# PROGRAMMATION ORIENTÉE OBJET : POO

**Les classes abstraites et Interface**

AFCI
CENTRE DE FORMATION

# Définition : Classe Abstraite

Une classe abstraite est une classe racine non instanciable, c'est-à-dire qu'on ne va pas pouvoir manipuler directement, et elle contient des méthodes abstraites.

L'idée ici va donc être de définir des classes mères abstraites et de pousser les développeurs à étendre ces classes.

Lors de l'héritage d'une classe abstraite, les méthodes déclarées comme abstraites dans la classe Parent doivent obligatoirement être définies dans la classe enfant avec des signatures(nom et paramètres) correspondantes.

Cette façon de faire va être utile pour fournir un rail, c'est-à-dire une ligne directrice dans La cas de développement futures.

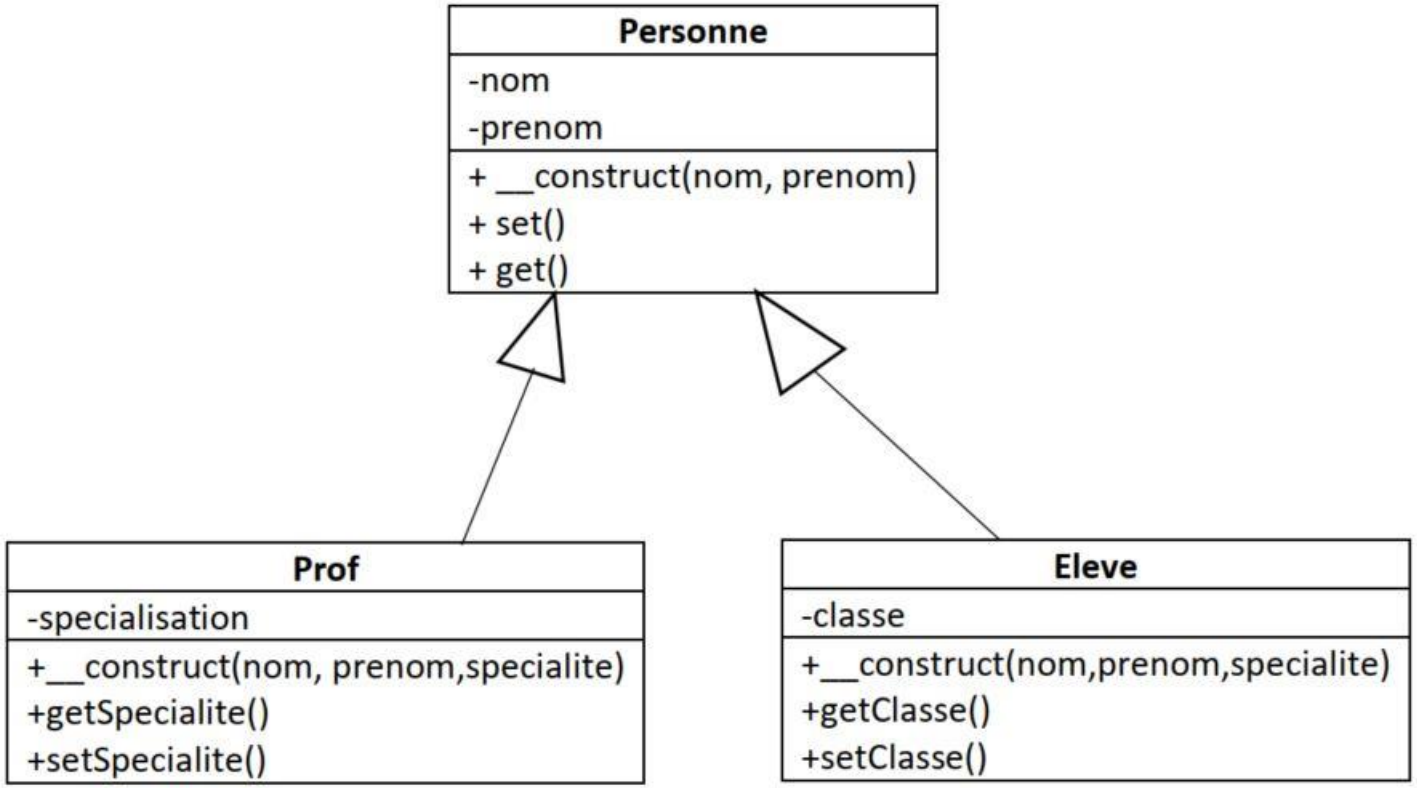
La syntaxe est la suivante :

**abstract** class Personne{…….}

La classe possède une méthode __construct() qui est ***« protected »*** et qui sera utilisée par les classes héritières.

AFCI
CENTRE DE FORMATION

# Classe Abstraite : Exemple




AFCI
CENTRE DE FORMATION

# L’interface :

L’interface est assez proche de la classe abstraite, mais aucune méthode n’est implémentée dedans.
Il y a juste la déclaration(la signature) des méthodes. Ainsi une classe qui implémente une Interface, devra obligatoirement implémenter les méthodes.

Pour mieux comprendre le principe :

- Supposons que nous avons une classe abstraite **« Vehicule »** et **« Maison »**
- Les classes filles qui peuvent en hériter sont par exemple : Voiture – Camion – Fourgon
- Ces classes sont en réalité des véhicules plus ou moins différents
- Par contre les classes Vehicule et Maison ne sont pas de la même famille
- Par contre chacune d’entre elles peut accueillir une méthode du nom de peindre()
- C’est là où une interface qui renferme la méthode peindre peut être utile

AFCI
CENTRE DE FORMATION

# L’interface : Exemple

```php
<?php
interface Entretien{
    public function peindre($color);
    public function nettoyer();
}

?>
```

Pour définir une interface, on utilise le mot clé **« interface »,** suivi du nom.

```php
class Vehicule implements Entretien{
    private $couleur;
    private $propre;

    public function peindre($c){
        $this->couleur = $c;
    }

    public function nettoyer(){
        $this->propre = true;
    }
}
```

Pour se servir de l’interface, il faut l’implémenter par une classe à l’aide du mot **« implements ».**

Dans la classe **Vehicule**, on doit obligatoirement redéfinir les méthodes **peindre() et nettoyer()**

AFCI
CENTRE DE FORMATION

# Héritage des interfaces

```php
interface IntA{
    public function fontionA();
}

Interface IntB extends IntA{
    public function fonctionB($a,$b);
}

class MaClasse implements IntB{
    public function fontionA(){

    }
    public function fonctionB($d,$c){

    }
}
```

Comme pour les classes, une interface peut hériter d'une autre. Dans ce cas, la réécriture d'une méthode dans l'interface fille n'est pas autorisée.

Il faut réécrire les méthodes de l'interface mère et l'interface fille dans la classe qui implémente cette dernière.

AFCI
CENTRE DE FORMATION

# Implémenter plusieurs interfaces

```php
interface IntA{
    public function fA();
}

interface IntB{
    public function fB();
}

class MaClasse implements IntA, IntB{
    public function fA()
    {

    }

    public function fB(){

    }
}
```

A l'inverse de l'héritage , on peut implémenter plusieurs Interface simultanément.

Dans ce cas, dans la classe **« MaClasse »** on doit redéfinir les Méthodes fA() et fB() dont le prototype est enfin défini Respectivement dans les interfaces IntA et IntB.

AFCI
CENTRE DE FORMATION

# Constante d'interface

```php
interface Inter{
    const MACONSTANTE = "Bonjour";
}

class MaClasse implements Inter{

}

echo Inter::MACONSTANTE . "\n";
echo MaClasse::MACONSTANTE;

?>
```

Comme pour les classes, une constante d'interface appartient à cette dernière. Elle peut donc être appelée statiquement à partir de l'interface dans laquelle elle est déclarée ou de la classe qui implémente l'interface.

AFCI
CENTRE DE FORMATION