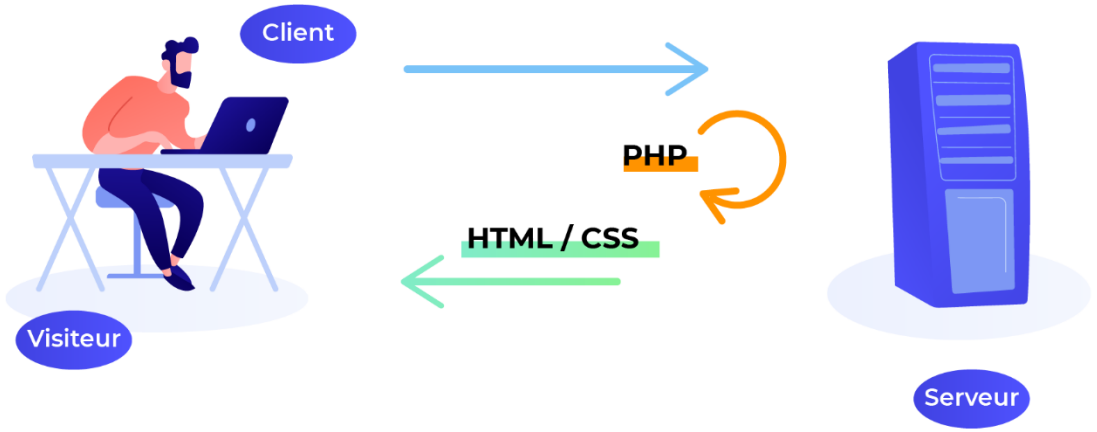
مدرسة علوم المعلومات
ⵜⵉⵣⵔⴰ ⵏⵜⵓⵙⵙⵉⵔⵉⵏ ⵉⵏⴼⵓⵔⵎⴰⵜⵉⵣⵏ
ECOLE DES SCIENCES DE L'INFORMATION
ISCHOOL.MA

Cours Programmation web en PHP



Semestre 2

2022/2023

Partie 3: Programmation Orientée Objet en PHP

- Programmation structurée vs Programmation Orientée Objet
- Classe & Objet
- Constructeur
- Visibilité des variables
- Héritage

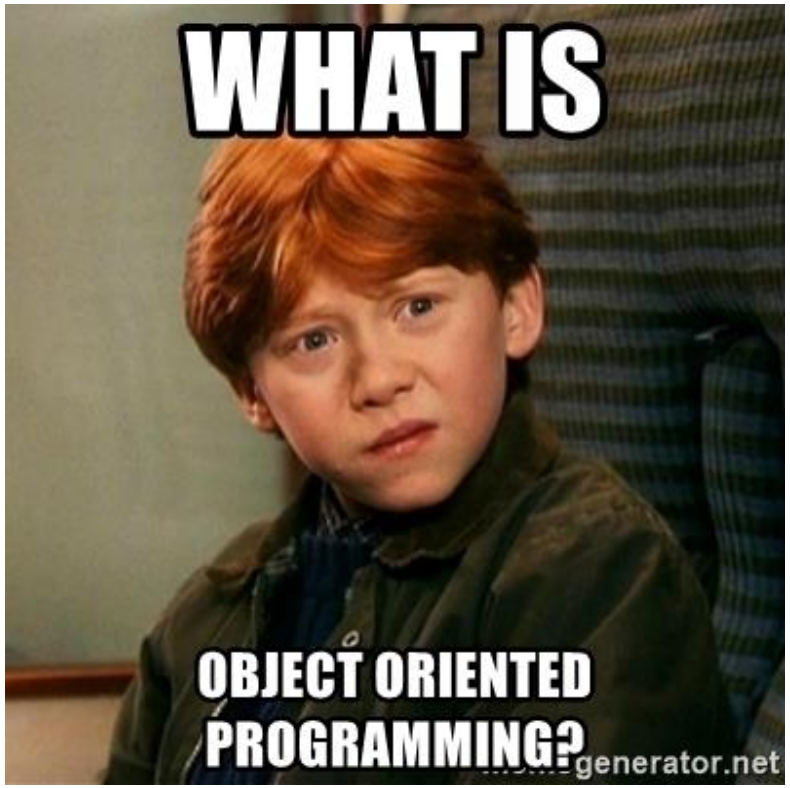
Programmation procédurale vs POO

Tout problème qui peut être résolu par un programme écrit dans le langage A peut aussi l'être par un programme rédigé en langage B. Il existe cependant:

- Différentes approches à la résolution algorithmique de problèmes,
- Différents mécanismes de programmation,
- Différents styles de programmation.
- Chacun de ces éléments peut influencer la qualité d'un programme: Correction, efficacité, modularité, concision, lisibilité, ergonomie, etc.

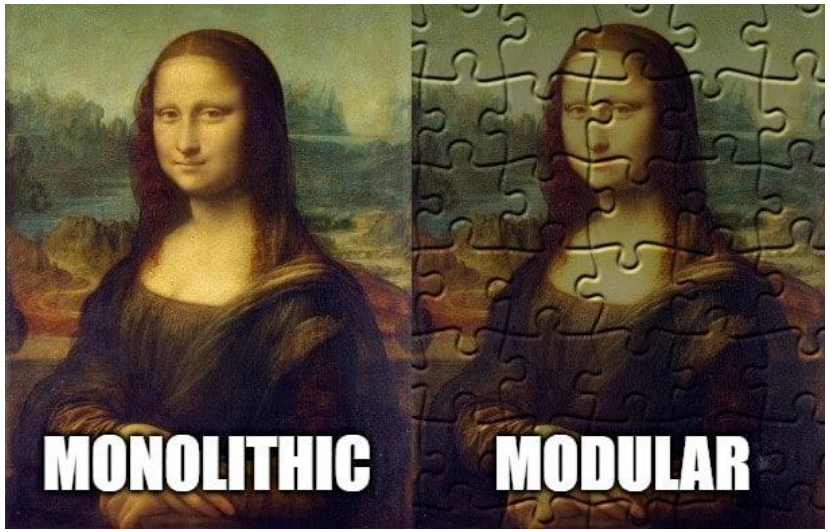
Programmation procédurale vs POO

- L'orienté objet est une approche à la résolution algorithmique de problèmes et à la programmation. Elle vise à produire des programmes possédant de bonnes qualités de modularité.
- Qu'est-ce-que la modularité ? Et quelles sont ses bénéfices?



Programmation procédurale vs POO

- Quand un projet prend de l'importance, et encore davantage quand il est réalisé en équipe, il y a tout l'intérêt à modulariser le code



Pour éviter un code spaghetti

- des modules INDEPENDANTS.
- Couplage faible
- le plus élémentaires possible.
- possédant leur propre vision des variables (visibilité)

Programmation procédurale vs POO

L'orienté objet est une approche à la résolution algorithmique de problèmes et à la programmation. Elle permet de:

- Rassembler autour d'un objet une définition: des attributs et des méthodes d'action;
- Apporter des modifications locales à un module, sans que cela affecte le reste du programme;
- Réutiliser les fragments de code développés dans un cadre différent.
- Développer une partie d'un programme sans qu'il soit nécessaire de connaitre les détails internes aux autres parties.

Programmation procédurale vs POO

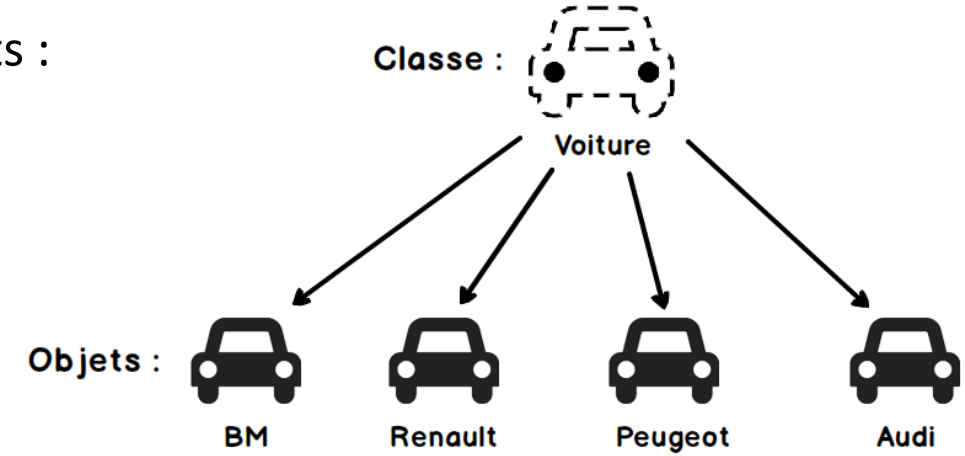
- D’où les avantages liés à:
  - La clarté du code
  - La modularité
  - La réutilisabilité
  - L’interopérabilité
  - Contribue à la fiabilité des logiciels.
- Elle introduit de nouveaux concepts, en particulier: les objets, les classes et l’héritage.

POO – Classe et objet

- L'orienté objet présente une approche proche des représentations réelles. La classe est un modèle/type de données. On peut la voir comme une famille d'objets

- **Une classe** contient des propriétés et des comportements :
  - Propriété = variable de la classe
  - Comportement = méthode de la classe

- **Un objet** est une instance de classe.
  - Avec une classe, on peut créer autant d'objets que l'on souhaite



POO – Classe et objet

- Déclarer une classe en php :

```
<?php
Class voiture
{
//contenu de la classe
}
?>
```

On commence par le mot-clé class suivi du nom de la classe

POO – Classe et objet

- Pour pouvoir utiliser un objet, il faut d'abord le créer à partir d'une classe.
- On utilise le mot clé ***new***;
- Une fois qu'un objet est instancié, on peut utiliser ses attributs et ses méthodes

```
<?php
Class voiture
{
//contenu de la classe
}
//Instanciation
$ma_voiture = new voiture();
?>
```

Instanciation de la classe voiture, c'est-à-dire création d'un objet de type voiture

Le mot-clé new permet de créer une instance de la classe (objet).
C'est à ce moment que le constructeur s'exécute, s'il y en a un dans la classe

POO – Classe et objet

- Le contenu de la classe est structuré en 2 parties: les attributs et les méthodes.

```php
<?php
Class voiture
{
Public $marque; //chaine de caractères
Private $nbre_chevaux= 0; //entier
Public $vitesse;//double
Public $prix; //flottant
Public $immatriculation;//chaine de caractères
    function affichage()
    {
        echo 'Bienvenue au bord de la voiture,
        la marque est'. $this->marque;
    }
}
$ma_voiture = new voiture();
$ma_voiture->marque = 'Ferrari';
$ma_voiture->affichage();
?>
```

Les attributs de la classe (propriétés)

Méthode de la classe

La définition de la classe se termine ici

Création d'un objet, accès aux attributs et aux méthodes

Exemple

```php
Class voiture
{
Public $marque = null; //chaine de caractères
Public $nbre_chevaux= 0; //entier
Public $vitesse;//double
Public $prix //flottant
Public $immatriculation//chaine de caractères
Function affichage()
{
echo "la voiture est une" . $this->marque;
}
}
$ma_voiture = new voiture();
$ma_voiture->marque = "Ferrari";
$ma_voiture->nbre_chevaux = 620;
$ma_voiture->affichage();
```

$this: est une métavariable qui est une référence permanente vers l'objet courant

Le constructeur

Une fonction qui est appelée automatiquement par la classe lors de son instanciation avec l'opérateur new.

En PHP, le constructeur est une méthode qui doit s'appeler __construct().

Elle prend en argument des variables dont le contenu sera affecté aux propriétés de la classe

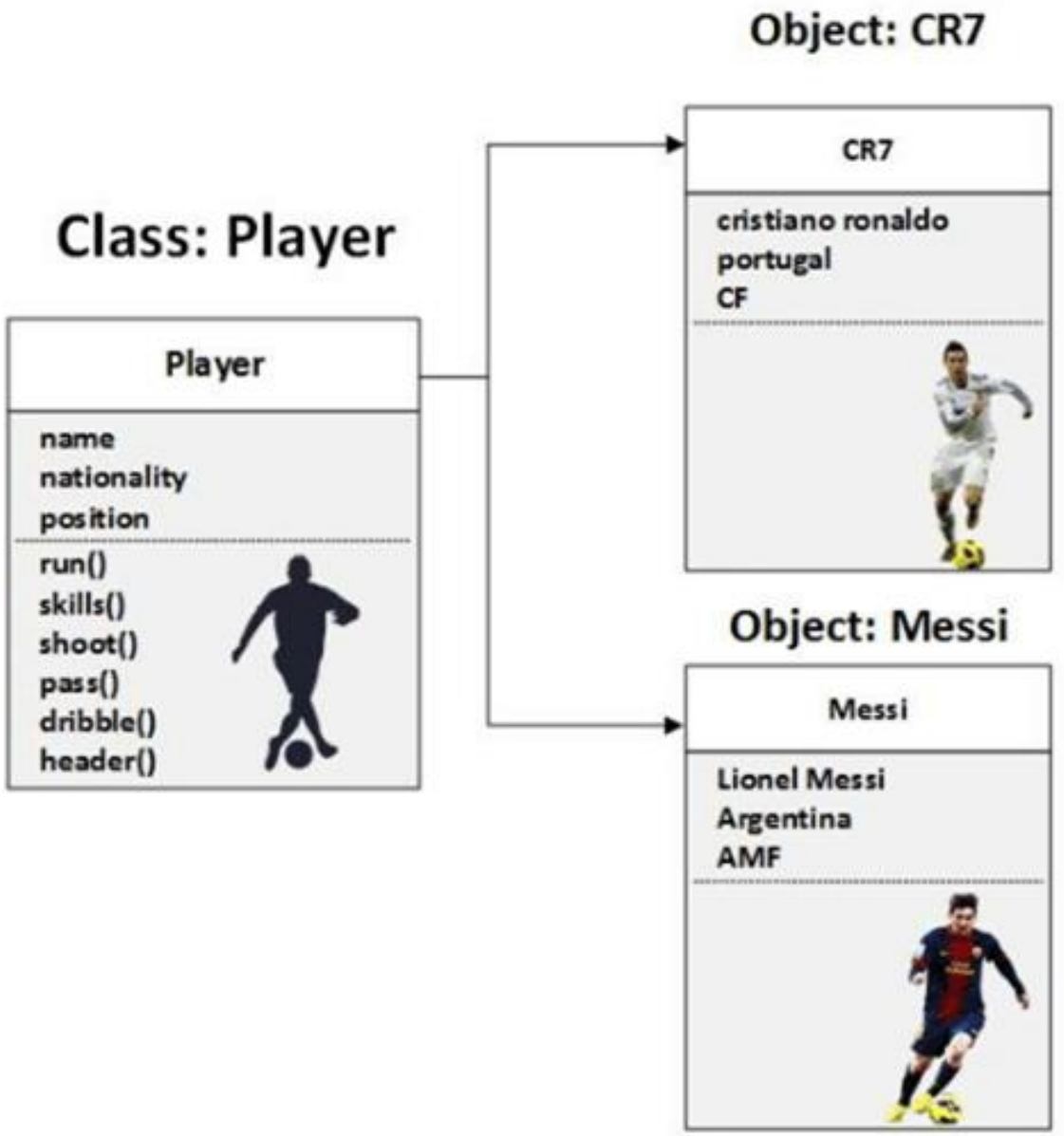
```php
class User
{
    public $login;
    public $password;
    public function __construct($log, $pwd) {
        $this->login= $log;
        $this->password = $pwd;
    }
    public function login() {
        echo "user connected". $this->login;
    }

}
$u1= new User ("admin","admin");
$u1->login();
```

Définition du constructeur

Appel du constructeur

POO – Exercice

- Ecrire le code de la classe Player
- Créer deux objets à partir de la classe Player, soit CR7 et Messi respectivement en renseignant les valeurs des propriétés



POO – Les attributs d'une classe

- Les attributs sont déclarés en utilisant la syntaxe des variables et un des mots-clés suivant : **public**, **private** ou **protected**.
- Les attributs d'une classe (ainsi que les méthodes) peuvent être déclarés en indiquant en début de classe leur visibilité appelée également **portée.**
- Ainsi, un attribut ou méthode peuvent être:
  - public
  - protected
  - private

POO – Accès public d'un attribut ou méthode

- ***Accès public***:
  - Une méthode (ou attribut) publique est accessible depuis toute votre application aussi bien dans la classe concernée que dans tout le script, c'est le cas par défaut.
  - On peut cependant le dire explicitement en utilisant le mot-clé « public »

```php
<?php
class voiture
{
    var $vitesse;
    function avancer ()
    {
    echo "J'avance";
    }
}
//////////////////Ou
class voiture{
    public $vitesse;
    public function avancer ()
    {
    echo "J'avance";
    }
}
?>
```

POO – Accès privé d'un attribut ou méthode

- ***Accès private***:
  - Une méthode (ou attribut) en accès privé n'est utilisable qu'à l'intérieur même de la classe
  - Il est interdit d'y faire appel à partir de l'extérieur
  - Il faut utiliser le mot clé « private »

```php
<?php
class voiture
{

    private $vitesse=100;
    public function avancer ()
    {
    echo "J'avance avec la vitesse".$this->vitesse;
    }
}
$ma_voiture = new voiture();
$ma_voiture->avancer();
$ma_voiture->$vitesse=120;
echo $ma_voiture->$vitesse;
?>
```

Notice: Undefined variable: vitesse in C:\wamp64\www\POO\voiture.php on line *36*

Call Stack

#	Time	Memory	Function	Location
1	0.0017	412608	{main}( )	...\voiture.php:0

Ces deux lignes vont générer une erreur lors de l'exécution, parce que nous essayons d'accéder à des propriétés privées de la classe depuis l'extérieur

POO – Accès privé d'un attribut ou méthode

- ***Accès protected***:

  - C'est une solution intermédiaire entre accès privé et public.
  - Il est interdit d'y faire appel à partir de l'extérieur, mais ils sont utilisables par les classes dérivées.
  - Il faut utiliser le mot clé « protected ».

```php
<?php
class vehicule_a_moteur
{
    protected $vitesse=30;
    private $roues = 4;
}
class voiture extends  vehicule_a_moteur
{
    public function avancer(){
        echo "J'avanceà la vitesse de ";
        echo $this->vitesse ."km/h";
                                }
    public function nombreRoues(){
        echo "Il y a "  .$this->roues. "</br>" ;
                                }
}
$ma_voiture = new voiture();
$ma_voiture->avancer();
$ma_voiture->nombreRoues();
?>
```

Appel interdit, le nombre de roues est privé. L'appel à la fonction génère une erreur

Exercice

- Créer une classe **Cercle** ayant les attributs suivants : **''xCentre'', ''yCentre'', ''diametre'', ''couleur'' et ''estVisible''**
- Déclarer le constructeur par défaut de la classe qui permettra de créer un cercle avec comme origine les coordonnées **(0,0)**, un diamètre de **5**, de couleur **rouge** et **visible**.
- Créer une méthode qui permet de savoir si un point est dans un cercle. Pour résoudre ceci, il suffit de calculer la distance du point de coordonnées (x, y) au centre C du cercle. Si cette distance est supérieure au rayon, alors vous êtes dehors, sinon, vous êtes dedans. Le calcul de la distance Euclidienne dans un plan se calcule simplement : sqrt((x-C_x)**2 + (y-C_y)**2)

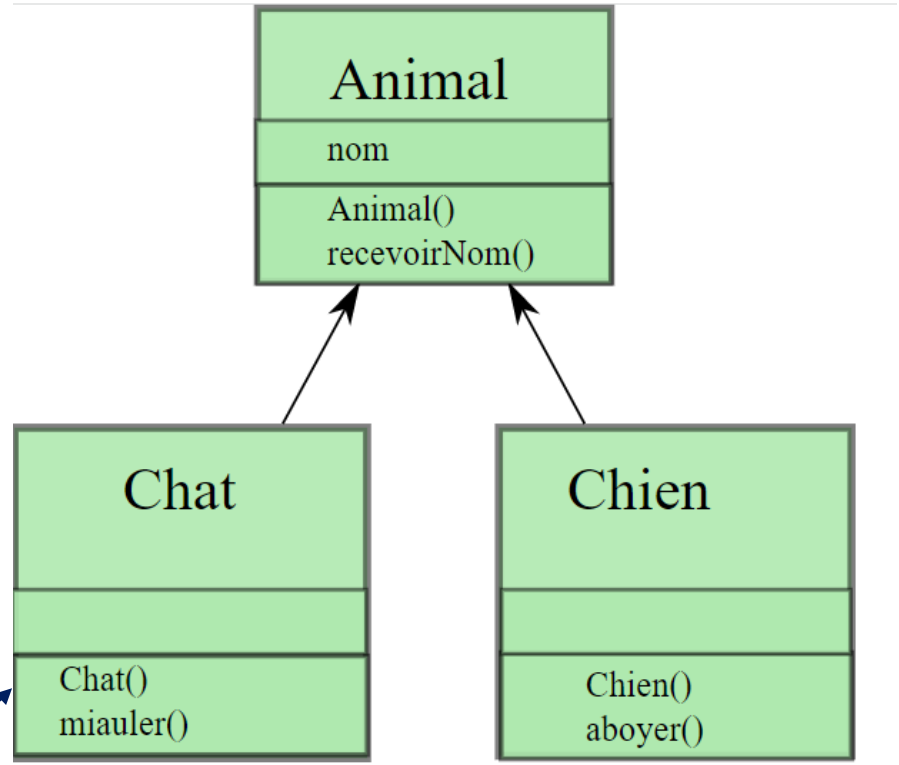
Héritage

- Souvent, plusieurs classes partagent les mêmes attributs et méthodes.
- On parle de catégories et de sous catégories: on explique cette relation par l'héritage.
- **Exemple:**
  - Ma voiture est créée en utilisant la classe voiture
  - La classe voiture **dérive (hérite)** de la classe des véhicules à moteur
  - c à d qu'elle peut en utiliser les attributs et les méthodes.
  - On peut ajouter une autre classe appelée camion qui **hérite** aussi des véhicules à moteur.

Héritage

- La classe dérivée hérite des caractéristiques (propriétés et méthodes) de la classe parente, et vous lui ajoutez des fonctionnalités supplémentaires.
- Syntaxe :

class nouvelle_classe **extends** super_classe



La nouvelle classe hérite des attributs et des méthodes appartenant à la super-classe tout en définissant ses propres fonctions et variables.

Héritage- exemple

Commençons par définir la super classe (appelée également classe mère)

```php
class Person
{
    public $name;
    public $age;
    public function __construct($name, $age) {
        $this->name = $name;
        $this->age = $age;
    }
    public function Bonjour() {
        echo "Bonjour, je m'appelle " . $this->name .
             ", j'ai " . $this->age . " ans.";
    }
}
```

Constructeur

Héritage- exemple

```php
class Student extends Person
{

    public $course;
    public function __construct($name, $age, $course) {
        parent::__construct($name, $age);
        $this->course = $course;
    }
    public function Bonjour() {
        echo "Bonjour, je m'appelle " . $this->name .
            ", j'ai " . $this->age . " ans, et j'étudie". $this->course;
    }
}
$std = new Student('Manal',20, 'Informatique');
$std->Bonjour();
```

Le mot clé extends veut dire que la classe Student hérite de la classe Person

Appel du constructeur de la super classe en lui passant en paramètre le nom et l'age

Redéfinition de la méthode Bonjour

En PHP, on dit qu'on « surcharge » une propriété ou une méthode d'une classe mère lorsqu'on la redéfinit dans une classe fille.

L'opérateur ::

Appelé également opérateur de **résolution de portée**, le symbole "**double deux-points**" (::), fournit un moyen d'accéder aux membres ***static*** ou ***constant,*** ainsi qu'aux propriétés ou méthodes surchargées d'une classe.

```php
class Produit
{
  public static $tva=0.2;
  private $nom;
  public function __construct($nom)
  {
      $this->nom= $nom;
  }
  public static function prix_ttc($prix)
  {
    return $prix-($prix*0.2);
  }
}
```

```php
$p1= new Produit("Ordinateur");
echo $p1::$tva;
echo Produit::prix_ttc(9000);

echo Pr->prix_ttc(9000); #Erreur
```

Accéder à une propriété statique d'une classe

Accéder à une méthode statique d'une classe

L'opérateur parent

Fait référence à des variables ou des fonctions présentes dans la super-classe à partir d'une autre classe héritant de cette dernière

Utilisation de l'opérateur :: pour accéder à une méthode surchargée d'une classe

```php
<?php
class vehicule_a_moteur
{
    function avancer (){echo "J'avance <br/>"; }
}
class voiture extends vehicule_a_moteur {
    function passer_la_vitesse ($vitesse)
    {
        echo "Je passe la vitesse $vitesse <br/>";
    }
function avancer ()
{
    $this->passer_la_vitesse(1);
    parent::avancer();
}
}
$maVoiture = new voiture();
$maVoiture->avancer();
//Affiche "Je passe la vitesse 1" puis "J'avance"
?>
```

Redéfinition d'attributs ou de méthodes

- Si un attribut est redéfini, c'est la dernière définition qui est utilisée pour déterminer la valeur par défaut de l'objet de la classe fille.

```php
<?php
class vehicule_a_moteur
{
Protected $nombre_roues = 2;
}

class voiture extends vehicule_a_moteur
{
Private $nombre_roues = 4;
Public function getRoues(){
    return $this->nombre_roues;
                              }
}
$voiture = new voiture();
echo $voituree->getRoues(); //affiche 4
?>
```

Accède à l'attribut nombre de roues de la classe dérivée (la classe fille)

Redéfinition d'attributs ou de méthodes

- Si une méthode est redéfinie, c'est la dernière définition qui est utilisée

```php
<?php
class vehicule_a_moteur
{
    function nombre_de_roues (){echo "2";}
}
class voiture extends vehicule_a_moteur
{
    function nombre_de_roues (){echo "4";}
}
$voiture = new voiture();
echo $voituree->nombre_de_roues(); //affiche 4
?>
```

La priorité est accordée à la méthode redéfinie

- La méthode redéfini doit avoir le même nombre d'arguments sur les deux classes.