W1 - Piscine PHP

W-WEB-024

Jour 12

Le dernier jour

2.0

Jour 12

repository name: poolphpday12
language: PHP

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

INFORMATIONS

AVANT DE COMMENCER

- Lisez attentivement toutes les consignes.
- Consultez vos mails plusieurs fois par jour, tous les jours.

Commencez par lire vos mails tout de suite à l'adresse : mail.office365.com.

- C'est une pangolinette (un programme) qui corrige vos exercices. Vérifiez le travail que vous allez rendre afin qu'il respecte scrupuleusement les consignes.
- Vous devez respecter les restrictions qui sont imposées dans chaque exercice. Le cas contraire, la pangolinette va considérer comme **triche** en attribuant la note de **-42**.
- Vous êtes susceptibles à tout moment de recevoir des corrections intermédiaires

Pour bénéficier de corrections intermédiaires, vous devez chaque jour :
- Etre inscrit au projet et aux activités dans l'intranet.
- Tenir à jour régulièrement le dépôt.

- Ne laissez jamais votre session ouverte sans surveillance.

JOUR 12

Votre répertoire ne doit pas contenir de fichiers inutiles (fichiers temporaires, …)
N'oubliez pas de push régulièrement vos fichiers, sans cela, pas de correction

Pensez à envoyer votre travail via **git!**
Le nom du rendu est spécifié dans les instructions pour chaque étape/exercice.
Pour garder votre répertoire propre, regardez du côté de `gitignore`.

N'oubliez pas de vous inscrire à toutes les activités possibles de la semaine.

ÉTAPE 1 - (2 POINTS)

Nom de rendu : ex_01.php
Restrictions : Aucune

Faire une fonction nommée **"printArray"**, qui prend en paramètre un tableau et qui affiche les clés et valeurs de ce dernier sous la forme suivante (en remplaçant les crochets par les éléments correspondants) : **"[cle] => [valeur]n"**

Prototype : void printArray(array $tableau);

ÉTAPE 2 - (2 POINTS)

Nom de rendu : ex_02.php
Restrictions : Aucune

Faire une fonction nommée **"cesar2"**, qui prend en paramètre une chaîne de caractères et qui affiche cette chaîne en remplaçant chacun des caractères par le caractère situé deux positions plus loin dans l'ordre alphabétique (a devient c, b devient d…), suivi d'un **"\n"**.
Les majuscules deviennent des minuscules.
Les minuscules restent des minuscules. Le **"y"** devient **"a"** et le **"z"** devient **"b"**.

Prototype : void cesar2(string $chaine);

ÉTAPE 3 - (2 POINTS)

Nom de rendu : ex_03.php
Restrictions : Aucune

Faire une fonction nommée **"rev_epur_str"**, qui prend en paramètre une chaîne de caractères et qui renvoie une chaîne de caractère, en renversant l'ordre de tous les mots de cette chaîne.
Les séparateurs de mots sont les espaces, les tabulations et les retours à la ligne.
La fonction ne retourne ni de tabulations, ni plusieurs espaces consécutifs, ni d'espace en début ou en fin de chaîne. Si la chaîne est manquante, la fonction retourne FALSE.

Prototype : mixed rev_epur_str(string $chaine);

ÉTAPE 4 - (2 POINTS)

Nom de rendu : ex_04.php
Restrictions : Aucune

Faire une fonction nommée **"resum_join_str"**, qui prend en paramètres deux chaînes de caractères et qui renvoie les 12 premiers caractères de la première chaîne, suivi par les 8 derniers caractères de la deuxième chaîne.
Si jamais la première chaîne fait moins de 12 caractères ou si la seconde fait moins de 8 caractères, il faut mettre un point pour chaque caractère manquant.
Si l'une des deux chaînes est manquante, la fonction doit retourner false.

Prototype : mixed resum_join_str(string $chaine_1, string $chaine_2);

ÉTAPE 5 - (2 POINTS)

Nom de rendu : ex_05.php
Restrictions : Aucune

Faire une fonction nommée **"my_sort"**, qui prend en paramètre la référence d'un tableau. Cette fonction doit renvoyer le nombre d'appel à celle-ci.
L'effet de cette fonction est d'inverser le premier élément avec le dernier élément du tableau.

Prototype : int my_sort(array &$tableau);

ÉTAPE 6 - (2 POINTS)

Nom de rendu : ex_06/character.class.php
Restrictions : Aucune

Faire une classe nommée **"Character"**, qui comporte les attributs privés **"name"**, **"strength"**, **"magic"**, **"intelligence"** et **"life"**.
Les valeurs par défaut lors de l'instanciation seront respectivement **"Character $i"**, **0**, **0**, **0** et **100**.
Faire les méthodes protégées **"getName"**, **"getStrength"**, **"getMagic"**, **"getIntelligence"** et **"getLife"**.
Faire une implémentation de la méthode **__toString** afin de satisfaire l'exemple affiché.

Prototype : __construct([string $name]);
Exemple :

Terminal

```
addPlayer(new Character);
echo $game->player(0);
$game->addPlayer(new Character);
game->player(0)->attack(game->player(1));
$game->addPlayer(new Character("Julien"));
game->player(2)->attack(game->player(0));
?>
```

Ce code affiche :

```
Terminal
New game !
New player "Character 1".
My name is Character 1.
New player "Character 2".
"Character 1" hits "Character 2".
"Character 2" loses 0 HP!
New player "Julien".
"Julien" hits "Character 1".
"Character 1" loses 0 HP!
```

Vous devrez implémenter la méthode “getPublicName” et “attack” de la classe “Character” pour mener à bien cet exercice.

N’oubliez pas d’inclure la classe “Character”.

ÉTAPE 8 - (2 POINTS)

Nom de rendu :

- ex_08/troll.class.php
- ex_08/goblin.class.php
- ex_08/pangolin.class.php

Restrictions : Ne pas modifier la visibilité des méthodes et attributs précédents.

Déposer les fichiers de l’exercice précédent dans le nouveau dossier.
Finalement, nous ne voulons plus qu’il soit possible de créer directement des personnages, mais des trolls, des gobelins et des pangolins.
Les caractéristiques sont les suivantes :

Character	Troll	Goblin	Pangolin
Default name	Troll $i	Goblin $j	Pangolin $k
Default strength	50	35	40
Default magic	15	20	10

Character	Troll	Goblin	Pangolin
Default intelligence	10	40	55
Default life	200	150	120

Modifiez les classes **“Character”** et **“Game”** de façon à respecter l’exemple affiché.

Exemple :

Terminal

```
addPlayer(new Goblin);
echo $game->player(0);
$game->addPlayer(new Troll);
game->player(1)->attack(game->player(0));
game->player(1)->attack(game->player(0));
$game->addPlayer(new Pangolin("Julien"));
game->player(2)->heal(game->player(0));
game->player(0)->attack(game->player(2));
$game->player(2)->heal();
game->player(2)->heal(game->player(2));
game->player(1)->attack(game->player(0));
game->player(1)->attack(game->player(0));
game->player(2)->attack(game->player(0));
?>
```

Ce code affiche :

Terminal

```
New game !
New player "Goblin 1".
My name is Goblin 1.
New player "Troll 1".
"Troll 1" hits "Goblin 1".
"Goblin 1" loses 50 HP!
"Troll 1" hits "Goblin 1".
"Goblin 1" loses 50 HP!
New player "Julien".
"Julien" heals "Goblin" for 55 HP.
"Goblin 1" hits "Julien".
"Julien" loses 20 HP!
"Julien" heals himself for 20 HP.
"Julien" heals himself for 0 HP.
"Troll 1" hits "Goblin 1".
"Goblin 1" loses 50 HP!
"Troll 1" hits "Goblin 1".
"Goblin 1" loses 50 HP!
"Julien" hits "Goblin 1".
"Goblin 1" loses 5 HP!
"Goblin 1" died.
```

Il faut abstraire la classe “Character” et utiliser l’héritage.

N’oubliez pas d’inclure toutes vos autres classes dans “Game”.

ÉTAPE 9 - (2 POINTS)

Nom de rendu :

- ex_09/igame.class.php
- ex_09/icharacter.class.php
- ex_09/itroll.class.php
- ex_09/igobelin.class.php
- ex_09/ipangolin.class.php

Restrictions : Ne pas modifier la visibilité des méthodes et attributs précédents.

Déposer les fichiers de l’exercice précédent dans le nouveau dossier. Créer toutes les interfaces et implémentez-les.

N’oubliez pas les méthodes magiques !

ÉTAPE 10 - (2 POINTS)

Nom de rendu : ex_10.php
Restrictions : Aucune

Faire un programme en **"CLI"** (Command-line Interface), qui prend plusieurs chaînes de caractères en paramètres et qui affiche le **"sha1"** de chacune d'entre elles, suivi d'un retour à la ligne.

Exemple :

Terminal

```
~/W-WEB-024> php ex_10.php "dernier exo" "ouf c'est fini" "cool"
5715f6aa2fdb34c05dd383c304c74735a259a517
22f1783f90878b6889c72506a53748b845e958a5
85d8d76ba15bde3ef1602f477f32fd64e32fea5a
~/W-WEB-024>
```