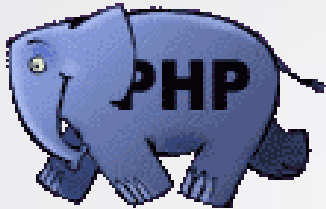




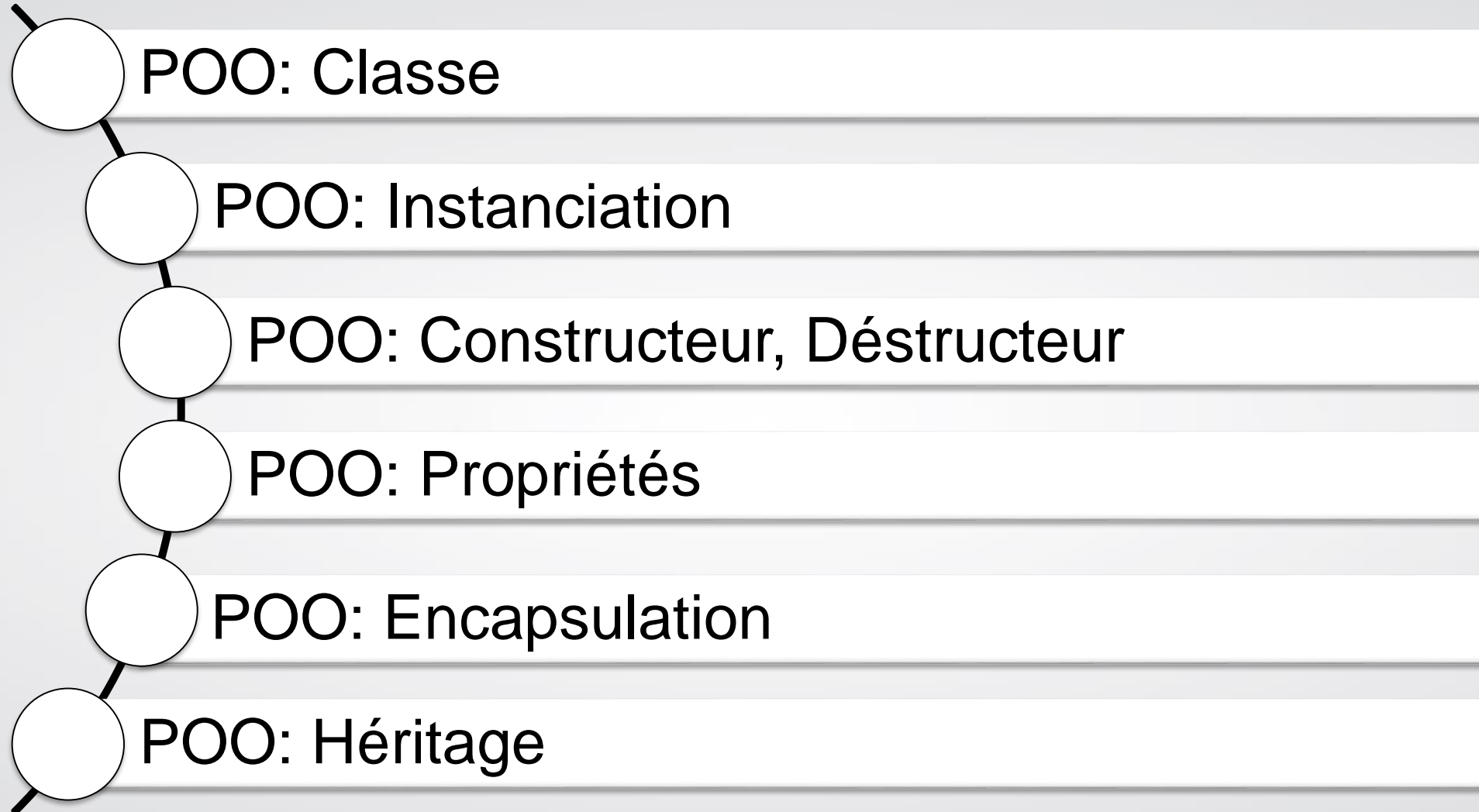
## Chapitre 4: PHP



# UP Web

AU: 2021/2022

# Plan





# Objectifs

- Les architectures du web
- Comprendre la syntaxe PHP
- Appréhender les notions de l'orientée objet
- Se connecter à une BD
- Manipuler les données d'une BD via PHP

## Prérequis

- Langage HTML



# ► POO: Notion de Classe (1/2)

- Une **classe** est une représentation abstraite d'un **objet**.
- Une classe peut être rendue concrète au moyen d'une **instance de classe**, que l'on appelle **objet**.
- Une classe s'écrit au moyen du mot "**class**" suivi du nom de la classe et d'accolades.

```
class Personne {  
    public string $nom;  
    public string $prenom;  
  
    public function saisir(string $nom, string $prenom) { ...  
    }  
  
    public function afficher(){ ...  
    }  
}
```



# POO: Notion de Classe (2/2)

Personne
+nom +prenom
+saisir(var,var) +afficher()

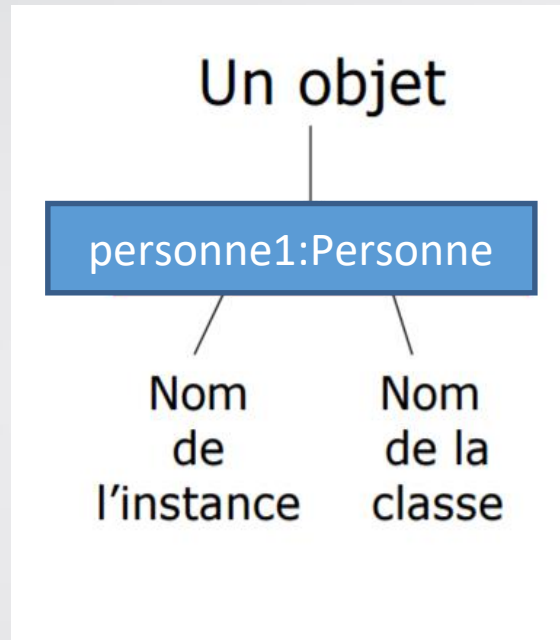
```
<?php
class Personne {
    public string $nom;
    public string $prenom;

    public function saisir(string $nom, string $prenom) {
        $this->nom = $nom;
        $this->prenom = $prenom;
    }

    public function afficher(){
        echo "Nom: $this->nom <br>";
        echo "Prenom: $this->prenom <br>";
    }
}
?>
```



# ► POO: Instanciation d'un objet



```
// Déclaration d'une instance
$personne1 = new Personne();

// Utilisation
$personne1->saisir('nom1', 'prenom1');

// Vérification
if ($personne1 instanceof Personne) {
    echo "L'objet \$personne1 est de type Personne. <br>";
}

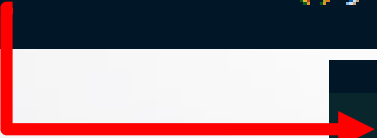
// Affichage
$personne1->afficher();
```



# ► POO: Constructeur (1/2)

- Le constructeur est la méthode qui va être appelée à l'instanciation de l'objet.
- Il doit être implémenté dans la classe elle-même de la manière suivante :

```
// Déclaration d'une instance  
$personne1 = new Personne();
```



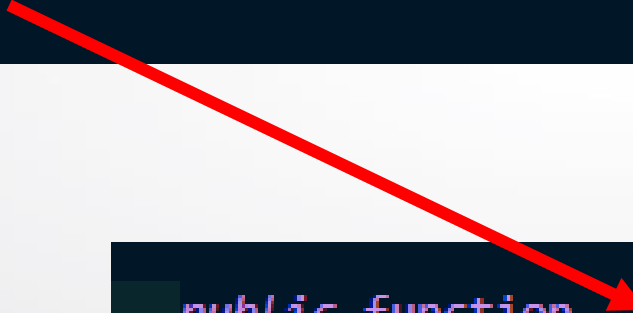
```
public function __construct () {  
    $this->nom = 'nom 1';  
    $this->prenom = 'prenom 1';  
}
```



## ► POO: Constructeur (2/2)

- Le constructeur peut aussi comporter des paramètres.
- En PHP, on ne peut avoir qu'un seul constructeur par classe.

```
// Déclaration d'une instance  
$personne1 = new Personne('nom1', 'prenom1');
```



```
public function __construct (string $nom, string $prenom) {  
    $this->nom = $nom;  
    $this->prenom = $prenom;  
}
```





# POO: Déstructeur

- Un destructeur d'une classe donnée est une méthode exécutée automatiquement à chaque fois qu'une instance de la classe donnée disparaît.
- C'est une méthode sans type de retour
- C'est une méthode sans paramètre, elle ne peut donc pas être surchargée.
- C'est une méthode en accès public.

```
public function __destruct(){  
    echo "Je suis le destructeur";  
}
```

```
// Affichage  
$personne1->afficher();  
  
unset($personne1);
```



# ► POO: Remarque (1/2)

## Constructeur

- ✓ **PHP 3 et 4** : une fonction portant le même nom que la classe
- ✓ **PHP5, PHP7 ou supérieur** : une fonction membre spécifique. PHP5 ne permet pas la **surcharge(overloading)** de fonction (ou de méthodes) et donc on ne peut attribuer le même nom à plusieurs fonctions. Par contre la redéfinition est possible.

## Destructeur

- ✓ **PHP 3 et 4** : il n'y a pas de destructeur
- ✓ **PHP 5, PHP7 et supérieur**: introduit la notion de destructeur. On utilise la fonction **unset(\$MonPoint)**.



## ► POO: Remarque (2/2)

### unset()

unset(\$nom) détruit une variable

```
//détruire une variable  
unset($x);
```

```
//détruire une case d'un tableau  
unset($t[0]);
```

```
//détruire plusieurs variables  
unset($a, $b, $c);
```



# ► POO: Propriétés d'un objet (1/2)

- Une propriété est une variable associée à un objet.
- On peut demander « le nom de cette personne » et non le nom en général.

```
$personne1 = new Personne();  
var_dump($personne1->nom);
```

→ Pas de caractère \$ devant le nom de la propriété.



## ► POO: Propriétés d'un objet (2/2)

- Une propriété est une variable associée à un objet.
- On peut demander « le nom de cette personne » et non le nom en général.

```
$personne1 = new Personne();  
var_dump($personne1->nom);
```



La propriété nom est liée intrinsèquement à l'objet.  
Cette liaison est notée par la flèche.



# POO: Droits d'accès

- Les méthodes et les variables “ **(+)public**” sont visibles et manipulables par tous les objets, même s'ils sont relatifs à d'autres classes.
- Les méthodes et les variables “ **(#)protected**” concernent les objets de la même classe ainsi que ses dérivés, mais pas ceux des classes étrangères.
- Les classes ont des variables et des méthodes internes et qui ne concernent pas l'extérieur. Ces propriétés sont déclarées en tant que “ **(-)private**”.



# POO: Encapsulation (1/3)

- L'encapsulation est la pratique consistant à regrouper des attributs au sein d'une même classe.
  - Pour améliorer la lisibilité des programmes, les attributs encapsulés sont souvent **privés** (inaccessibles aux autres classes)
  - Les données et méthodes accessibles sont dites **publiques**

# POO: Encapsulation (2/3)

```
<?php
class Personne {
    private string $nom;
    private string $prenom;
    private int $age;

    public function __construct (string $nom, string $prenom, int $age) {
        $this->nom = $nom;
        $this->prenom = $prenom;
        $this->age = $age;
    }

    public function afficher(){
        echo "Nom: $this->nom <br>";
        echo "Prenom: $this->prenom <br>";
        echo "Age: $this->age <br>";
    }

    public function setNom (string $n) {
        $this->nom = $n;
    }

    public function getNom () {
        return $this->nom;
    }
}
```

## Mutateurs (Setters)

permettent de  
modifier la valeur  
d'une propriété.





# ► POO: Encapsulation (3/3)

```
public function afficher(){  
    echo "Nom: $this->nom <br>";  
    echo "Prenom: $this->prenom <br>";  
    echo "Age: $this->age <br>";  
}
```

```
public function setNom (string $n) {  
    $this->nom = $n;  
}
```

```
public function getNom () {  
    return $this->nom;  
}
```

```
public function setPrenom (string $p) {  
    $this->prenom = $p;  
}
```

```
public function getPrenom () {  
    return $this->prenom;  
}
```

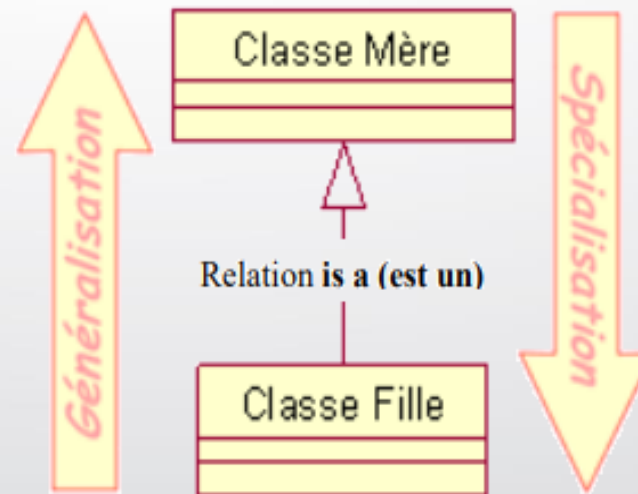
## Accesseurs (Getters)

permettent de  
renvoyer la valeur  
d'une propriété.



# POO: Héritage

- L'**héritage** consiste à définir différents niveaux d'abstraction permettant ainsi de **factoriser** certains attributs et/ou méthodes communs à plusieurs classes.
- Une classe générale définit alors un ensemble d'attributs et/ou méthodes qui sont partagés par d'autres classes, dont on dira qu'elles héritent de cette classe générale.



# POO: Redéfinition

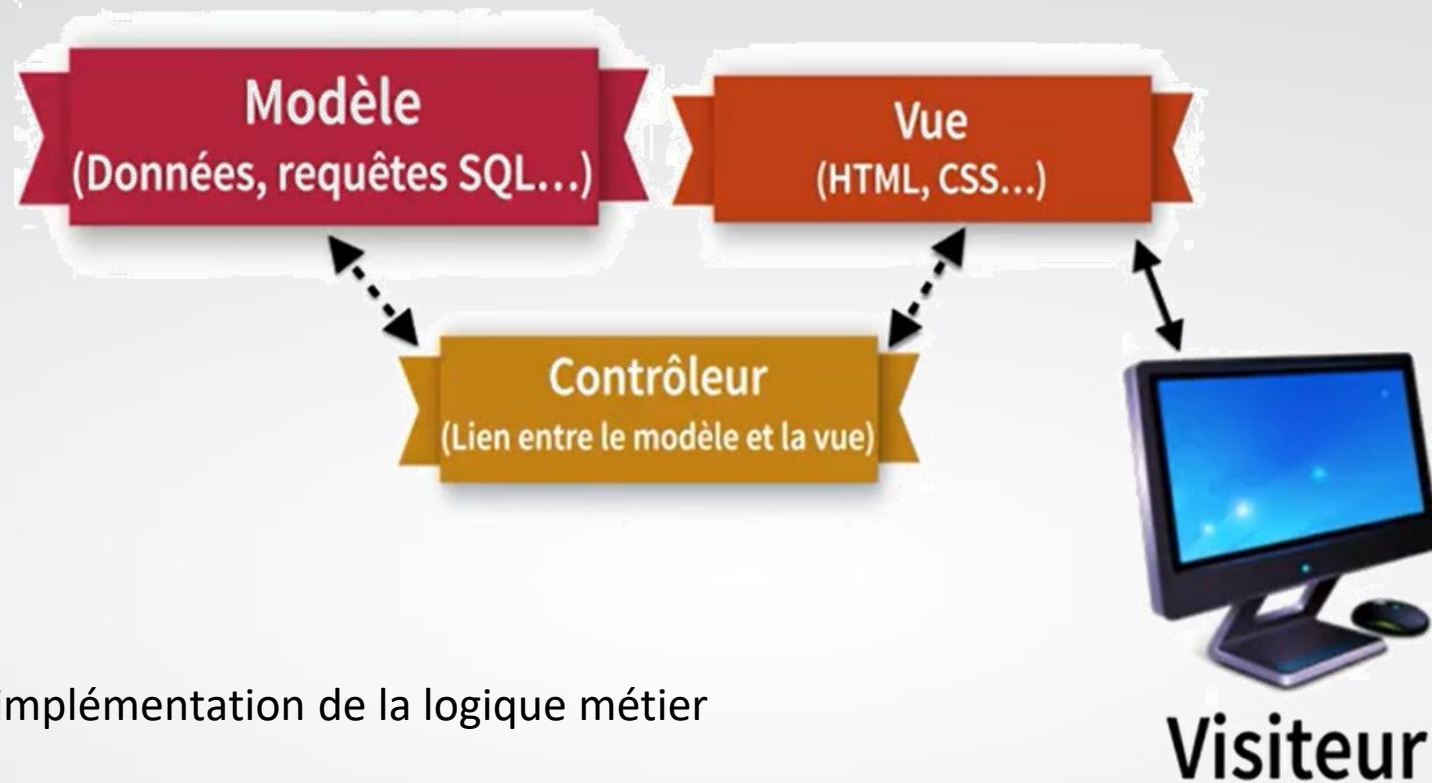
- PHP permet la redéfinition des méthodes, c'est-à-dire la possibilité de re-déclarer les mêmes attributs et opérations d'une super classe au sein d'une sous classe. Nous pouvons aussi modifier la signature de la fonction, et son traitement.

```
//Déclaration d'une instance  
$car = new voiture();  
$car->setNbRoues();  
echo $car->getNbRoues();
```

```
class vehicule {  
    private int $nbRoues = 0;  
  
    public function getNbRoues(){  
        return $this->nbRoues;  
    }  
}  
  
class voiture extends vehicule {  
  
    public function setNbRoues() {  
        $this->nbRoues = 4;  
    }  
  
    public function getNbRoues(){  
        echo "Nouvelle méthode: <br>";  
        return $this->nbRoues;  
    }  
}
```



# ► L'architecture MVC



- ✓ **Modèle**: l'implémentation de la logique métier
- ✓ **Vue**: C'est plus lié à l'affichage.
- ✓ **Contrôleur**: le lien entre le modèle et la vue



► **Merci de votre attention**