Fiche de révision

Les Interface en PHP

1. Qu'est-ce que c'est ?

Les interfaces sont des outils puissants qui aident les développeurs à concevoir des codes flexibles et maintenables en utilisant les principes de la programmation orientée objet.

Une interface en PHP est un contrat qui définit un ensemble de méthodes qu'une classe doit implémenter. Une interface n'implémente pas les méthodes qu'elle déclare, elle spécifie simplement quelles méthodes une classe doit implémenter. Cela permet de définir des capacités spécifiques qui peuvent être partagées entre des classes différentes, assurant ainsi un niveau d'abstraction et de modularité.

Il faut voir cela comme un contrat que la classe s'engage à remplir (à implémenter). Cela évite d'oublier d'écrire des méthodes !

Question: "What did the Java code say to the C code?"

Answer: "You've got no class"

2. Comment faire ?

Nous allons définir une interface IUtilisable qui spécifie que tout véhicule doit avoir des méthodes pour démarrer et s'arrêter.

2.1. Définir l'interface

Commencez par définir l'interface IUtilisable. Utilisez le mot-clé `interface` suivi du nom de l'interface. Déclarez ensuite les méthodes que toutes les classes implémentant cette interface doivent définir.

```
interface IUtilisable {
    public function demarrer();
    public function arreter();
}
```

Cette interface IUtilisable exige que toute classe l'implémentant ait une méthode `demarrer` et une méthode `arreter`.

2.2. Implémenter l'interface dans deux classes

Créez une classe Voiture qui implémente l'interface IUtilisable. Vous devez fournir une implémentation concrète pour chaque méthode déclarée dans l'interface.

```
class Voiture implements IUtilisable {
    public function demarrer() {
        echo "La voiture démarre.\n";
    }
    public function arreter() {
        echo "La voiture s'arrête.\n";
    }
}
```

Ici, Voiture est une classe qui implémente IUtilisable et fournit des définitions spécifiques pour demarrer et arreter. On peut également créer une classe FourABois et fournir des définitions spécifiques pour demarrer et arreter.

```
class FourABois implements IUtilisable {
    public function demarrer() {
        echo "je mets du petit bois et j'allume le feu.\n";
    }
    public function arreter() {
        echo "Je renverse de l'eau pour éteindre le feu.\n";
    }
}
```

Il n'y a pas de lien entre les deux classes, simplement que pour me faciliter la vie, je veux avoir à disposition les mêmes noms de méthodes pour démarrer et arrêter. S'il y avait un lien fort (entre une classe Voiture et une classe Camion, il aurait fallu créer une classe-mère VehiculeAutomobile.

2.3. Utiliser les méthodes de l'interface

Créez une instance de la classe Voiture et FourABois et utilisez-les pour appeler les méthodes définies.

```
$maVoiture = new Voiture();
$maVoiture->demarrer(); // Affiche "La voiture démarre."
$maVoiture->arreter(); // Affiche "La voiture s'arrête."
$monFour = new FourABois();
$monFour->demarrer();// Affiche "je mets du petit bois et j'allume le feu.";
$monFour->arreter();// Affiche "Je renverse de l'eau pour éteindre le feu."
```

2.4. Conclusion

En définissant et en implémentant l'interface IUtilisable, nous avons créé un contrat qui garantit que toute classe Voiture (ou toute autre classe) qui implémente cette interface fournira des implémentations spécifiques pour démarrer et arrêter, renforçant ainsi la structure et la prévisibilité du code.

3. Bonnes Pratiques

- **Cohérence** : Assurez-vous que les méthodes définies dans l'interface sont cohérentes et logiquement regroupées.
- **Documentation** : Documentez clairement l'interface et ses méthodes pour expliquer ce qui est attendu des classes qui l'implémentent.
- **Nom de l'interface** : Choisissez un nom descriptif pour l'interface qui reflète clairement son rôle ou les fonctionnalités qu'elle représente.