

TP PHP 2 - Objets, r flexion, espaces de noms, tests unitaires

D compressez l'archive d pos e sur Moodle pour ce TP. Le dossier r sultant contient diff rents fichiers   r utiliser ou   compl ter. Pensez   consulter le site [PHP](#).

Exercice 1. Le dossier contient l'archive PHAR de l'outil `composer`. Suivez les  tapes 1   6 du **README** pour organiser votre r pertoire, mettre en place espaces de noms et l'autochargeur de `composer`, et pr parer les tests (exercices 6 et 7).

Exercice 2. Impl mentez la classe `Employee` dans le fichier `Employee.php` en dotant chaque instance des propri t s suivantes :

- `id` : entier, acc s priv 
- `name` : cha ne de caract res, acc s public
- `salary` : r el, acc s prot g 
- `age` : entier, acc s priv .

`Employee` doit impl menter l'interface `IEmployee` qui d clare les m thodes suivantes :

- constructeur avec `id`, `name`, `salary` et `age` en param tres
- accesseurs (*getters*) et mutateurs (*setters*)
- m thode magique d'affichage des propri t s de l'objet.

Exercice 3. Cr ez le script `employee_display.php` qui cr e un tableau de trois employ s et les affiche (en mode console ou page web). Le script affiche  galement le salaire moyen de ces trois employ s. Pensez   utiliser les fonctions `array_*`.

Exemple d'affichage :

```
employee: id=1 name=superman salary=1.27 age=80
employee: id=2 name=batman salary=1 age=73
employee: id=3 name=spiderman salary=0.82 age=50
mean salary = 1.03
```

Exercice 4. Cr ez le script `employee_raise.php` qui augmente de 5% le salaire des employ s cr s pr c demment. Le calcul s'effectue avec une fonction `employee_raise` qui v rifie que ce param tre est bien une instance de la classe `Employee` ou l ve une exception dans le cas contraire.

Exemple d'appel correct :

```
Avant augmentation :
employee: id=1 name=superman salary=1.27 age=80
employee: id=2 name=batman salary=1 age=73
employee: id=3 name=spiderman salary=0.82 age=50
Apr s augmentation :
employee: id=1 name=superman salary=1.3335 age=80
employee: id=2 name=batman salary=1.05 age=73
employee: id=3 name=spiderman salary=0.861 age=50
```

Exemple d'appel incorrect :

```
Notice: Array to string conversion in .../employee_raise.php on line 13
Le param tre n'est pas une instance de Employee
```

Exercice 5. Cr ez le script `employee_sort.php` qui trie un tableau d'employ s en ordre de salaire croissant. Exemple :

```

Key-preserving salary-increasing sorting
Array
(
    [spider] => Employee Object
        (
            [id:Employee:private] => 3
            [name] => spiderman
            [salary:protected] => 0.82
            [age:Employee:private] => 50
        )
    [bat] => Employee Object
        (
            [id:Employee:private] => 2
            [name] => batman
            [salary:protected] => 1
            [age:Employee:private] => 73
        )
    [super] => Employee Object
        (
            [id:Employee:private] => 1
            [name] => superman
            [salary:protected] => 1.27
            [age:Employee:private] => 80
        )
)

```

Exercice 6. Le développement piloté par les tests (*test-driven development*) consiste à programmer des tests sur le code source (tests unitaires, tests d'intégration, tests fonctionnels, etc.). Ces tests sont alors exécutés de manière automatique sur chaque version du logiciel cible à l'aide d'outils d'intégration continue (eg. **jenkins**). **PHPUnit** est le framework de référence en PHP pour développer des tests unitaires.

Suivez les étapes 7 et 8 du **README** concernant la mise en place et l'exécution de tests avec **PHPUnit**.

Le fichier **ManagerTest.php** contient un ensemble de tests unitaires destinés à être exécuter avec **PHPUnit** sur une classe **Manager** à implanter. Un manager est un employé qui a sous ses ordres un ensemble d'employés (ses subordonnés). Implémentez d'abord la classe **Manager** (fichier **Manager.php**) qui hérite de **Employee** et implémente l'interface **IManager.php**. Les subordonnés d'un manager pourront être stockés sous forme d'un tableau contenant leurs identifiants.

Afin de tester votre classe :

- (1) exécutez les tests définis dans **ManagerTest.php** avec la commande :
`./vendor/bin/phpunit tests`
- (2) examinez le rapport d'exécution produit par **PHPUnit**
- (3) si tout s'est bien passé, provoquez volontairement un échec du test en introduisant une erreur dans votre classe ou dans l'une des assertions de votre classe de test

Complétez ensuite la classe **ManagerTest.php** (marqueurs **TODO**) :

- en ajoutant un test pour la méthode **setAge** de **Manager**
- en complétant la méthode de test **testAddEmployee**.

Re-testez votre classe avec **PHPUnit**.

Exercice 7. Implémentez une classe **Team** (fichier **Team.php**) qui stocke des employés et des managers. Implémentez la méthode magique d'affichage d'une équipe. Pour une équipe comportant un manager, on affichera le nom des employés qu'il a sous ses ordres. Exemple d'affichage d'équipe :

```

employee: id=1 name=superman salary=1.27 age=80

employee: id=2 name=batman salary=1 age=73

employee: id=3 name=spiderman salary=0.82 age=50

employee: id=4 name=wonder woman salary=3.14 age=71
subordinates=[superman batman spiderman ]

```

Exercice 8. Créez le script **employee_reflex1.php** qui affiche les informations relatives à un employé en utilisant les fonctions de réflexion (**get_object_vars**, ...):

- nom de la classe
- nom de la classe parente
- nom des champs et valeurs.

```

**Classe :
Employee
**Classe parente :
Pas de classe parente
** Propriétés visibles ayant une valeur par défaut :
Array
(
    [name] => anonymous
)
** Propriétés publiques :
Array
(
    [name] => euler
)
** Toutes les propriétés :
Array
(
    [id] => 0
    [name] => euler
    [salary] => 2.718
    [age] => 305
)

```

Exercice 9. Créez le script `employee_reflex2.php` qui affiche les mêmes informations que le script `employee_reflex1.php` mais en utilisant l'API [Reflection](#).

```

Employee
Array
(
    [0] => ReflectionProperty Object
        (
            [name] => id
            [class] => Employee
        )

    [1] => ReflectionProperty Object
        (
            [name] => name
            [class] => Employee
        )
    ...
)
Array
(
    [0] => ReflectionMethod Object
        (
            [name] => __construct
            [class] => Employee
        )

    [1] => ReflectionMethod Object
        (
            [name] => getId
            [class] => Employee
        )
    ...
)

```