Sujet 4 : L'objet simple et les tableaux

Avant de commencer

Dans le répertoire `POO` créez un sous-répertoire `TP4`. Vous y stockerez l'ensemble des solutions développées tout au long de ce TP. Votre enseignant devra y retrouver la trace de tout le travail effectué, ne remplacez donc pas vos solutions par d'autres, privilégiez, en cas de nécessité, une mise en commentaire. Pour une question de lisibilité, veillez à reporter les numéros des questions dans votre code. De la même façon les noms des auteurs devront y apparaître en haut de chaque script. Les réponses aux questions *d'ordre théorique* (qui ne sont pas du code) seront enregistrées dans un fichier TP4.txt.

Création du dépôt Git

Votre travail sera impérativement déposé sur un dépôt Git. Si vous avez opté pour un dépôt commun à tous les TPs, n'oubliez pas de le compléter systématiquement en créant un répertoire par sujet de TP. Dans le cas contraire initialisez un nouveau dépôt en suivant les directives d'initialisation de dépôt décrites ci-dessous.

Consignes

Vous veillerez à respecter scrupuleusement les consignes suivantes :

- Vous effectuerez un commit après chaque exercice.
- A la fin de chaque séance, vous effectuerez un commit. Si ce commit contient du code incomplet ou ne fonctionnant pas, mentionnez-le dans le message de commit. Vous pousserez ensuite votre travail vers le dépôt distant.
- Ce dépôt sera utilisé par votre enseignant(e) de TP pour évaluer votre travail. Assurez-vous donc régulièrement que tous les fichiers que vous souhaitez lui rendre sont bien présents dans le dépôt.
- Le dépôt lui-même sera évalué : soignez l'écriture de vos messages.

Initialisation du dépôt

- Ouvrez un terminal et utilisez la commande *cd* pour vous placer dans le répertoire $TP4$
- Initialisez le dépôt *Git* du projet.
- Si nécessaire, éditez le fichier *.gitignore*
- Effectuez un premier commit
- Connectez-vous à l'application *Gitlab* et créez un dépôt nommé $POO - TP4$
- Assignez le rôle *Reporter* à votre enseignant(e) de TP.
- Poussez le dépôt local vers le dépôt distant.

Liens utiles

- Aide-mémoire Git : *https://iut-info.univ-reims.fr/users/nourrit/git_aide_memoire.pdf*
- Gitlab : *https://iut-info.univ-reims.fr/gitlab*

Les bons usages

D'après les recommandations PSR-2, un script PHP débute par la balise <?*php* mais ne se termine par aucune balise. Vous éviterez de fermer vos scripts avec la balise ? >.

Afin que le typage de vos fonctions et de vos méthodes soit vérifié de manière stricte nous procéderons dorénavant à une typage strict. Vous ajouterez pour cela la directive suivante en tant que première instruction de vos scripts :

```
declare(strict_types = 1);
```

Vos classes doivent être clairement documéntées. Pour générer une documentation associée à votre classe vous le ferez à partir d'un terminal. Vous vous placerez pour ceci au niveau du répertoire contenant les fichiers à documenter. Voici le ligne de commande permettant d'obtenir la documentation de la classe `MaClasse` :

```
phpdoc -t Documentation --visibility public -f MaClasse.php
```

Introduction

L'objectif de ce sujet est d'implémenter une classe simple dont l'un des attribut est un tableau 1D, tout en préservant le principe d'encapsulation des données.

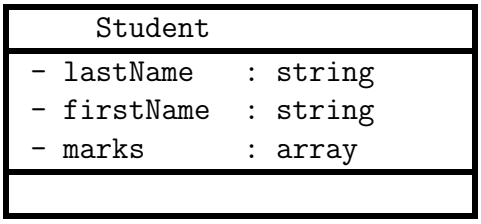
Dans ce sujet, même si ceci n'est pas demandé explicitement, une documentation complète est exigée pour chaque classe et/ou méthode définie.

La classe Student

L'objectif de cet exercice est d'implémenter un classe représentant un étudiant : `Student`.

Les attributs d'instance de la classe

Un `Student` est caractérisé par son nom (`lastName`), son prénom (`firstName`), ainsi que les notes qu'il a obtenues (`marks`). Ci-dessous vous trouverez le diagramme de classe correspondant :



Question 1.
Dans le répertoire *TP4*, créez et éditez un script nommé `Student.php`. Définissez-y les attributs d'instance de cette classe.

Dans le script `TestStudent.php` instanciez un objet $*inconnu* en utilisant l'instruction :

```
$inconnu = new Student;
```

Inspectez le contenu de cette instance en utilisant la fonction `var_dump`. Quels sont les types et valeurs des attributs d'instance de l'objet $*inconnu*. Justifiez votre réponse.

Le(s) constructeur(s)

Question 2.
On souhaite pouvoir instancier un objet de type `Student` comme suit :

```
$louis = new Student("Dupont", "Louis");
```

Ce constructeur initialisera les nom et prénom de l'étudiant à partir des valeurs des paramètres. Il permettra par ailleurs d'initialiser les notes avec un tableau vide. Définissez ce constructeur.

Instanciez un objet de type `Student` avec ce constructeur (l'intanciation précédente devra être mise en commentaire). Inspectez le contenu de cette instance en utilisant la fonction `var_dump`. Quels sont les types et valeurs des attributs d'instance de cet objet.

Question 3.
On souhaite pouvoir instancier un objet de type `Student` comme suit :

```
$notes = array(9.78 , 18 , 12.5 , 10 , 16.25);
$kevin = new Student("Laplace", "Kevin", $notes);
```

Ce constructeur initialisera les nom et prénom de l'étudiant à partir des valeurs des paramètres. Il permettra par ailleurs d'initialiser les notes avec un tableau vide dont les valeurs seront celles du tableau passé en paramètre. Modifiez le constructeur précédent en conséquence.

Instanciez un objet de type `Student` avec ce constructeur. Inspectez le contenu de cette instance en utilisant la fonction `var_dump`. Quels sont les types et valeurs des attributs d'instance de cet objet. Vérifiez que l'instance précédente garde ses propriétés en terme de nombre et de type de ses attributs.

Question 4.
On souhaite pouvoir cloner un objet de type `Student` en faisant appel à la directive *clone*. Complétez la définition de la classe `Student` si nécessaire, sinon justifiez pourquoi.

L'affichage

Question 5.
On souhaite pouvoir afficher une instance de l'objet `Student`. Le rendu désiré pour l'instance `$kevin` est :

```
Laplace Kevin
Notes : [ 9.78  18  12.5  10  16.25 ]
```

On souhaite pour ceci pouvoir procéder de la manière suivante :

```
echo $kevin."\n";
```

Quelle méthode d'instance faut-il alors définir dans la classe `Student` ? Définissez-la sans vous inspirer des méthodes implémentées dans les classes précédentes. On ne s'attardera pas sur le format d'affichage des nombres réels.

Testez la méthode implémentée dans le script `TestStudent.php`.

Question 6.
On souhaite maintenant vérifier que l'implémentation du constructeur prenant en paramètre un tableau respecte le principe d'encapsulation des données. Complétez le script `TestStudent.php` avec les instructions suivantes :

```
echo $kevin."\n";
$notes[0] = 0;
echo $kevin."\n";
```

Si l'implémentation proposée ne respecte pas le principe d'encapsulation des données, la première note de l'étudiant `kevin` aura pour valeur `0`. Expliquez pourquoi. Modifiez la définition de votre constructeur si nécessaire. Testez la nouvelle implémentation.

Les *accesseurs*, *modificateurs* et méthodes assimilées

Question 7.
On souhaite disposer d'un accesseur et d'un modificateur pour l'attribut d'instance `lastName` dans la classe `Student`. Définissez ces deux méthodes dans la classe `Student`. Testez les dans le script `TestStudent.php`.

Question 8.
On souhaiterait disposer d'un accesseur sur l'attribut d'instance `marks` dans la classe `Student`. La solution communément adoptée dans ce cas là est de proposer une méthode d'instance permettant d'accéder à la i^e case du tableau et non au tableau dans sa totalité. Définissez la méthode d'instance `getMark` qui prend en paramètre l'indice de la case dont on veut récupérer la valeur. Si cet indice se trouve en dehors du tableau cette méthode lancera une exception de type `OutOfRangeException`. Pour rappel, la syntaxe d'une fonction qui lève une exception est :

```
public function getMark (int $index) : float // throw OutOfRangeException
{
    if($index < 0 || $index >= count($this->marks))
        throw new OutOfRangeException ( "getMark - indice invalide : $index");
    return [à compléter];
}
```

La documentation de cette méthode devra alors contenir entre autres le mot-clé `@throws`.

Complétez le script `TestStudent.php` en faisant appel à la méthode `getMark`. Pour rappel, la syntaxe d'un bloc qui attrape une exception est :

```
try
{
    echo $kevin->getMark(5)."\n";
    [et tout ce qui en dépend]
}
catch (OutOfRangeException $e)
{
    //gestion de l'exception, par exemple :
    echo $e->getMessage()."\n" ;
}
```

Question 9.
Pour les mêmes raisons, un modificateur de l'attribut d'instance `marks` ne sera pas proposé. A la place, définissez la méthode d'instance `setMark` qui prend en paramètre l'indice de la note que l'on souhaite modifier, ainsi que sa nouvelle valeur. Si cet indice se trouve en dehors du tableau cette méthode lancera une exception de type `OutOfRangeException`.

Complétez le script `TestStudent.php` en faisant appel à la méthode `setMark`.

Autres méthodes

Dans cette partie on veillera à ne pas utiliser les fonctions sur les tableaux existantes en PHP.

Question 10.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `isEqual` qui permet de vérifier si deux étudiants ont obtenu les mêmes notes (l'ordre devra être respecté). Définissez la méthode `isEqual` dans la classe `Student`[1]. Complétez le script `TestStudent.php` afin de valider le bon fonctionnement de la méthode `isEqual`.

Question 11.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getMarksCount` qui retourne le nombre de notes dont dispose un étudiant. Définissez cette méthode et testez-la dans le script `TestStudent.php`.

Question 12.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getMean` qui retourne la moyenne des notes d'un étudiant. Définissez cette méthode et testez-la dans le script `TestStudent.php`.

Question 13.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getMinimum` qui retourne la note minimale obtenue par un étudiant. Définissez cette méthode et testez-la dans le script `TestStudent.php`.

Question 14.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getMaximumIndex` qui retourne l'indice de la note maximale obtenue par un étudiant. Définissez cette méthode et testez-la dans le script `TestStudent.php`. Dans votre définition vous veillerez à ne pas stocker la valeur du maximum dans une variable, seul l'indice sera stocké.

Question 15.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `contains` qui prend en paramètre une note. Elle retourne `true` si la note est présente parmi les notes de l'étudiant, `false` sinon. Définissez cette méthode et testez-la dans le script `TestStudent.php`.

Question 16.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getOccurenceCount` qui prend en paramètre une note. Elle retourne le nombre de fois où la note est présente parmi les notes de l'étudiant. Définissez cette méthode et testez-la dans le script `TestStudent.php`.

Question 17.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getFirstOccurrenceIndex` qui prend en paramètre une note. Elle retourne l'indice de la première occurrence de la note

[1] Rappel : l'égalité entre deux variables d_1 et d_2 de type *double*, ne peut pas être testée comme suit : $d_1 == d_2$. Ce test d'égalité sera remplacé par $abs(d_1 - d_2) < 10e - 5$.

parmi les notes de l'étudiant. Cette méthode lèvera une exception de type `UnexpectedValueException` si la note n'est pas présente. Définissez cette méthode et testez-la dans le script `TestStudent.php`.

Question 18.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getLastOccurrenceIndex` qui prend en paramètre une note. Elle retourne l'indice de la dernière occurrence de la note parmi les notes de l'étudiant. Cette méthode lèvera une exception de type `UnexpectedValueException` si la note n'est pas présente. Définissez cette méthode et testez-la dans le script `TestStudent.php`. Lors de la définition de cette méthode vous veillerez à effectuer un minimum de comparaisons.

Question 19.
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `swapMarks` qui prend en paramètre deux indices. Elle échange le contenu de deux cases de l'attribut d'instance `marks` dont les indices sont passés en paramètre. Si l'un de ces deux indices se trouve en dehors du tableau, cette méthode lancera une exception de type `OutOfRangeException`. Définissez cette méthode et testez-la dans le script `TestStudent.php`.

Question 20. ***Complémentaire***
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `enterMarks` qui ne prend rien en paramètre. Elle permet de saisir au clavier les notes d'un étudiant, tant que l'utilisateur ne passe pas à la ligne sans avoir saisi de valeur (touche *Entrée*). Proposez la définition de la méthode `enterMarks` puis testez-la dans le script `TestStudent.php`.

Question 21. ***Complémentaire***
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `addMark` qui prend en paramètre une nouvelle note et l'indice auquel devra se trouver cette note. Cette méthode préservera les notes précédemment obtenues par l'étudiant, elle rajoutera donc la nouvelle note à l'indice désiré. Si l'indice pris en paramètre est en dehors du tableau (à l'exception de la taille de ce dernier), cette méthode lancera une exception de type `OutOfRangeException`.

Question 22. ***Complémentaire***
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getMaximumIndexFrom` qui prend en paramètre l'indice qui marquera le début de la recherche. Cette méthode retourne l'indice de la note maximale obtenue par un étudiant, en débutant la recherche à l'indice pris en paramètre. Définissez cette méthode et testez-la dans le script `TestStudent.php`. Dans votre définition vous veillerez toujours à ne pas stocker la valeur du maximum dans une variable, seul l'indice sera stocké. Si l'indice pris en paramètre se trouve en dehors du tableau, cette méthode lancera une exception de type `OutOfRangeException`.

Question 23. ***Complémentaire***
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `sortMarks`. Cette méthode permet de trier dans l'ordre décroissant les notes d'un étudiant. L'algorithme de tri que vous implémenterez est le tri par sélection. Une description de cet algorithme est disponible à l'adresse :

https://fr.wikipedia.org/wiki/Tri_par_sélection

Faites dérouler l'algorithme sur les notes de l'instance `kevin`.

Définissez la méthode `sortMarks`. Vous pourrez pour ceci utiliser les méthodes `swapMarks` et `getMaximumIndexFrom` précédemment définies.

Question 24. ***Complémentaire***
On souhaiterait revoir la définition la méthode `isEqual`. Cette méthode permet de vérifier si deux étudiants ont obtenu les mêmes notes, cette fois-ci les notes pouvant être ordonnées différemment pour les deux étudiants.

Question 25. ***Complémentaire***
Dans la classe `Student`, on souhaite disposer de la méthode d'instance `getHistogram`. Cette méthode permet de déterminer la distribution des notes d'un étudiant. Elle détermine le nombre de notes comprises entre 0 (inclus) et 1 (exclu), entre 1 (inclus) et 2 (exclu), ..., entre 19 (inclus) et 20 (exclu) ou égales à 20. Elle retourne cette distribution sous forme d'un tableau d'entiers. Définissez la méthode `getHistogram`.

Proposez une méthode de d'instance `printHistogram` qui calcule l'histogramme puis l'affiche dans la console. Pour l'histogramme :
`[ 2, 0, 0, 1, 0, 0, 3, 0, 2, 4, 5, 7, 0, 4, 3, 0, 0, 0, 3, 0, 1 ]`,
le rendu désiré est :

```
0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20
*        *        *     *  *  *  *     *  *           *     *
*                 *     *  *  *  *     *  *           *
                  *        *  *  *     *  *           *
                           *  *  *     *
                              *  *
                                 *
                                 *
```