Exercice de Programmation Orientée Objet en PHP : Le Livre

Cet exercice permet de comprendre :

- ❖ Définition des classes et des objets
- ❖ Attribution de propriétés et de méthodes
- ❖ Création d'instances d'objets
- ❖ Appel de méthodes d'objets
- ❖ Construction de programmes plus complexes en utilisant les concepts de la POO.

Explication :

Qu'est-ce qu'un Objet ?

En POO, un objet représente une entité ou un concept du monde réel avec des propriétés (attributs) et des méthodes (fonctions). Imaginez un livre - ses propriétés pourraient être le titre, l'auteur, le nombre de pages et le genre. Les méthodes pourraient être des fonctions pour afficher les informations du livre ou vérifier s'il est disponible pour l'emprunt.

La Classe `Livre` **:**

1. ***Définir la Classe :***

PHP

```
class Livre {
  // Propriétés de l'objet Livre
  public $titre;
  public $auteur;
  public $pages;
  public $genre;

  // Méthodes (fonctions) de l'objet Livre
  public function afficherInfos() {
    echo "Titre : $this->titre<br>";
    echo "Auteur : $this->auteur<br>";
    echo "Pages : $this->pages<br>";
    echo "Genre : $this->genre<br>";
  }
}
```

- **Propriétés:** Nous définissons des propriétés publiques comme `titre`, `auteur`, `pages` et `genre` pour contenir des informations sur le livre.
- **Méthodes:** La méthode `afficherInfos` affiche les détails du livre de manière formatée en utilisant `$this` pour accéder aux propriétés de l'objet.

2. ***Création d'Objets (Instances) :***

PHP

```
$livre1 = new Livre();
$livre1->titre = "Le Guide du Voyageur Galactique";
$livre1->auteur = "Douglas Adams";
$livre1->pages = 184;
$livre1->genre = "Science-Fiction";

$livre2 = new Livre();
$livre2->titre = "Orgueil et Préjugés";
$livre2->auteur = "Jane Austen";
$livre2->pages = 225;
$livre2->genre = "Romance";
```

- Nous utilisons le mot-clé `new` pour créer de nouvelles instances (objets) de la classe `Livre`.
- Ensuite, nous assignons des valeurs aux propriétés de l'objet en utilisant la notation par flèche (->).

3. ***Utilisation des Objets :***

PHP

```
$livre1->afficherInfos();
$livre2->afficherInfos();
```

- Nous appelons la méthode `afficherInfos` sur chaque objet pour afficher ses informations.

4. ***Getters et Setters (Encapsulation) :***

- Nous pouvons contrôler l'accès aux propriétés des objets en utilisant des getters et des setters.
- **Getters:** Méthodes publiques qui renvoient la valeur d'une propriété privée.
- **Setters:** Méthodes publiques qui permettent de définir des valeurs pour des propriétés privées, souvent avec validation.

PHP

```
class Livre {
  private $titre;
```

```
  private $auteur;
  private $pages;
  private $genre;

  public function __construct($titre, $auteur, $pages, $genre) {
    $this->setTitre($titre);
    $this->setAuteur($auteur);
    $this->setPages($pages);
    $this->setGenre($genre);
  }

  public function getTitre() {
    return $this->titre;
  }

  public function setTitre($titre) {
    if (empty($titre)) {
      throw new Exception("Le titre ne peut pas être vide.");
    }
    $this->titre = $titre;
  }
}
```

- On a rendu les propriétés privées (`private`) pour appliquer l'encapsulation et contrôler l'accès.
- Le constructeur (`__construct`) prend des arguments pour l'initialisation et appelle les méthodes setter pour la validation.
- Les méthodes getter et setter fournissent un accès contrôlé aux propriétés.

5. <u>Héritage :</u>

- On peut créer une nouvelle classe qui hérite des propriétés et des méthodes d'une autre classe.
- Cela favorise la réutilisation du code et permet la spécialisation.

PHP

```
class LivreAudio extends Livre {
  private $narrateur;

  public function __construct($titre, $auteur, $pages, $genre, $narrateur) {
    parent::__construct($titre, $auteur, $pages, $genre); // Appeler le
constructeur parent
    $this->narrateur = $narrateur;
  }

  public function getNarrateur() {
    return $this->narrateur;
  }

  public function afficherInfos() {
    parent::afficherInfos(); // Appeler la méthode parente
    echo "Narrateur : $this->narrateur<br>";
  }
}
```

- `LivreAudio` hérite de `Livre`.
- Le constructeur `LivreAudio` appelle le constructeur parent en utilisant `parent::__construct` pour l'initialisation.
- `LivreAudio` ajoute une propriété spécifique `narrateur` et son getter.
- Il redéfinit la méthode `afficherInfos` pour inclure les informations du narrateur, en appelant la méthode du parent avec `parent::afficherInfos`.

Exercice

1. Modifiez la classe `Livre` pour utiliser des getters et des setters pour toutes les propriétés.
2. Implémentez une logique de validation dans les setters pour garantir l'intégrité des données (par exemple, vérifier si le titre est vide ou si le nombre de pages n'est pas numérique).
3. Étendez la classe `LivreAudio` en ajoutant des méthodes pour lire ou mettre en pause le livre audio.
4. Créez une autre classe appelée `LivreNumérique` qui hérite de `Livre` et qui possède des caractéristiques spécifiques comme le format de fichier ou les informations DRM.

Partie 1 : Classe de base `Livre`

PHP

```
class Livre {
  private $titre;
  private $auteur;
  private $pages;
  private $genre;

  public function __construct($titre, $auteur, $pages, $genre) {
    $this->setTitre($titre);
    $this->setAuteur($auteur);
    $this->setPages($pages);
    $this->setGenre($genre);
  }

  public function getTitre() {
    return $this->titre;
  }

  public function setTitre($titre) {
    if (empty($titre)) {
      throw new Exception("Le titre ne peut pas être vide.");
    }
    $this->titre = $titre;
  }

  // Similaires pour auteur, pages et genre

  public function afficherInfos() {
    echo "**Titre:** $this->titre" . PHP_EOL;
    echo "**Auteur:** $this->auteur" . PHP_EOL;
    echo "**Pages:** $this->pages" . PHP_EOL;
    echo "**Genre:** $this->genre" . PHP_EOL;
  }
}
```

Partie 2 : Héritage et classes spécialisées

2.1 Classe `LivreAudio`

PHP

```
class LivreAudio extends Livre {
  private $narrateur;

  public function __construct($titre, $auteur, $pages, $genre, $narrateur) {
    parent::__construct($titre, $auteur, $pages, $genre);
    $this->narrateur = $narrateur;
  }

  public function getNarrateur() {
    return $this->narrateur;
  }

  public function afficherInfos() {
    parent::afficherInfos();
    echo "**Narrateur:** $this->narrateur" . PHP_EOL;
  }
}
```

2.2 Classe `LivreNumérique`

PHP

```
class LivreNumérique extends Livre {
  private $format;
  private $drm;

  public function __construct($titre, $auteur, $pages, $genre, $format, $drm) {
    parent::__construct($titre, $auteur, $pages, $genre);
    $this->format = $format;
    $this->drm = $drm;
  }

  public function getFormat() {
    return $this->format;
  }

  public function getDrm() {
    return $this->drm;
  }

  public function afficherInfos() {
    parent::afficherInfos();
    echo "**Format:** $this->format" . PHP_EOL;
    echo "**DRM:** $this->drm" . PHP_EOL;
  }
}
```

Partie 3 : Exemples d'utilisation

PHP

```
$livre1 = new Livre("Le Guide du Voyageur Galactique", "Douglas Adams", 184,
"Science-Fiction");
$livre1->afficherInfos();

echo PHP_EOL;

$livreAudio1 = new LivreAudio("Orgueil et Préjugés", "Jane Austen", 225,
"Romance", "Kate Winslet");
$livreAudio1->afficherInfos();

echo PHP_EOL;

$livreNumérique1 = new LivreNumérique("Le Petit Prince", "Antoine de Saint-
Exupéry", 96, "Conte", "EPUB", "Non");
$livreNumérique1->afficherInfos();
```

Sortie:

```
**Titre:** Le Guide du Voyageur Galactique
**Auteur:** Douglas Adams
**Pages:** 184
**Genre:** Science-Fiction

**Titre:** Orgueil et Préjugés
**Auteur:** Jane Austen
**Pages:** 225
**Genre:** Romance
**Narrateur:** Kate Winslet

**Titre:** Le Petit Prince
**Auteur:** Antoine de Saint-Exupéry
**Pages:** 96
**Genre:** Conte
**Format:** EPUB
**DRM:** Non
```