



Programmation orientée objet en PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [POO PHP](#) ► La programmation orientée objet : premiers pas ► [Les classes en PHP](#)

Les classes en PHP

Même s'il est possible en PHP de créer plusieurs classes dans un seul fichier source, il est préférable de ne créer qu'une seule classe par fichier source. Il est aussi préférable de donner au fichier le même nom qu'à la classe : une classe Humain dans un fichier "Humain.php".

Le nom de la classe commencera toujours par une majuscule et sera écrit en camelCase (les mots seront collés et chaque première lettre de mot sera écrite en majuscule).

Pour créer une classe nous utiliserons le mot clé "class" suivi du nom de la classe écrit en camelCase. Ensuite, entre accolades, nous créerons les attributs d'instance sous forme de variables puis les méthodes d'instance sous forme de fonctions. Les noms des attributs d'instance et méthodes d'instance seront écrits en camelCase mais la première lettre sera toujours en minuscule. Le mot clé public devant les attributs et méthodes a une signification particulière que nous aborderons plus tard.

```
1 <?php
2
3 // Humain.php
4 class Humain {
5
6     public $age;
7     public $taille;
8     public $poids;
9
10    public function marcher() {
11        echo 'je marche';
12    }
13
14    public function courir() {
15        echo 'je cours';
16    }
17
18    public function parler() {
19        echo 'je parle';
20    }
21
22 }
```

Pour créer une instance de classe Humain, nous allons utiliser l'opérateur "new" et une "méthode" spéciale qui s'appelle un constructeur. Pour appeler ce constructeur nous utiliserons le nom de la classe suivi de parenthèses.

```
1 <?php
2
3 // Humain.php
4 class Humain {
5
6     public $age;
7     public $taille;
8     public $poids;
9
10    public function marcher() {
11        echo 'je marche';
12    }
13
14    public function courir() {
15        echo 'je cours';
```

```

16     }
17
18     public function parler() {
19         echo 'je parle';
20     }
21
22 }
23
24 $sparrow = new Humain();

```

La variable `$sparrow` est une référence à un objet de type Humain.

Pour accéder en lecture et écriture aux attributs, nous utiliserons l'opérateur `"->"` à partir de la référence. Pour l'instant, les attributs de `sparrow` n'ont pas de valeurs (ils sont null). Nous allons assigner une valeur aux attributs de `sparrow` puis afficher ces valeurs :

```

1 <?php
2
3 // Humain.php
4 class Humain {
5
6     public $age;
7     public $taille;
8     public $poids;
9
10    public function marcher() {
11        echo 'je marche';
12    }
13
14    public function courir() {
15        echo 'je cours';
16    }
17
18    public function parler() {
19        echo 'je parle';
20    }
21
22 }
23
24 $sparrow = new Humain();
25 $sparrow->age = 42;
26 $sparrow->taille = 1.84;
27 $sparrow->poids = 75;
28 echo "age = $sparrow->age" . PHP_EOL;
29 echo "taille = $sparrow->taille" . PHP_EOL;
30 echo "poids = $sparrow->poids" . PHP_EOL;

```

Nous pouvons aussi appeler les méthodes en utilisant l'opérateur `"->"` et la référence.

```

1 <?php
2
3 // Humain.php
4 class Humain {
5
6     public $age;
7     public $taille;
8     public $poids;
9
10    public function marcher() {
11        echo 'je marche';
12    }
13
14    public function courir() {
15        echo 'je cours';
16    }

```

```

17
18     public function parler() {
19         echo 'je parle';
20     }
21
22 }
23
24 $sparrow = new Humain();
25 $sparrow->age = 42;
26 $sparrow->taille = 1.84;
27 $sparrow->poids = 75;
28 echo "age = $sparrow->age" . PHP_EOL;
29 echo "taille = $sparrow->taille" . PHP_EOL;
30 echo "poids = $sparrow->poids" . PHP_EOL;
31 $sparrow->courir();
32 echo PHP_EOL;
33 $sparrow->marcher();
34 echo PHP_EOL;
35 $sparrow->parler();

```

Lors de la déclaration des attributs, il est possible de leur assigner tout de suite une valeur, si cette valeur est de type integer, double, string, boolean.

```

1 <?php
2
3 // Humain.php
4 class Humain {
5
6     public $age = 42;
7     public $taille = 1.84;
8     public $poids = 75;
9
10    public function marcher() {
11        echo 'je marche';
12    }
13
14    public function courir() {
15        echo 'je cours';
16    }
17
18    public function parler() {
19        echo 'je parle';
20    }
21
22 }
23
24 $sparrow = new Humain();
25 echo "age = $sparrow->age" . PHP_EOL;
26 echo "taille = $sparrow->taille" . PHP_EOL;

```

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](#)