Programmation Orientée Objet (POO)

Christel Ehrhart
contact@ce-formation.com

Définitions

Encapsulation, Attributs et Méthodes

Utilisation des classes et des objets

Les methodes magiques

Héritage

Chainage des methodes

Autres concepts

Qu'avez-vous retenu ?

Des question ?

Cours suivant...

A suivre :

- Mise en place de la POO en PHP

DÉFINITIONS

Qu’est ce que la POO ?

Programmation Orientée Objet

- Représentation sous forme d’objets

- Un objet contient
  - Des caractéristiques (= variables) : attributs
  - Des capacités (= fonctions) : méthodes

- La programmation orientée objet repose sur 3 concepts clés:
  - l'encapsulation
  - l'héritage
  - le polymorphisme

Avantages et inconvénients

Avantages

- Possibilité de réutiliser le code dans différents projets
- Conception de l'algorithme plus claire et organisée
- Code modulaire

Inconvénients

- Mal conceptualisée elle sera difficilement maintenable et modulaire
- Nécessite généralement plus de ressources et de temps d'exécution

la programmation par objet n'est pas synonyme de bonne programmation

Qu'est-ce qu'un objet ?

- Un **objet** est une représentation d'une chose à laquelle on associe des propriétés et des actions.
- Les **attributs** (aussi appelés **données membres**) sont les propriétés propres à un objet.
- Les **méthodes** sont les actions applicables à un objet.

Qu'est-ce-qu'une classe ?

- Une **classe** est un modèle de données définissant la structure commune à tous les objets qui seront créés à partir d'elle.

Un moule qui permet de créer autant d'objets de même type et de même structure qu'on le désire.

Qu'est-ce-qu'une instance ?

- Lorsque l'on crée un objet, on réalise ce que l'on appelle une **instance de la classe**.

  C'est à dire que du moule, on en extrait un nouvel objet qui dispose de ses attributs et de ses méthodes.

- L'objet ainsi créé aura pour type le nom de la classe.

Remarque :

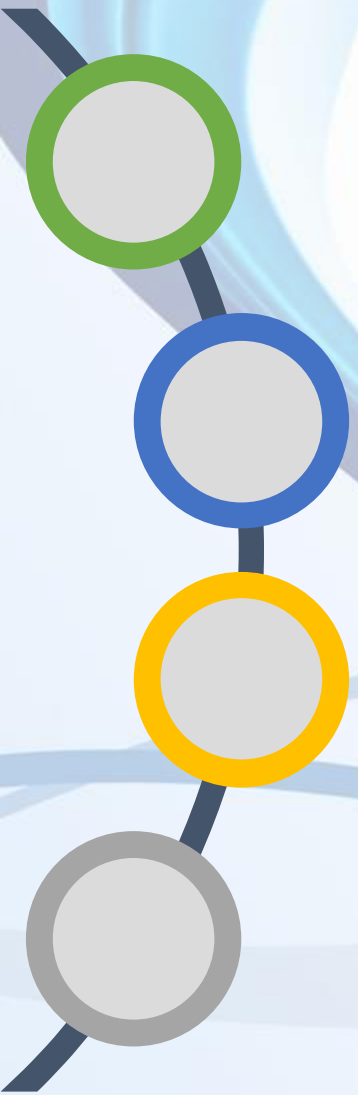
une classe n'est pas un objet.

C'est un abus de langage de dire qu'une classe et un objet sont identiques.

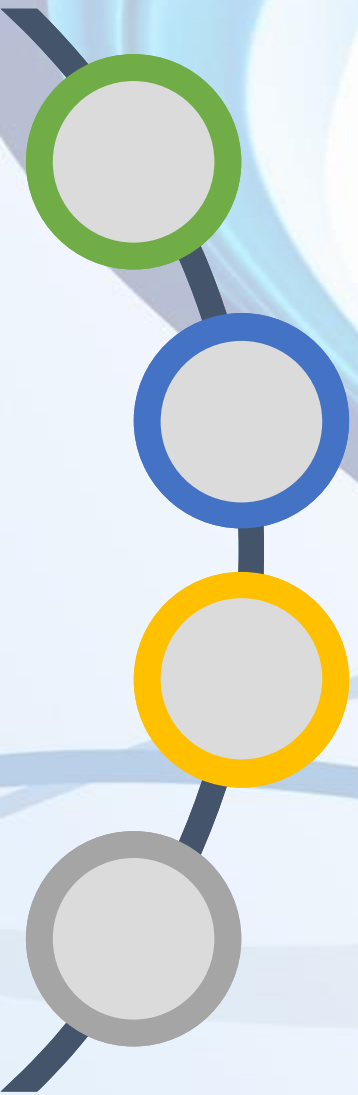
Déclarer et accéder à une classe

- Syntaxe de déclaration d'une classe

```
<?php
        class NomDeMaClasse  {
                // Attributs

                // Constantes

                // Méthodes
        }

$objNom = new NomDeMaClasse;
```



Encapsulation, Attributs et Méthodes



ENCAPSULATION

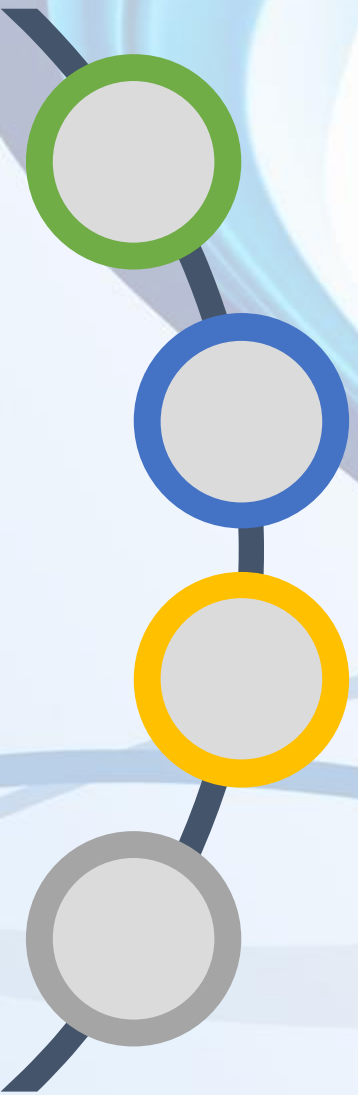
Principe et importance

- L'encapsulation correspond à la façon dont on définit la visibilité et l’accessibilité de nos propriétés et méthodes de classe

- Lorsque l’on crée une nouvelle classe, il faut toujours se demander quelles propriétés et méthodes vont devoir ou pouvoir être accessibles depuis l’extérieur de la classe

Visibilité des attributs

- Public (par défaut) : accessibles depuis n'importe où dans le programme principal
- Private :
  - propriété ou méthode n’est accessible qu’à l’intérieur de la classe en soi.
  - la propriété ou méthode ne sera accessible ni depuis une classe fille, ni depuis une instance de la classe.
- Protected : restreint l’accès à une méthode ou propriété à la classe en soi ainsi qu’aux classes filles étendant cette classe.
- Convention de nommage :
  - les éléments private et protected sont précédés d’un _ afin d’augmenter la lisibilité du code



Attributs

Attributs

- Attribut = propriété nommée d'une classe
- Syntaxe
  - visibilité nom = valeur initiale;
- Construire les noms de variables avec leur type :
  $strCustName, $intNb, $floatAmount, $arrList
- Nom de variables en anglais
- Commencer par une minuscule, puis chaque mot commence par une majuscule
- Les variables privées sont précédées d'un _

Exemples :

```
public $strName = "Dupont";

private $_intAge;
```

Déclarer les attributs

```php
<?php
// Déclaration de la classe Personne
class Personne  {
        // Attribut public Nom par défaut Dupont
        public $strName         = "Dupont";
        // Attribut public Prénom par défaut Jean
        public $strFirstName    = "Jean";
        // Attribut privé taille par défaut 1,60
        private $_intHeight     = 160;
}
```

Accéder aux attributs

```php
// Instanciation de la classe Personne pour créer l'objet objPersonne
$objPersonne = new Personne;

// Affichage de l'attribut Nom
echo($objPersonne->strName);

// Affichage de l'attribut taille
echo($objPersonne->_intHeight);
```

Constantes

- Une constante est une sorte d'attribut appartenant à la classe dont la valeur ne change jamais
- Faire précéder son nom du mot-clé const
- Syntaxe
  - pas de $ devant
  - nom en MAJUSCULE

- L'accès aux constantes se fait à par :: au lieu de ->

⇒Attention : 2 façons sont disponibles en PHP pour définir une constante :

- define('MAX_VALUE', '1.0'); // Fonctionne uniquement en dehors d'une classe
- const MAX_VALUE = 1.0; //Fonctionne à la fois dans une classe et en dehors

Déclarer les constantes

```
class Personne  {
        // Constantes
         const NOMBRE_DE_BRAS = 2;
         const NOMBRE_DE_JAMBES = 2;
         const NOMBRE_DE_YEUX = 2;
         const NOMBRE_DE_PIEDS = 2;
         const NOMBRE_DE_MAINS = 2;
}
$objPersonne = new Personne;
echo "Chaque personne a ". $objPersonne:: NOMBRE_DE_YEUX."yeux";
```

MÉTHODES

Méthodes

- Une méthode est une fonction appartenant à la classe
- Syntaxe
  - visibilité function nomDeLaFonction() { … }
- Exemple:

```
class Personne  {
    public function parler() {
                    echo 'Bonjour tout le monde';
    }
}
$objPersonne = new Personne;
$objPersonne->parler();
```

Attributs et méthodes

- Par convention les attributs doivent être privées
- Raison : il est "dangereux d'accéder directement à un attribut d'une classe
- Comment y accéder ? Par les méthodes qui sont quant à elles publiques

Setters / Getters

- Les fonctions définies à l'intérieur d'une classe sont appelées des méthodes.
- Les méthodes qui servent à définir / modifier / mettre à jour une valeur sont appelées des setters (setName)
- Les méthodes qui servent à récupérer des valeurs sont appelées des getters (getName)

Mot clé $this

- Ce mot clef est appelé pseudo-variable
- Il sert à faire référence à l'objet couramment utilisé

Attention :

il s'agit d'une variable donc ne pas oublier le $ devant

Le $ étant devant le this, il n'existe plus après la ->

Setters / Getters - exemple

```
class Personne  {
        private $_strName;

        public function getName(){
                return $this->_strName;
        }

        public function setName($strNewName){
                $this->_strName = $strNewName;
        }


}
```

UTILISATION DES CLASSES ET DES OBJETS

Appeler une classe

- Inclure **le fichier** correspondant

```
<?php
        include("maclasse.php");
```

Instanciation d'une classe

- On utilise le mot-clé **new** suivi du nom de la classe

- Création d'objets de type Personne :

```
<?php
    $objPersonne1 = new Personne();
    $objPersonne2 = new Personne();
```

- Un objet est une variable dont le type est celui de la classe qui est instanciée

- L'instanciation d'une classe fait appel au constructeur de la classe

Accès aux attributs - Accès en écriture

```php
<?php
    $objPersonne1 = new Personne();
    // Définition des attributs de la personne 1
    $objPersonne1->strName = 'Hamon';
    $objPersonne1->strFirstName = 'Hugo';
    $objPersonne1->dateBirthDate = '02-07-1987';
    $objPersonne1->intHeight = '180';
    $objPersonne1->strSex = 'H';
```

```php
$objPersonne2 = new Personne();
// Définition des attributs de la personne 2
$objPersonne2->strName = 'Dubois';
$objPersonne2->strFirstName = 'Michelle';
$objPersonne2->dateBirthDate = '18-11-1968';
$objPersonne2->intHeight = '166';
$objPersonne2->strSex = 'F';
```

Remarque :

seules les **variables publiques** sont accessibles directement

Accès aux attributs - Accès en lecture

- Affichage du nom et du prénom de la personne 1

```
<?php
        echo "Personne 1 :";
        echo "Nom : ". $objPersonne1->strName;
        echo "Prénom : ". $objPersonne1->strFirstName;
```

Personne 1 :

Nom : Hamon

Prénom : Hugo

Remarque :

seules les **variables publiques** sont accessibles directement

Accès aux constantes

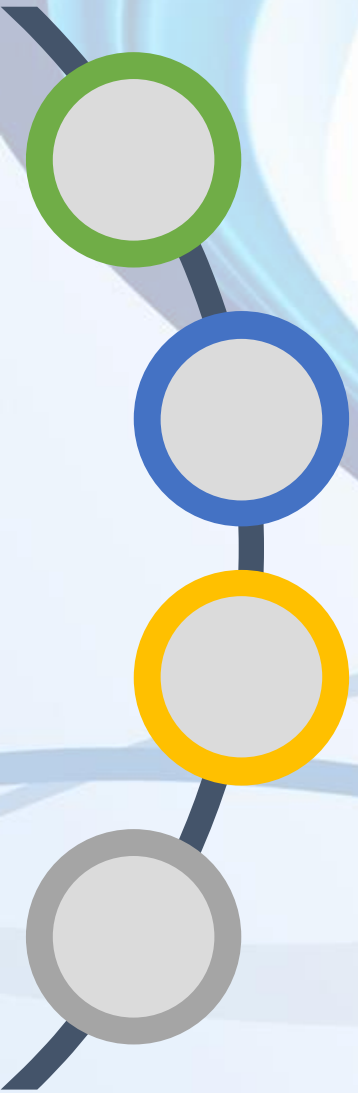
```
<?php
    echo "Chaque personne a ".$objPersonne1::NOMBRE_DE_BRAS." bras.";
?>
```

Remarque :
si l'on tente de redéfinir la valeur d'une constante, PHP génèrera une erreur de type

Parse error: syntax error, unexpected '=' in ….

Accès aux méthodes

```
<?php
    $objPersonne1->setName("Pierre");

    echo ($objPersonne1->getName());


     $objPersonne1->boire();
     $objPersonne2->manger();


?>
```



LES METHODES MAGIQUES

Définition

- Les méthodes magiques sont des méthodes qui vont être appelées automatiquement dans le cas d'un évènement particulier.

__construct()	__destruct()
__call()	__callStatic()
__get()	__set()
__isset()	__unset()
__toString()	__clone()
__sleep()	__wakeup()
__invoke()	__set_state()
__debugInfo()	

Constructeur - Définition

- Méthode qui permet de rendre la classe immédiatement opérationnelle et utilisable en définissant des propriétés dès qu'un utilisateur va créer une nouvelle instance de cette classe

- N'est pas obligatoire mais fortement recommandé

Constructeur - Fonctionnement

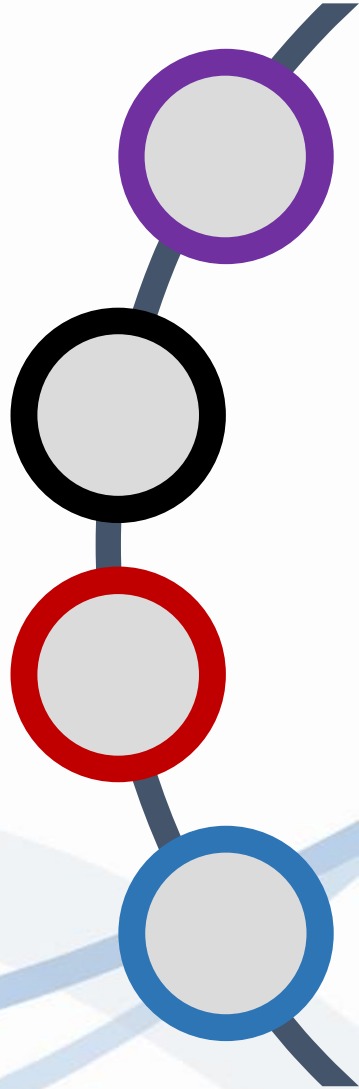
- On déclare un constructeur de classe avec la méthode __construct() (double underscore en début de nom)

```
class Personne  {
        protected $_strName;
        protected $_dateCreate;
        public function __construct($strNewName) {
                $this->_strName = $strNewName;
         }

         public function getName() {
                return $this->_strName;
        }
}
```

```
<?php
        $objPersonne = new Personne("Jean");

        echo $objPersonne->getName();

?>
```

⇒ Jean sera affiché

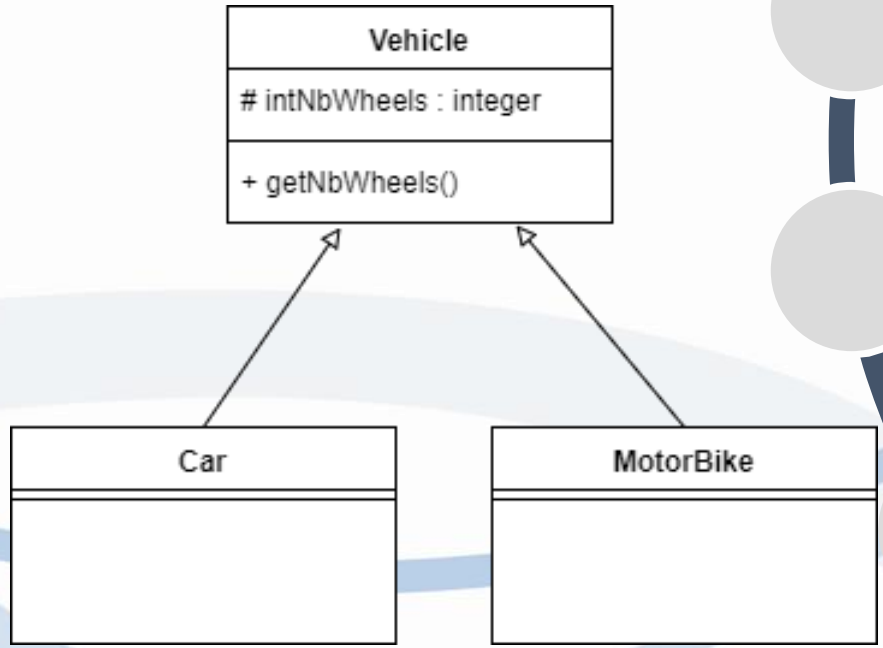
HÉRITAGE



Quel intérêt ?

- Il s'agit d'un moyen simple de favoriser la réutilisabilité du code
- L'héritage va permettre une réutilisation verticale du code

c'est à dire que les enfants (les classes filles)
pourront réutiliser les comportements et
les données de leur(s) parent(s)
(les classes mères)

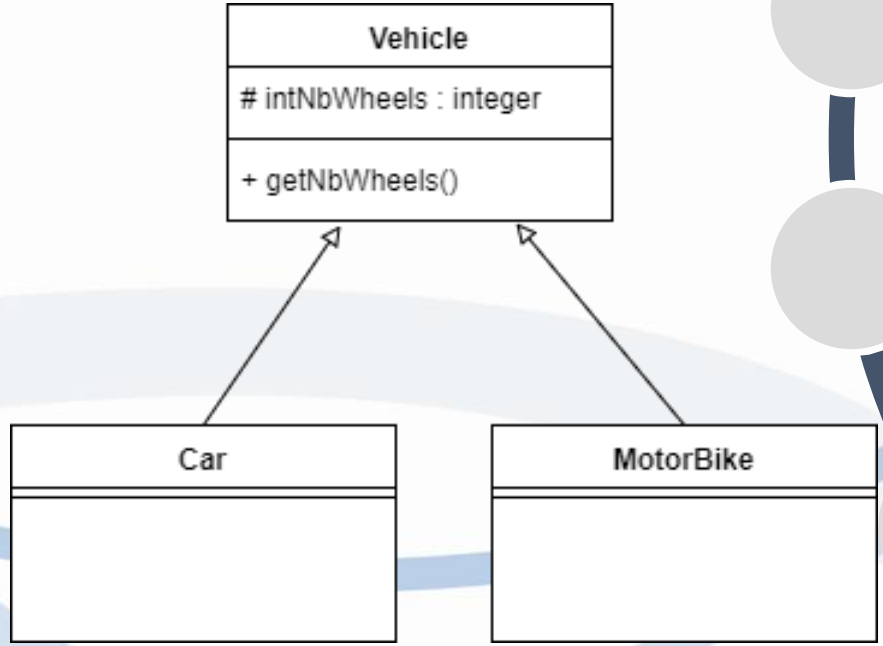


Fonctionnement

- L'héritage entre deux classes caractérise la relation "est un espèce de..."

Pour que la classe "Car" hérite de "Vehicle"

on utilise le mot clé *extends*



Opérateur de résolution de portée

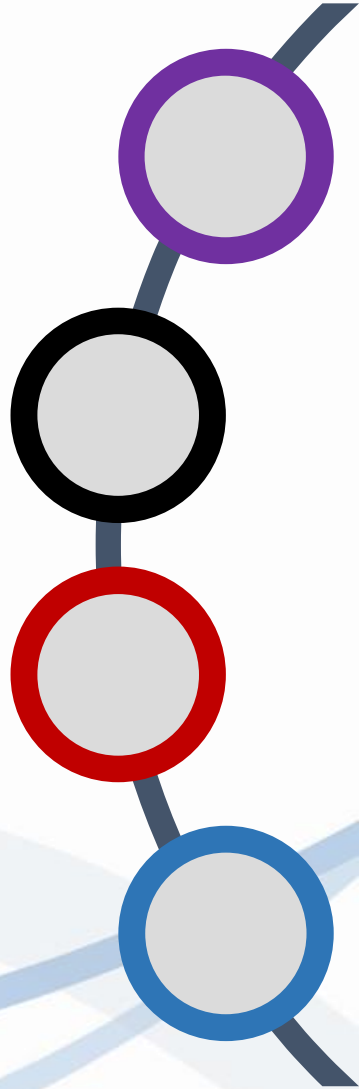
- L’opérateur de résolution de portée est symbolisé par le signe ::
- Il sert à accéder à des méthodes et des propriétés surchargées (c’est-à-dire modifiées par une classe fille)

Héritage

- Création d'une classe Vehicle qui contient le nombre de roues (définit dans le constructeur) et affiche le nombre de roues
- Création d'une classe Car qui est fille de Vehicle, avec 4 roues
- Création d'une classe Motorbike qui est fille de Vehicle, avec 2 roues
- Création d'un objet voiture, afficher le nombre de roues
- Création d'un objet bike, afficher le nombre de roues

⇒heritage.php

CHAINAGE DES METHODES



Principe et intérêt

- Chainer des méthodes permet d'exécuter plusieurs méthodes d'affilée de façon simple et plus rapide

- Il suffit d'utiliser l'opérateur d'objet pour chainer différentes méthodes

```
$objet->methode1()->methode2()
```

- Pour pouvoir chainer des méthodes il faut que celles-ci retournent l'objet

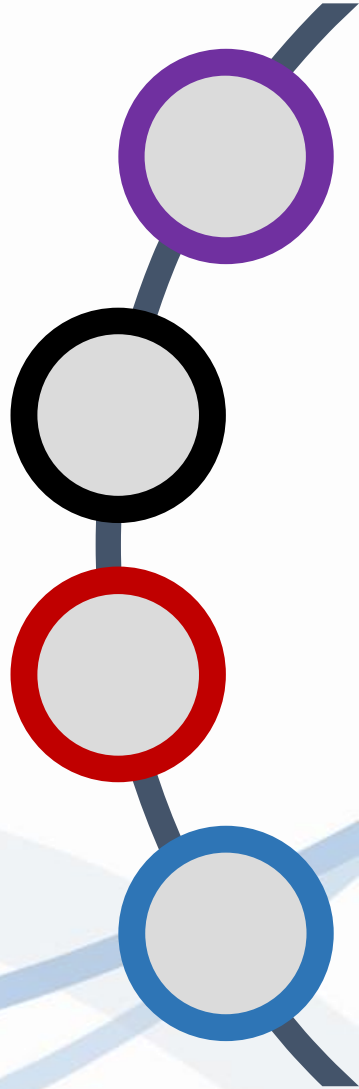
Chainage des méthodes

- Créer une classe Character qui contient le nom et la position du personnage
- Ajouter les méthodes pour faire avancer ou reculer le personnage

- Dans un autre fichier, faire bouger un personnage en utilisant le chainage des méthodes

⇒chainage.php et chainage_index.php

AUTRES CONCEPTS

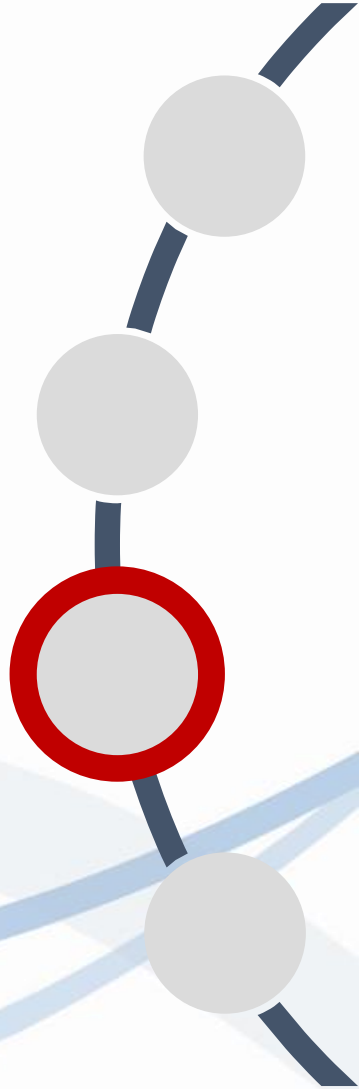


Filiation, parents, enfants et arbre généalogique

- Une classe peut être considérée comme un type au même sens qu'une chaine ou un entier

- Hiérarchie de types : Quand on réalise un héritage, la classe fille caractérise un nouveau type mais est toujours du type de la mère (une pomme est toujours considérée un fruit, qui est toujours considéré comme un végétal)

- instanceOf permet de savoir si une instance est d'un type donné



Hierarchie

Créer une classe Vivant

Créer une classe Vegetal fille de Vivant

Créer une classe Fruit fille de Vegetal

Créer une classe Apple fille de Fruit

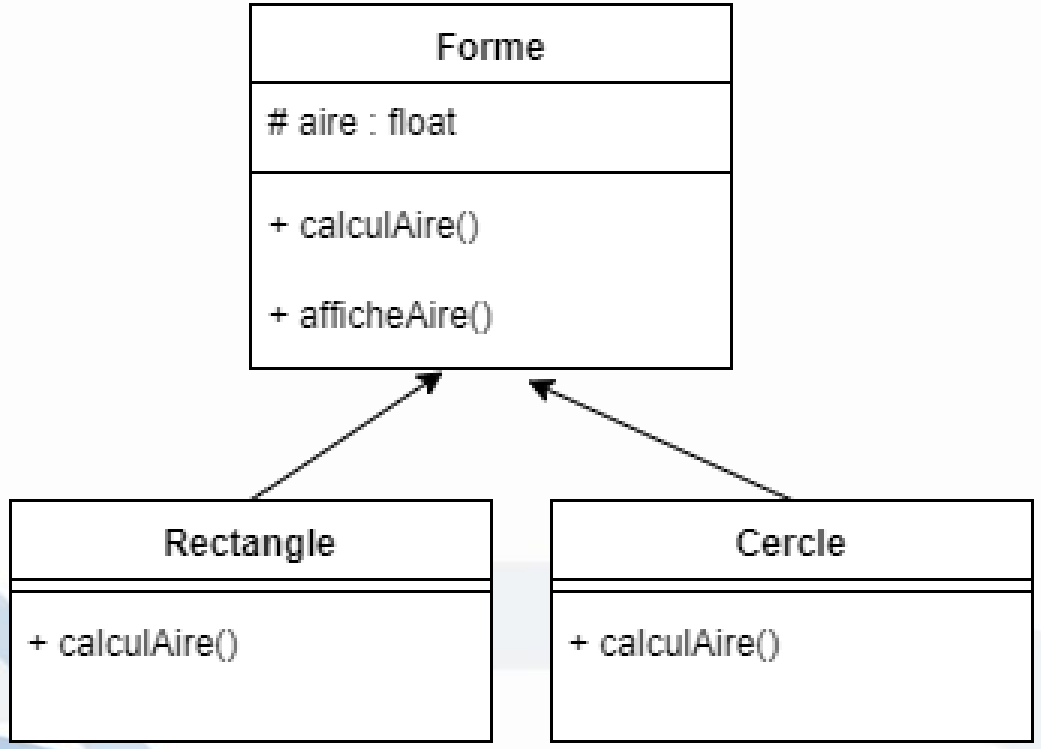
Créer une classe GoldenLady fille de Apple

Créer un objet, instance de la classe GoldenLady et afficher à l'aide de instanceOf, si l'objet est de type Apple, Fruit et Végetal

=> Voir le fichier hierarchie.php

Le polymorphisme

- Mécanisme par lequel une méthode peut être surchargée
- Une méthode peut ainsi être déclarée dans la classe parente, et redéfinie dans les classes filles



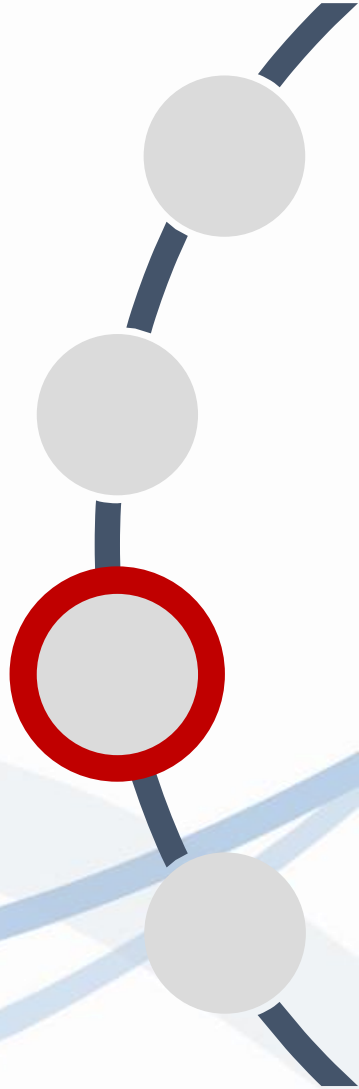
Abstraction

- Il est possible de déclarer des méthodes dont la définition n'est pas encore connue dans la classe mère mais le seront dans une classe fille

  On parle alors de méthodes abstraites

- Une classe qui contient au moins une méthode abstraite doit elle aussi être déclarée abstraite et ne peut plus être instanciée

  On parle alors de classe abstraite



Polymorphisme - abstraction

- Créer une classe Form contenant la valeur de l'aire, le calcul de l'aire et l'affichage de l'aire
- Créer les classes filles rectangle et cercle, qui calculent l'aire
- Créer un objet rectangle (10 x 5) et un objet Cercle (rayon = 5), et afficher leur aire

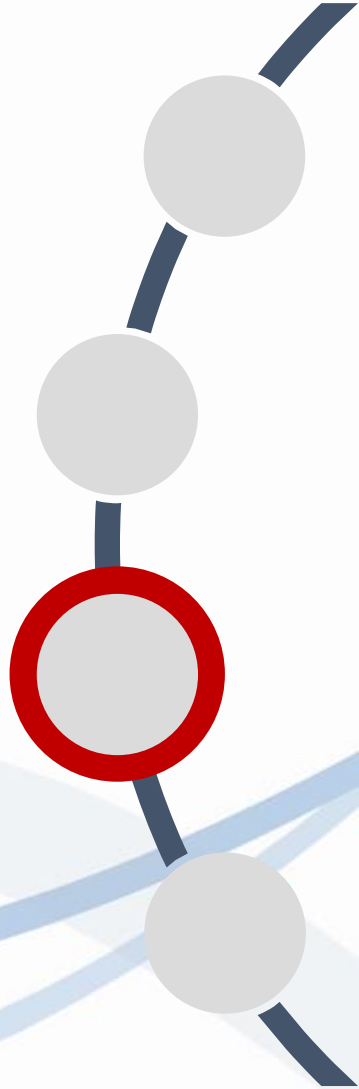
⇒Voir le fichier polymorphisme.php

Interfaces

- Si toutes les méthodes d'une classe sont abstraites, cette classe est alors une interface

  On utilise le mot clé interface en lieu et place du mot clé class

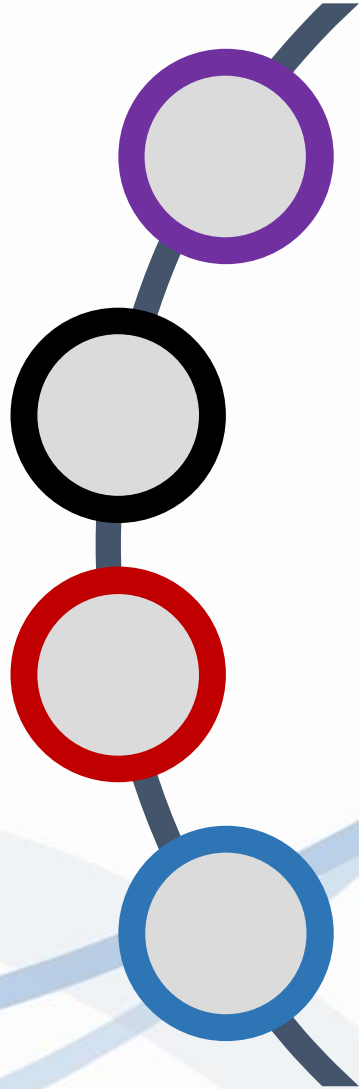
- L'interface donne une "forme" a notre classe: elle va définir comment celle-ci doit se présenter (méthodes) mais pas comment elle doit se comporter (pas de contenu de méthodes)



Interfaces

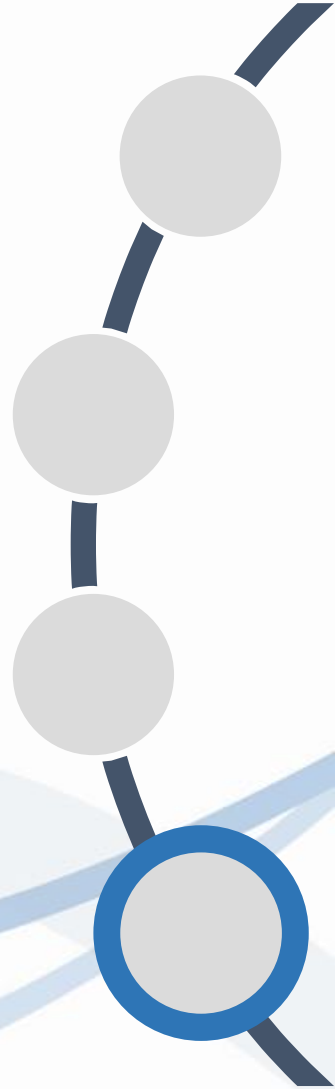
⇒Voir le fichier interface.php

QU'AVEZ-VOUS RETENU ?



Qu'avez-vous retenu ?

1. Qu'est-ce qu'une classe ?
2. Comment appelle-t-on une variable définie à l'intérieur d'une classe ?
3. Dans un contexte objet, quelle est l'utilité de la pseudo-variable $this ?
4. Lorsqu'on étend une classe, de quelles propriétés et méthodes la classe fille hérite-t-elle ?
5. Comment étendre une classe en pratique ?
6. Qu'est-ce que l'encapsulation ? Quel est l'intérêt de ce concept ?
7. Qu'est-ce qu'un constructeur ?
8. Que signifie "surcharger" une propriété ou une méthode ?
9. Quel opérateur permet d'accéder à des méthodes et propriétés surchargées ?



Des question ?

Cours suivant…

A suivre :

- Mise en place de la POO en PHP