



# Programmation orientée objet en PHP

[Accueil](#) ▶ [Mes cours](#) ▶ [Développement logiciel](#) ▶ [POO PHP](#) ▶ [Le typage](#) ▶ [Le Type Hinting](#)

## Le Type Hinting

Le Type Hinting (ou "vérification" du type) permet d'indiquer explicitement le type d'un paramètre d'une méthode ou d'une fonction. Les types qui peuvent être indiqués sont : array, callable, les classes et les interfaces. Si le type du paramètre n'est pas satisfait alors une erreur fatale est déclenchée.

```
1 <?php
2
3 // Programmeur.php
4 interface Programmeur {
5
6     public function coder();
7
8     public function debogguer();
9 }
```

```
1 <?php
2
3 // Personne.php
4 class Personne implements Programmeur {
5
6     protected $nom;
7     protected $prenom;
8
9     function __construct($nom, $prenom) {
10         $this->nom = $nom;
11         $this->prenom = $prenom;
12     }
13
14     public function getNom() {
15         return $this->nom;
16     }
17
18     public function getPrenom() {
19         return $this->prenom;
20     }
21
22     public function setNom($nom) {
23         $this->nom = $nom;
24     }
25
26     public function setPrenom($prenom) {
27         $this->prenom = $prenom;
28     }
29
30     public function coder() {
31         echo 'je code...';
32     }
33
34     public function debogguer() {
35         echo 'je déboggue...';
36     }
37
38     public function parler() {
39         echo 'je parle...';
40     }
41 }
```

```
40     }
41
42 }
```

Par exemple, dans ce lanceur, la fonction `faireParler()` attend un argument de type `Personne`, la fonction `faireCoder()` attend un argument de type `Programmeur`.

```
1 <?php
2
3 // lanceur.php
4 include './Programmeur.php';
5 include './Personne.php';
6
7 function faireParler(Personne $personne) {
8     $personne->parler();
9 }
10
11 function faireCoder(Programmeur $dev) {
12     $dev->coder();
13 }
14
15 $pers = new Personne('Sparrow', 'Jack');
16 faireParler($pers);
17 faireCoder($pers);
```

Lorsque j'impose à une méthode un argument du type `Programmeur`, cela signifie que l'argument pourra être de n'importe quelle classe à partir du moment où cette classe implémente l'interface `Programmeur`. Si je crée une classe `Robot` qui implémente l'interface `Programmeur`, alors je pourrai passer une instance de `Robot` en argument de la méthode `faireCoder()`.

Fin

## NAVIGATION

### Accueil

#### ■ Ma page

Pages du site

Mon profil

Cours actuel

#### POO PHP

Participants

Généralités

La programmation orientée objet : premiers pas


L'héritage

Les interfaces

Le typage

 [Introduction](#)

 [Le Type Hinting](#)

 [T.P. 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](#)