les méthodes

- Constructeurs
- Getters et setters
- Méthodes

Attributs d'une classe

La déclaration des attributs d'une classe est similaire à la déclaration des variables, sauf il faut préciser la visibilité des attributs.

```php
<?php
class Produit{

    // déclaration des attributs
    public $designation;
    public $description;
    // déclaration attribut privé
    private $prixUnit;
    private $idProd;
    //déclaration des attributs protégé
    protected $qteStock;

}
```

- **Public** attribut public accessible sans méthode pour toutes les parties du programme
- **Private** attribut privé nécessite des méthodes getter/ setter pour récupérer et modifier sa valeur accessible seulement par la classe qui l'a défini
- **Protected** attribut accessible seulement par la classe qui l'a défini et la méthode et ses classes filles

Les méthodes (par défaut)

- Les méthodes permettent de programmer les traitements et de définir le comportement des objets
- Il existe plusieurs types de méthodes:
  - Constructeur : instancier un objet
  - Getter : récupérer la valeur d'un attribut privé
  - Setter : modifier la valeur d'un attribut privé
  - toString : affichage des attributs d'un objet

Les méthodes (Constructeur)

Le Constructeur (__construct): C'est une méthode qui s'exécute automatiquement au moment précis où vous créez un nouvel objet (avec le mot-clé *new*)

=> Son rôle est de préparer l'objet dès sa naissance. Etape d'initialisation.

La déclaration d'un constructeur

```
 public function __construct($idProd, $prixUnit, $designation,
$description, $qte)   {
        $this->idProd = $idProd;
        $this->prixUnit = $prixUnit;
        $this->designation = $designation;
        $this->description = $description;
        $this->qteStock = $qte;
    }
```

Constructeur

Préciser le type de paramètres php v7

```
public function __construct(string $designation, string $description, float $prixUnit, int $idProd, int $qteStock) {
    $this->designation = $designation;
    $this->description = $description;
    $this->prixUnit = $prixUnit;
    $this->idProd = $idProd;
    $this->qteStock = $qteStock;
}
```

Syntaxe concise php v8

```
public function __construct(
    public string $designation,
    public string $description,
    private float $prixUnit,
    private int $idProd,
    protected int $qteStock
) {}
```

Les méthodes (Constructeur)

- Pour l'instanciation d'un nouvel objet

```
$nomObjet = new Classe(arg1, arg2, arg3, ….);
```

```php
<?php
include 'Produit.php';

$prod1 = new Produit(1, 10, "Clavier", "Clavier azerty", 25);
```

Exercice 1:

Objectif: Créer une classe, l'instancier avec un constructeur, et tester la protection des données.

Enoncé:

1. Créez une classe *Robot.*
2. Ajoutez deux attributs:
   - *$nom*(Public): Le nom du robot.
   - *$codeSecret* (Privé): Son numéro de série unique.
3. Créez le **Constructeur**: Il doit recevoir le nom et le code lors de la création *new* et initialiser les attributs.
4. Test (Main):
   - Créez un robot nommé ‘’Stewie’’ avec le code ‘’1234’’.
   - Afficher: ‘’ Je suis le robot [Nom]’’
   - Essayez d'afficher son code secret directement. => Que se passe t-il dans ce cas?

Les méthodes (get et set)

- Les getters permettent de récupérer les valeurs des attributs déclarés privés
  - Convention de nommage: get + NomDeLAttribut (ex: getNom, getPrix)
- Les setters permettent de modifier les valeurs des attributs déclarés privés
  - Convention de nommage: set + NomDeLAttribut (ex: setNom, setPrix)

```php
public function getPrixUnit()
    {
        return $this->prixUnit;
    }

public function getIdProd()
    {
        return $this->idProd;
    }
```

```php
public function setPrixUnit(): float {
    return $this->prixUnit;
}

public function setIdProd(): int {
    return $this->idProd;
}
```

Les méthodes (accesseurs et modificateurs)

- Accès aux attributs publiques d'un Objet

```
$nomObjet->Attribut // attribut publique
$nomObjet->getAttribut() // en cas d'attribut privé
```

```php
<?php
include 'Produit.php';

$prod1 = new Produit(1, 10, "Clavier", "Clavier azerty", 25);

echo "Id du produit " .  $prod1->getIdProd() . "<br/>";
echo "Prix unitaire " .  $prod1->getPrixUnit() . "<br/>";
echo "Designation " . $prod1->designation  . "<br/>";
echo "Description " . $prod1->description . "<br/>";
echo "Quantité disponible = " . $prod1->getQteStock() . "<br/>";
```

Les Méthodes (toString)

La méthode to String permet de récupérer une chaine de caractère qui représente la concaténation des attributs d'un objet.

```
public function __toString() {
        return "id = $this->idProd prix = $this->prixUnit
                designation = $this->designation description = $this->description
                quantité = $this->qteStock";
    }
```

```
public function __toString(): string {
    return "Produit: {$this->designation}, {$this->description}, Prix: {$this->prixUnit},
                ID: {$this->idProd}, Stock: {$this->qteStock}";
}
```

Héritage

En PHP une classe peut hériter les méthodes et les attributs de la super-classe

Les méthodes héritées peuvent être surchargées => signifie simplement qu'une **classe enfant** (Fille) a le droit de **modifier le comportement** d'une méthode qu'elle a héritée de sa **classe parente** (Mère)

Par exemple

```php
class ProduitLogiciel extends Produit{
    public $version;
    public $taxe;
}
```

```php
class RobotTesla extends Robot{
    public $version;
    public $batterie;
}
```

Héritage

Surcharge des méthodes

Constructeur

```
public function __construct($idProd, $prixUnit, $designation, $description, $qte,
$version, $taxe) {
        parent::__construct($idProd, $prixUnit, $designation, $description, $qte);
        $this->version = $version;
        $this->taxe = $taxe;
    }
```

toString

```
public function __toString() {
        return parent::__toString() . " version = $this->version taxe = $this->taxe";
    }
```

Héritage

Exemple d'utilisation du nouveau constructeur

```
$prod2 = new ProduitLogiciel(2, 500, "MSWord", "MS word 2018", 100, "2018
2.4", 0.2);

echo "produit 2 <br/>";
echo $prod2;
```

Exercice 2:

Objectif: Vous développez le backend d'une application RH. Vous devez gérer différents types d'employés qui ne sont pas payés de la même façon.

Enoncé:

1. La Classe Parente (Employe):
   - *Attribut $nom (chaine) et $salaireFixe (Float)*
   - *Important: Ces attributs doivent être protected*
   - *Constructeur: Initialise le nom et le salaire.*
   - *Méthode calculePaie(): Retourne le salaire fixe.*
   - *Méthode __toString(): Retourne ''Employé: [Nom], Salaire: [Montant] DH''.*
2. La Classe Enfant (Commercial):
   - *Hérite de Employe*
   - *Attribut supplémentaire: $commision (Float) en private.*
   - *Constructeur: Doit recevoir le nom, le salaire fixe ET la commission.*
   - *Surcharge:*
     - Redéfinissez *calculerPaie()* pour qu'elle retourne: *salaireFixe + commision*
     - Redéfinissez *__toString()* pour ajouter la mention ''(Profil Commercial)'' à la fin.
3. Test :
   - Créez un employé classique ''Alice'' (10 000DH).
   - Créez un commercial ''Bob'' (4000 DH fixe + 5000 DH commission).
   - Affichez les deux objets avec *echo*.

Challenge

Soit les classes Etudiant, professeur et chef de département

- Un étudiant est caractérisé par son nom, prenom, Salaire, age, taux
- Un professeur est caractérisé par son nom, prenom, Salaire, age , taux, spécialité
- Un chef de département est caractérisé par son nom, prenom, Salaire, age , taux ,spécialité, et expérience
- Chaque personne paye ces impôts selon le nombre d'heures travaillées selon un taux donné
- Le taux pour les étudiants est de 2,5% et pour les prof est 5% pour les chefs de département 6%.

Challenge

- Donnez l’implémentation de toutes les classes
- Donnez l’implémentation des méthodes de base constructeur, getters, setters, toString
- Donnez l’implémentation des méthode calc_Impot() pour les trois classes
- Créez des objets qui corresponds aux trois classes dans un fichier resultat.php
- Testez le résultat d’appel de la méthode calc_Impot() pour les trois instances crées.

Classe Abstraite

Une classe abstraite est simplement une classe incomplète => Ca sert comme modèle de base pour d'autres classes.
=> Elle contient souvent des méthodes abstraites (des méthodes qui ont un nom, mais pas de code) que les enfants sont obligés de remplir.

```php
abstract class Produit {

    protected string $designation;
    protected string $description;
    protected float $prixUnit;
    protected int $idProd;
    protected int $qteStock;

    // Constructeur
    public function __construct(string $designation, string $description, float $prixUnit, int $idProd, int $qteStock) {
        $this->designation = $designation;
        $this->description = $description;
        $this->prixUnit = $prixUnit;
        $this->idProd = $idProd;
        $this->qteStock = $qteStock;
    }

     abstract public function calculerPrixTTC(): float;
}
```

Classe Abstraite (classe fille)

```php
class ProduitElectronique extends Produit {
    private float $tva = 0.20;

    public function calculerPrixTTC(): float {
        return $this->prixUnit * (1 + $this->tva);
    }
}
```

Interface

Déclarer dans un fichier php, pour imposer l'implémentation des méthodes (contrat)

```
interface Calculable {
    public function calculerPrixTTC(): float;
}
```

Interface

Implémentation du contrat (l'interface)

```
class Produit implements Calculable {
    protected string $designation;
    protected string $description;
    protected float $prixUnit;
    protected int $idProd;
    protected int $qteStock;
    private float $tva = 0.20;

    public function __construct(string $designation, string $description, float $prixUnit, int $idProd, int $qteStock) {
        $this->designation = $designation;
        $this->description = $description;
        $this->prixUnit = $prixUnit;
        $this->idProd = $idProd;
        $this->qteStock = $qteStock;
    }

    public function calculerPrixTTC(): float {
        return $this->prixUnit * (1 + $this->tva);
    }
```