PHP POO

Exercices

Exercice 1

Organisation du code

1. Faire un nouveau répertoire de travail php-poo et y accéder.
2. Faire un nouveau fichier index.php avec à l'intérieur <?php.
3. Lancer notre serveur PHP avec php -S localhost:8000.
4. Réorganiser le code de la société R en utilisant la POO
    - Définir la classe Employe et ses attributs.
    - Faites en sorte que l'employé puisse se présenter avec une méthode presentation().
    - Ajouter 3 instances d'employés qui se présenteront chacun.

Exercice 2

L’encapsulation

Reprenons notre cas sur la société R.

1. Appliquer l'encapsulation sur la classe Employe.
2. Définir un constructeur pour pouvoir y définir directement les valeurs des attributs.
3. Sécuriser nos méthodes pour que :
    - Nous ne puissions pas avoir plus de 40 ans d'ancienneté.
    - Un employé ait entre 18 ans et 65 ans

Exercice 3

L’encapsulation

Reprenons notre cas sur la société R.

1. Ajouter à la classe Employe :
    - Une constante NB_EMPLOYE_MAX qui aura comme valeur 10, qui correspond à la limite du nombre d'employés.
    - Une propriété statique $nbEmploye qui aura comme valeur 0.
    - Une fonction statique incrementeEmploye qui aura pour fonction d’incrémenter $nbEmploye.
    - Incrémenter le nombre d’employés à chaque création d’un Employe
2. À la fin du code, affichez le pourcentage d'employés par rapport à la limite, via une fonction.

Exercice 4

Héritage

Continuons d'améliorer l'outil de gestion de la société R. En effet, elle souhaiterait gérer aussi le responsable qui est un employé mais avec des privilèges.

1. Ajouter une classe Responsable qui étendra de Employe.
2. Responsable aura comme nouvelle propriété equipe (accesseur/mutateur compris) qui sera un tableau d'employés.
3. Ajouter-lui une nouvelle méthode ajouterEmploye qui acceptera en paramètre un objet de type Employe.

Note : il est possible depuis PHP5 de typer les variables de type classe comme suit :

```
public function test(ClassName $var) { ... }
```

Exercice 5

Interface

Pour avoir plus de contrôle sur le code et les classes dans le projet de la société R, nous allons utiliser les interfaces.

1. Ajouter une interface pour la classe Employe avec ses méthodes.
2. Ajouter une interface pour la classe Responsable avec ses méthodes.
3. Attribuez les interfaces aux classes

Exercice 6

Abstraction

Pour mieux assimiler l'abstraction, nous allons améliorer à nouveau notre société R :

1. Créer une classe Travailleur qui reprendra l'ensemble du code défini dans Employe
2. Étendre Employe et Responsable de Travailleur, supprimer ce qui est nécessaire dans ces classes pour éviter la redondance
3. Définir une méthode abstraite presentation() dans Travailleur obligeant ses enfants à l'implémenter
4. Réimplémenter cette fonction dans les enfants. Testez.

Exercice 7

Polymorphisme

La société R pense à l'éducation des jeunes et souhaite mettre en place la gestion des stagiaires. Ils ne seront pas considérés comme des employés mais pourront tout de même travailler.

1. Ajoutons une nouvelle classe Stagiaire qui n’étendra pas de Travailleur
    - Le stagiaire aura un nom, prénom, âge (vous pouvez utiliser les traits)
2. Ajouter une interface Exploite qui aura une fonction travailler()
3. Implémenter cette interface aux classes Employe et Stagiaire
4. Définir une méthode faireTravailler à la classe Responsable qui pourra faire travailler un objet qui implémente TravailleurInterface. Puis, l'utiliser.
5. Définir une méthode faireTravaillerEquipe à la classe Responsable qui fera travailler toute son équipe. Puis, l'utiliser.

Exercice 8

Namespace

Nous continuons d'améliorer notre projet pour la société R.

1. Ajouter un répertoire Classes/, Traits/, Abstracts/ et Interfaces/ pour stocker nos différentes classes.
2. Réorganiser les classes, traits, abstracts et interfaces dans les dossiers respectifs.
3. Pour chaque fichier, y définir les namespace :
    - namespace Classes;
    - namespace Interfaces
    - namespace Traits
    - namespace Abstracts
4. Faire une require_once de chaque fichier dans index.php
5. Utiliser le mot clé use pour résoudre les erreurs de classe indéfinie dans chaque fichier.

Exercice 9

Exceptions

Nous allons appliquer la gestion d'erreur sur notre application de la société R.

1. Dans la méthode Employe::setAnciennete, gérer les cas de plus de 40 ans
2. Dans la méthode Employe::setAge, gérer les cas des personnes n'entrant pas dans la tranche d'âge de 18 - 65 ans.
3. Dans la méthode Employe::incrementeEmploye, gérer une exception lorsque le quota max est atteint.
4. Utiliser un try-catch lors du set de l’âge, de l'ancienneté ou de la création d’un Travailleur.