



# W1 - Piscine PHP

---

W-WEB-024

## Jour 10

---

Programmation orientée objet



# Jour 10

repository name: poolphpday10  
repository rights: ramassage-tek  
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](mailto: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 10

---



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 créer votre répertoire en début de journée et à envoyer votre travail via git! Le nom du répertoire 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 (2PTS)

---

Nom de rendu : ex\_01.php

Restrictions : Aucune.

Créez une classe abstraite **"Warrior"**. Elle aura une méthode publique nommée **"attack"** qui affichera **"I'll kill you, poor noob !"** suivi d'un retour à la ligne, et 2 méthodes publiques abstraites: **"represent"** et **"shout"**. Créez une classe **"Booba"** qui hérite de **"Warrior"**. Sa méthode **"represent"** devra afficher **"92"** suivi d'un retour à la ligne, et sa méthode **"shout"** devra afficher **"Bah bien Morray !"**. Créez une classe **"LaFouine"** qui hérite de **"Warrior"**. Sa méthode **"represent"** devra afficher **"78"** suivi d'un retour à la ligne, et sa méthode **"shout"** devra afficher **"Je suis propriétaire !"**.



abstract methods

## ÉTAPE 2 (2PTS)

---

Nom de rendu : ex\_O2.php

Restrictions : Aucune.

Copiez le code de votre exercice 1. Vous allez apprendre les bonnes manières à “Booba”.

Créez une interface “**GoodManners**” qui contiendra 3 méthodes publiques : “say\_please”, “say\_thanks” et “say\_sorry”.

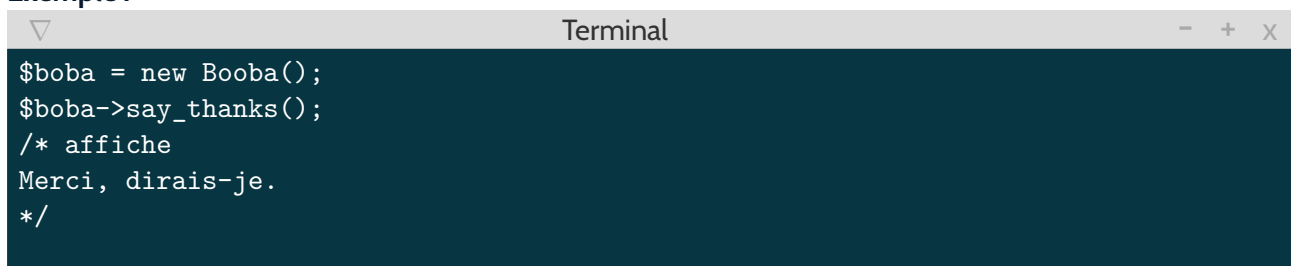
Cette dernière méthode prendra un paramètre nommé “name”.

L'interface contiendra aussi une constante nommée **END\_WORDS** qui sera égale à “, dirais-je.”.

La classe “**Booba**” implémentera cette interface avec ses méthodes. Sa méthode “say\_please” affichera “S’il-te-plait”, sa méthode “say\_thanks” affichera “Merci”, et sa méthode “say\_sorry” prendra un nom en paramètre et affichera “Mille excuses,” suivi du nom.

Chacun de ces affichages se termineront par la constante **END\_WORDS** suivi d'un retour à la ligne.

Exemple :

A terminal window titled "Terminal" with standard window controls (minimize, maximize, close). It displays the following PHP code and its output:

```
$boba = new Booba();
$boba->say_thanks();
/* affiche
Merci, dirais-je.
*/
```

## ÉTAPE 3 (2PTS)

---

Nom de rendu : ex\_O3.php

Restrictions ::

- Utiliser le Trait avec le mot-clé “use”.
- Utiliser la fonction **str\_replace** 2 fois.
- Ne pas écrire 2 fois la fonction **sing\_a\_song**

Copiez le code de votre exercice 2. Vous allez apprendre à chanter à vos 2 warriors.

Créez un trait “**Singer**” qui contiendra une fonction “**sing\_a\_song**”. Cette fonction affichera la chaîne de caractères “**Trololololololol**” suivi d'un retour à la ligne, puis transformera tous les ‘o’ en ‘a’ et réaffichera la chaîne suivi d'un retour à la ligne, pour enfin transformer tous les ‘a’ en ‘i’ et réaffichera la chaîne suivi d'un retour à la ligne



traits

## ÉTAPE 4 (2PTS)

---

Nom de rendu : ex\_O4.php

Restrictions :: Aucune.

Créez une classe **"Pony"** qui aura 3 attributs publics: **"name"**, **"gender"** et **"color"**.

Faites en sorte qu'on puisse setter ces 3 attributs directement à la construction (dans le même ordre que cité précédemment).

Faites en sorte, qu'à la destruction du poney, la chaîne de caractères **"I'm a dead pony."** suivi d'un retour à la ligne soit affichée.

Faites aussi en sorte que lorsqu'on utilise la commande **"echo"** sur un **"Pony"**, cela affiche **"Don't worry, I'm a pony !"** suivi d'un retour à la ligne.



Magic methods



## ÉTAPE 5 (2PTS)

---

Nom de rendu : ex\_O5.php

Restrictions :: Aucune.

Copiez le code de votre exercice 4.

Ajoutez une méthode publique nommée **“speak”** qui affiche **“Hiii hiii hiiii”** suivi d'un retour à la ligne.

Faites en sorte que lorsqu'on appelle une méthode qui n'existe pas dans la classe **“Pony”**, cela affiche **“I don't know what to do...”** suivi d'un retour à la ligne.





## ÉTAPE 6 (2PTS)

---

Nom de rendu : ex\_O6.php

Restrictions ::

- Vous devrez utiliser `**'__get'`
- Vous devrez utiliser `'__set'`

Copiez le code de votre exercice 5.

Mettez tous les attributs en privé, mais faites en sorte que l'on puisse quand même y accéder via get et set.

Quand on y accède en get, cela doit afficher "Ce n'est pas bien de getter un attribut qui est privé !" suivi d'un retour à la ligne, puis retourner l'attribut.

Quand on y accède en set, cela doit afficher "**Ce n'est pas bien de setter un attribut qui est privé !**" suivi d'un retour à la ligne, puis affecter l'attribut de la valeur.

Pour tout get ou set d'un attribut qui n'existe pas, cela doit afficher "**Il n'y a pas d'attribut :**" suivi du nom de l'attribut, d'un point et d'un retour à la ligne.

## ÉTAPE 7 (2PTS)

---

Nom de rendu : ex\_07.php

Restrictions :: Aucune.

Créez une classe **"Body"** qui aura dans le même ordre: un attribut private **"head"** qui vaudra 1; Et un attribut protected **"arm"**, un attribut public **"hand"**, un attribut protected **"leg"** et un attribut public **"foot"** qui vaudront tous 2.

Cette classe aura une fonction publique **"print\_inside\_attributes"** qui affichera dans un **"foreach"** tous les attributs à l'aide de parcours d'objets. L'affichage de chaque attribut se fera de la façon suivante: **"attribut: valeur"**, suivi d'un retour à la ligne.

En dehors de cette classe, créez une fonction **"print\_object\_attributes"** qui prendra en paramètre un objet nommé **"object"**, et qui affichera dans un **"foreach"** tous les attributs public de cet objet à l'aide du parcours d'objets.

L'affichage de chaque attribut se fera de la façon suivante: **"attribut => valeur"**, suivi d'un retour à la ligne.



get\_object\_vars



get\_class\_methods

## ÉTAPE 8 (2PTS)

---

Nom de rendu : ex\_O8.php

Restrictions :: Vous devrez utiliser le mot-clé **“clone”**.

Créez une classe **“Dolly”** qui aura 3 attributs publics: **“animal”**, **“age”** et **“doctor”** .Le constructeur prendra ces 3 attributs en paramètres (dans le même ordre que cité précédemment).

Faites en sorte que lorsque qu'on clone un objet de cette classe, cela affiche **“I will survive !”** suivi d'un retour à la ligne.

En dehors de cette classe, créez une fonction **“clone\_object”** qui prendra en paramètre un objet nommé **“object”** et qui retournera un clone de ce même objet.



clone

## ÉTAPE 9 (2PTS)

---

Nom de rendu : ex\_O9.php

Restrictions :: Aucune.

Créez une fonction “**objects\_comparison**” qui prendra 2 objets en paramètres nommés respectivement “**object1**” et “**object2**”.

Elle devra afficher “**Objects are equal.**” suivi d’un retour à la ligne, si les 2 objets sont égaux ou “**Objects are the same.**” suivi d’un retour à la ligne, si les 2 objets font référence à la même instance ou sinon dans les autres cas affichera un simple retour à la ligne.



objects comparaison

## ÉTAPE 10 (2PTS)

---

Nom de rendu : ex\_10.php

Restrictions :: Aucune.

Créez une classe “**Pangolin**” qui aura 2 attributs publics : “**friends**” de type array et “**skills**” de type Skill qui sera une classe définie par nos soins.

Le constructeur prendra ces 2 attributs en paramètre (dans le même ordre que cité précédemment).

Faites en sorte qu’il ne soit pas possible de passer un autre type que le type initial de chaque attribut.

De plus, on devra pouvoir passer le type “**null**” à l’argument qui correspond à l’attribut “**friends**”.