



Programmation orientée objet en PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [POO PHP](#) ► La programmation orientée objet : premiers pas ► [Histoire de références](#)

Histoire de références

Dans un fichier lanceur.php nous allons instancier les classes Adresse et Personne.

```
1 <?php
2
3 // lanceur.php
4 include './Adresse.php';
5 include './Personne.php';
6
7 // création d'une instance d'adresse
8 $adr = new Adresse('Lignier', 'Paris');
9 // création d'une instance de Personne avec son adresse
10 $pers = new Personne('Sparrow', 'Jack', 42, $adr);
```

La méthode `getAdressePrincipale()` de `Personne` permet de récupérer l'objet `Adresse`, et la méthode `getRue()` d'`Adresse` permet de récupérer la rue.

```
1 <?php
2
3 // lanceur.php
4 include './Adresse.php';
5 include './Personne.php';
6
7 // création d'une instance d'adresse
8 $adr = new Adresse('Lignier', 'Paris');
9
10 // création d'une instance de Personne avec son adresse
11 $pers = new Personne('Sparrow', 'Jack', 42, $adr);
12
13 // récupération de l'adresse de pers via le getter
14 $adresse_de_la_personne = $pers->getAdressePrincipale();
15
16 // récupération et affichage de la rue via le getter
17 echo $adresse_de_la_personne->getRue();
18
19 // ou en une seule ligne
20 echo $pers->getAdressePrincipale()->getRue();
```

Dans cet exemple, l'attribut `adressePrincipale` de l'objet `pers` et la variable `adresse_de_la_personne` indiquent le même objet, ce sont en fait deux références au même objet, c'est-à-dire deux "flèches" qui indiquent le même objet en mémoire.

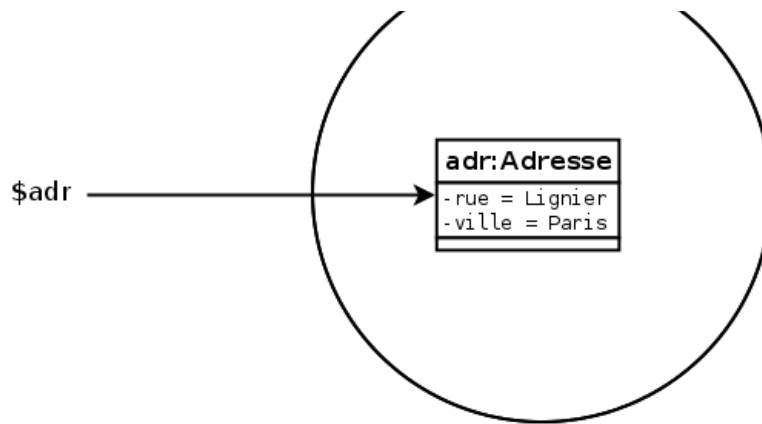
Quand nousinstancions une classe, nous créons un objet dans une zone particulière de la mémoire que nous appellerons le "tas". Les objets vivent donc sur le tas. Les variables de notre script seront, elles, créées dans un autre endroit de la mémoire que nous appellerons la "pile". Les termes de "tas" et de "pile" ont été choisis pour simplifier le débat...

Dans notre lanceur, l'instruction

```
$adr = new Adresse('Lignier', 'Paris');
```

crée une variable dans la pile qui sera une référence au nouvel objet de type `Adresse` sur le tas.

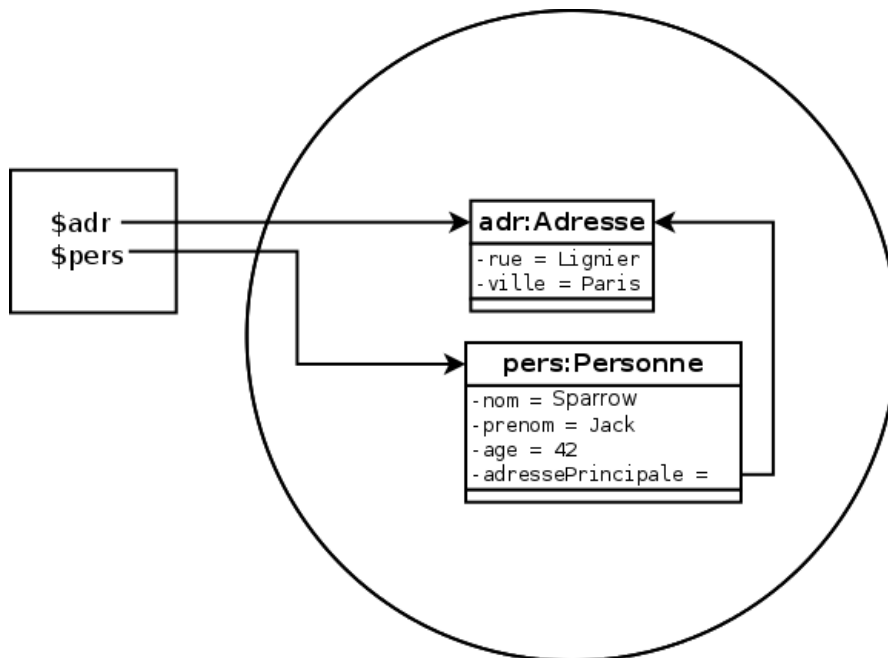




L'instruction

```
$pers = new Personne('Sparrow', 'Jack', 42, $adr);
```

va créer une référence `pers` à un objet de type `Personne`, cet objet possédera en attribut d'instance l'objet référencé par `adr`. L'attribut `adressePrincipale` de l'objet `pers` sera donc une autre référence à l'objet de type `Adresse` créé sur le tas.

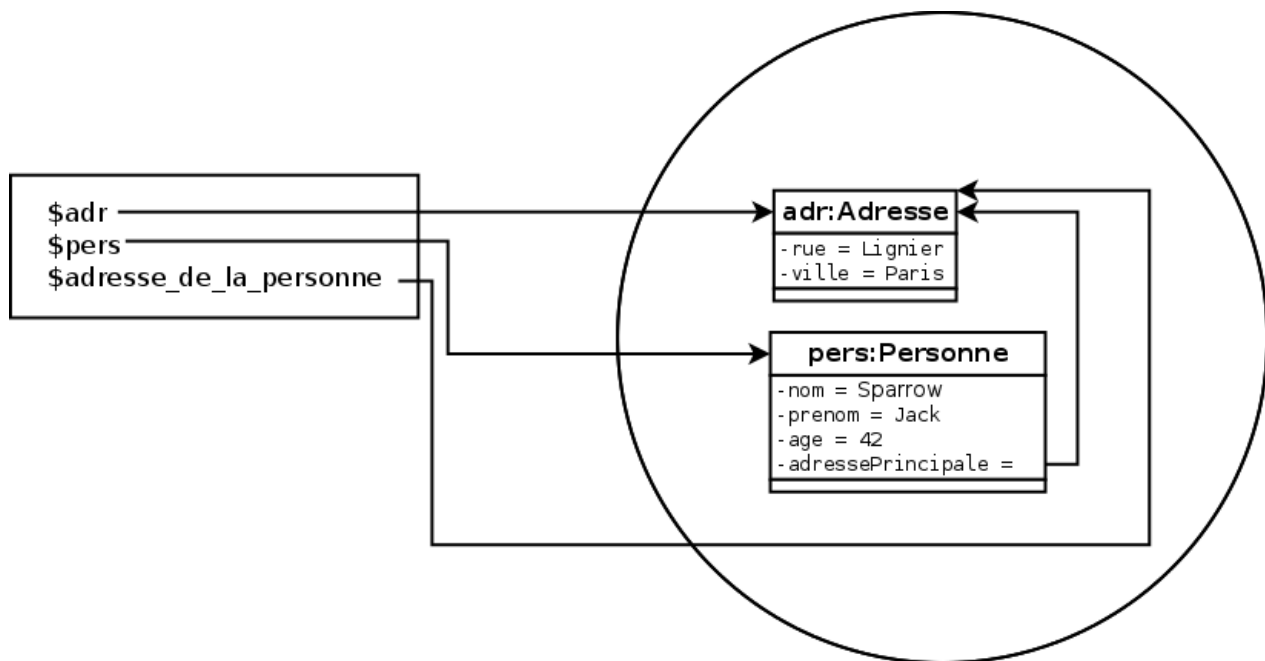


La variable `$adr` et l'attribut d'instance `adressePrincipale` font tous les deux références au même objet. Ce sont en fait des "flèches" qui indiquent un objet sur le tas, nous pouvons donc accéder à cet objet en utilisant l'une ou l'autre de ces références.

L'instruction

```
$adresse_de_la_personne = $pers->getAdressePrincipale();
```

créé une variable `adresse_de_la_personne` et lui assigne la valeur renvoyée par le getter, c'est-à-dire une référence à l'objet `Adresse` de `pers`.



La variable `adresse_de_la_personne` constitue donc une troisième référence à l'objet `adr` de type `Adresse`.

Pour en finir (provisoirement) avec les références, je vous rappelle que dans une fonction les objets en argument sont passés par référence. Cela signifie que dans une fonction, ou méthode, le traitement s'effectue sur le même objet que celui passé en argument et non sur une copie.

Voici un exemple : nous allons créer dans notre lanceur une fonction `modifierAge()` qui prendra en paramètre un objet de type `Personne` et un entier `age` et qui assignera à l'attribut d'instance `age` de `Personne` la valeur `age`. Puis nous afficherons la valeur de l'age de la `Personne` :

```

1  <?php
2
3  // lanceur.php
4  include './Adresse.php';
5  include './Personne.php';
6
7  // création d'une instance d'adresse
8  $adr = new Adresse('Lignier', 'Paris');
9
10 // création d'une instance de Personne avec son adresse
11 $pers = new Personne('Sparrow', 'Jack', 42, $adr);
12
13 // récupération de l'adresse de pers via le getter
14 $adresse_de_la_personne = $pers->getAdressePrincipale();
15
16 // récupération et affichage de la rue via le getter
17 echo $adresse_de_la_personne->getRue();
18
19 // ou en une seule ligne
20 echo $pers->getAdressePrincipale()->getRue();
21
22 function modifierAge($personne, $age) {
23     $personne->setAge($age);
24 }
25
26 modifierAge($pers, 789);
27 echo $pers->getAge();

```

Fin

NAVIGATION

[Accueil](#)

■ [Ma page](#)



Pages du site

Mon profil

Cours actuel

POO PHP

Participants

Généralités

La programmation orientée objet : premiers pas

 [Introduction](#)

 [Classes et objets](#)

 [T.P. première classe](#)


 [Les classes en PHP](#)

 [T.P. Personne](#)

 [Le constructeur et this](#)

 [T.P. Personne v2](#)

 [Accesseurs et mutateurs](#)

 [Les méthodes magiques](#)

 [T.P. Personne v3](#)

 [Les constantes de classe](#)

 [Le modificateur static](#)

 [T.P. Personne v4](#)

 [Une Personne et une Adresse : la relation has-a](#)

 **[Histoire de références](#)**

 [T.P. formation](#)

 [T.P. formation v2](#)

 [Une Personne et plusieurs Adresse\(s\)](#)

 [T.P. formation v3](#)

 [Bricolage et dépendance](#)

L'héritage

Les interfaces

Le typage

Les namespaces

Les exceptions

Les bases de données avec PDO

Les tests avec PHPUnit

Petite application version 2

Petite application version 3

[Mes cours](#)

ADMINISTRATION



Administration du cours

Réglages de mon profil

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))
[POO PHP](#)