Fiche de révision

L’Encapsulation en PHP

1. Qu’est-ce que c’est ?

L'encapsulation est un principe fondamental de la programmation orientée objet (POO) qui consiste à regrouper les données (attributs) et les méthodes (fonctions) qui manipulent ces données au sein d'une même unité structurée, ou classe, tout en restreignant l'accès direct à certains composants internes de cette unité.

En PHP, l'encapsulation est réalisée à l'aide de modificateurs d'accès sur les propriétés et les méthodes des classes. Ces modificateurs déterminent la visibilité et l'accessibilité des éléments de la classe.

Voici les trois principaux modificateurs d'accès utilisés en PHP pour l'encapsulation :

Public : Si un attribut ou une méthode est déclaré(e) comme public, il/elle peut être accédé(e) de n'importe où, ce qui signifie que toute partie du code externe à la classe peut lire ou modifier cet attribut ou appeler cette méthode.

Protected : Les membres **protected** ne sont accessibles que depuis la classe elle-même et ses classes dérivées (enfants). Cela signifie que vous ne pouvez pas accéder à un attribut **protected** ou appeler une méthode **protected** en dehors de la définition de classe ou sans héritage.

Private : Lorsqu'un attribut ou une méthode est défini(e) comme **private**, il/elle ne peut être accédé(e) ou utilisé(e) que par la classe qui les a définis. Même les classes qui héritent de cette classe ne peuvent pas accéder à ses membres privés.

L'encapsulation permet de masquer les détails **d'implémentation** d'un **objet** et de n'exposer qu'une **interface contrôlée**. Cela aide à préserver l'intégrité des données et à réduire les couplages entre les composants du système, ce qui rend le code plus modulaire, plus facile à maintenir et à étendre.

“I've finally learned what - *upward compatible* - means. It means we get to keep all our old mistakes.”

2. Comment faire ?

L'encapsulation est un concept clé de la programmation orientée objet (POO) qui permet de regrouper les données (propriétés) et les méthodes (fonctions) au sein de classes, tout en contrôlant l'accès à ces données. Voici un guide pour comprendre et appliquer l'encapsulation en PHP.

2.1. Définir les Propriétés et Méthodes

Commencez par définir les propriétés et méthodes de votre classe. Par défaut, elles sont public, mais vous pouvez contrôler l'accès en utilisant private ou protected.

```
class CompteBancaire {
    private float $_solde;

    public function __construct(float $soldeInitial) {
        $this->_solde = $soldeInitial;
    }
}
```

2.2. Utiliser des Modificateurs d'Accès

PHP propose trois niveaux de visibilité : public, protected, et private.

public: Accessible de partout.

protected: Accessible dans la classe et les classes dérivées.

private: Accessible uniquement dans la classe où il est défini.

2.3. Fournir des Accesseurs (Getters) et des Mutateurs (Setters) :

Utilisez des méthodes publiques pour obtenir (get) et définir (set) les valeurs des propriétés privées ou protégées.

```
class CompteBancaire {
    private float $_solde;

    public function __construct(float $soldeInitial) {
        $this->_solde = $soldeInitial;
    }
    public function getSolde():float {
        return $this->_solde;
    }

    public function setSolde(float $solde) {
        if ($solde >= 0) {
            $this->_solde = $solde;
        } else {
            echo "Le solde ne peut pas être négatif.";
        }
    }
}
```

2.4. Tester l'Encapsulation

Créez une instance de votre classe et essayez d'accéder ou de modifier ses propriétés directement.

```
$compte = new CompteBancaire(1000);
echo $compte->getSolde(); // 1000
$compte->setSolde(500); // Modifie le solde
$compte->_solde = 50;// ne fonctionne pas !
```

Depuis l'instance $compte, il n'est pas possible d'accéder directement au solde. La méthode setSolde() permet de le modifier. En procédant ainsi, le solde ne peut pas être modifié n'importe comment : des tests (ici le fait que le solde ne peut pas être négatif) sont fait systématiquement.

3. Bonnes Pratiques

L'encapsulation est un pilier fondamental de la programmation orientée objet, permettant de cacher les détails d'implémentation d'une classe et de ne présenter aux utilisateurs de cette classe qu'une interface contrôlée. Voici quelques bonnes pratiques pour l'encapsulation en PHP :

- **Utiliser des Modificateurs d'Accès** : Définissez toujours le niveau d'accès le plus restrictif possible pour les propriétés et les méthodes. Utilisez private par défaut, et ne choisissez protected ou public que si vous avez une raison valable.
- **Accesseurs et Mutateurs** : Utilisez des getters et des setters pour accéder et modifier les propriétés d'une classe. Cela permet non seulement de contrôler l'accès aux données, mais aussi de valider les modifications ou de réaliser des opérations supplémentaires lors de l'accès ou de la modification.
- **Validation dans les Setters** : Utilisez les méthodes set pour valider les données avant de les affecter aux propriétés. Cela garantit que l'objet reste toujours dans un état valide.
- **Minimiser la Visibilité des Méthodes** : Tout comme les propriétés, les méthodes devraient également avoir la visibilité la plus faible possible. Si une méthode n'est utilisée qu'à l'intérieur de la classe, elle doit être private.
- **Encapsulation des Collections** : Si votre classe contient des collections (comme des tableaux ou des objets ArrayObject), ne les retournez pas directement via un getter. Fournissez plutôt des méthodes pour travailler avec la collection sans l'exposer directement (par exemple, addItem, removeItem).
- **Considérer l'Immutabilité** : Dans certains cas, il est bénéfique de rendre les objets immuables, ce qui signifie qu'une fois qu'un objet est créé, son état ne peut plus être modifié. Cela se fait en omettant les setters et en rendant toutes les propriétés private.
- **Pas de Logique Métier dans les Getters/Setters** : Évitez d'ajouter une logique métier complexe dans les getters et les setters. Leur rôle principal est de lire ou d'écrire des valeurs. Toute autre logique doit être placée dans d'autres méthodes dédiées à ces responsabilités.

- **Utiliser le Typage** : Dans les dernières versions de PHP, vous pouvez définir des types pour les propriétés, ce qui ajoute une couche supplémentaire de validation et de clarté sur ce que chaque propriété est censée contenir.

En suivant ces bonnes pratiques, vous vous assurez que votre utilisation de l'encapsulation en PHP est robuste, sécurisée et facilite la maintenance et l'évolution du code.