PROGRAMMATION WEB

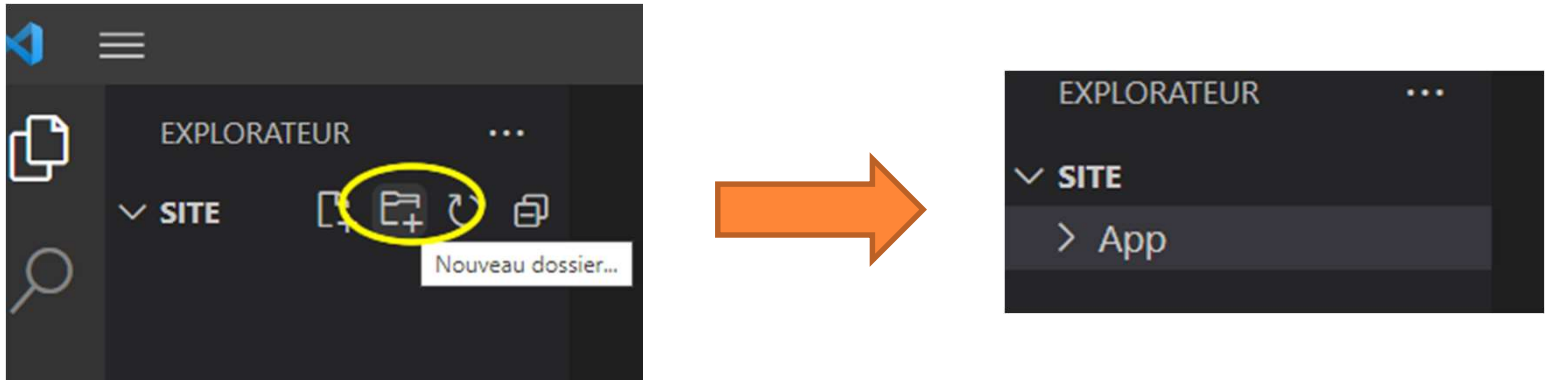
Cours 3

La programmation orientée objet en PHP

Mansour Sihem

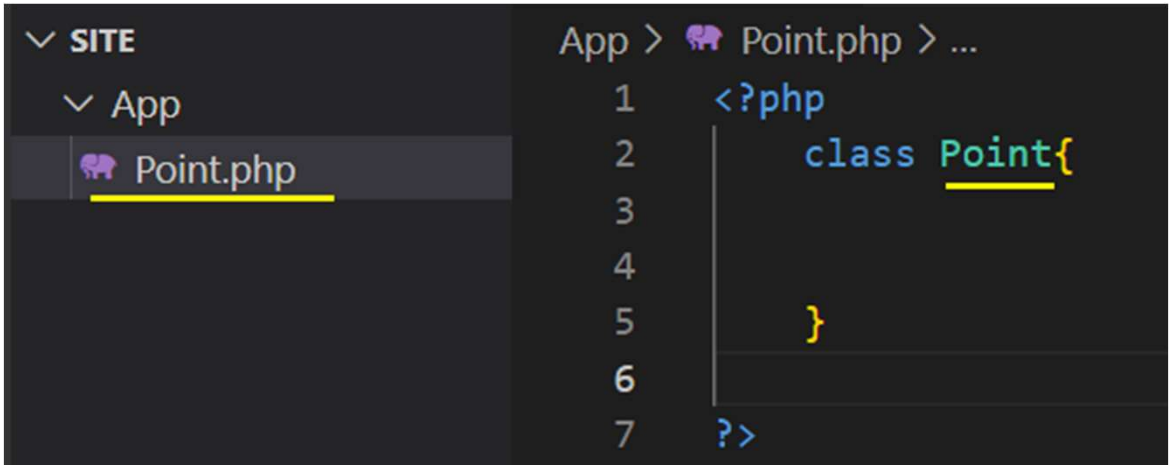
Comment écrire les classes correctement

- Pour créer des classe en php, vous devez Créer un dossier sous votre dossier du site où vous allez stocker vos classes, voir le graphique:



Comment écrire les classes correctement

- Le fichier de la classe doit porter son nom, voir l'exemple:



Les attributs d'instanciation et statiques

- Voir l'exemple ci-dessous pour savoir comment déclarer des attributs d'instanciation et statiques dans une classe

```php
<?php
    class Point{
        private $x;
        private $y;
        static $nb=0;

    }

?>
```

LES GETTERS / SETTERS EN PHP

- L'accès aux membres d'instanciation dans la classe se fait via le terme $this->
- Voir l'exemple:

```
lass Point{
    private $x;
    private $y;
    static $nb=0;
    public function setX($x){
        $this->x=$x;
    }
    public function setY($y){
        $this->y=$y;
    }
    public function getX(){
        return $this->x;
    }
    public function getY(){
        return $this->y;
    }
```

LE CONSTRUCTEUR EN PHP

- Pour définir un constructeur en PHP, utiliser le terme clé __construct, voir l'exemple ci-dessous:

```php
class Point{
    private $x;
    private $y;
    static $nb=0;
    public function setX($x){
        $this->x=$x;
    }
    public function setY($y){
        $this->y=$y;
    }
    public function __construct($x,$y){
        $this->setX($x);
        $this->setX($x);
    }
    public function getX(){
        return $this->x;
    }
    public function getY(){
        return $this->y;
    }

}
```

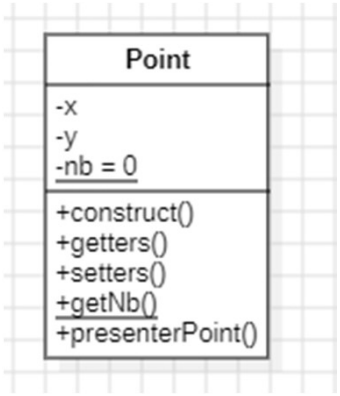
L'ACCÈS AU MEMBRE STATIQUE DANS LA CLASSE DE DÉFINITION

- L'accès au membre statique dans la classe de définition se fait via le terme clé **self::**
- Voir l'exemple:

```php
class Point{
    private $x;
    private $y;
    static $nb=0;
    public function setX($x){
        $this->x=$x;
    }
    public function setY($y){
        $this->y=$y;
    }
    public function __construct($x,$y){
        $this->setX($x);
        $this->setX($x);
        self::$nb++;
    }
    public function getX(){
        return $this->x;
    }
    public function getY(){
        return $this->y;
    }
```

DÉFINITION DES FONCTIONS D'INSTANCIATION ET STATIQUES

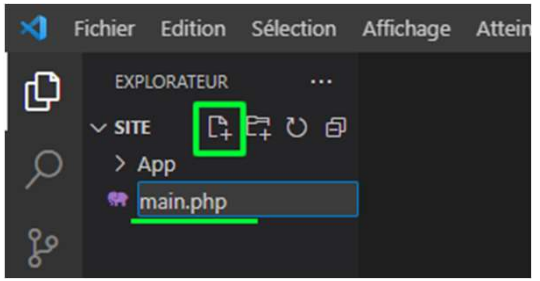
- Dans le graphique ci-dessous, un exemple de conception et implémentation d'une classe ayant des membres d'instanciation et statique




```php
class Point{
    private $x;
    private $y;
    static $nb=0;
    public function setX($x){
        $this->x=$x;
    }
    public function setY($y){
        $this->y=$y;
    }
    public function __construct($x,$y){
        $this->setX($x);
        $this->setX($x);
        self::$nb++;
    }
    public function getX(){
        return $this->x;
    }
    public function getY(){
        return $this->y;
    }
    public function presenterPoint(){
        return 'le point a pour coordonnées '.$this->getX().' '.$this->getY();
    }
    public static function getNb(){
        return self::$nb;
    }
```

CRÉATION D'OBJET ET EXÉCUTION DE SES DIFFÉRENTES MÉTHODES

- Pour créer un objet, commencer par créer un fichier .php sous votre dossier du site et qui n'appartient pas au dossier de classes. Voir l'exemple:



- Commencer votre fichier par l'inclusion du fichier de classe. L'inclusion peut se faire via l'une de quatre instructions suivantes: include(), include_once(), require, require_once(), voir l'exemple:

```
main.php
1  <?php
2      require_once('App/Point.php');
3
4  ?>
```

CRÉATION D'OBJET ET EXÉCUTION DE SES DIFFÉRENTES MÉTHODES

- Pour instancier l'objet, utiliser cette instruction

```php
<?php
    require_once('App/Point.php');
    $a=new Point(4,5);
?>
```

- Pour exécuter une méthode d'instanciation, voir l'exemple:

```php
<?php
    require_once('App/Point.php');
    $a=new Point(4,5);
    echo $a->presenterPoint();
?>
```

- Pour exécuter une méthode de classe, voir l'exemple:

```php
<?php
    require_once('App/Point.php');
    $a=new Point(4,5);
    echo $a->presenterPoint().'<br>';
    echo Point::getNb();

?>
```

Héritage en PHP

- Pour déclarer qu'une classe hérite une autre, voir l'exemple:

```php
<?php
    class PointColor extends Point{

    }
?>
```

- Pour appeler les membre de classe mère dans une classe fille, utiliser le terme clé **parent**, voir l'exemple ci-dessous:

Héritage en PHP

```php
class Point{
    private $x;
    private $y;
    static $nb=0;
    public function setX($x){
        $this->x=$x;
    }
    public function setY($y){
        $this->y=$y;
    }
    public function __construct($x,$y){
        $this->setX($x);
        $this->setX($x);
        self::$nb++;
    }
    public function getX(){
        return $this->x;
    }
    public function getY(){
        return $this->y;
    }
    public function presenterPoint(){
        return 'le point a pour coordonnées '.$this->getX().' '.$this->getY();
    }
    public static function getNb(){
        return self::$nb;
    }

}
```

Classe mère

```php
class PointColor extends Point{
    private $color;

    public function setColor($color){
        $this->color=$color;
    }
    public function getColor(){
        return $this->color;
    }

    public function __construct($x,$y,$color){
        parent::__construct($x,$y);
        $this->color=$color;
    }
    public function presenterPoint(){
        return parent::presenterPoint().' '.$this->getColor();
    }

}
```

Classe fille

L'AUTOLODING EN PHP

- Si on veut créer un objet de type PointColor, on doit charger les deux classes Point et PointColor, donc plusieurs instructions **require_once** à écrire, pour éviter ce genre du problème on peut utiliser l'autoloding, Voir l'exemple:

```php
<?php
    spl_autoload_register(function($name){
        require_once('App/'.$name.'.php');
        });
    $a=new PointColor(4,5,'red');
    echo $a->presenterPoint().'<br>';
    echo 'le nbre de Point créé est '. Point::getNb();

?>
```

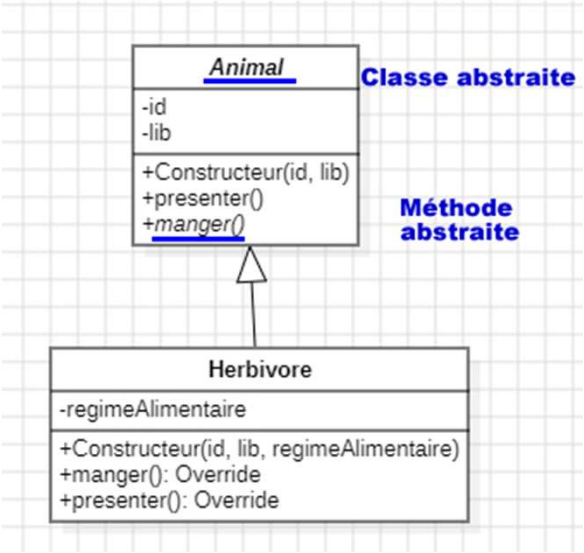
- Sachant que **App** est le dossier où on a stocké nos classes et **.php** est l'extension des fichiers qui contiennent nos classes

LES CLASSES ABSTRAITES EN PHP

- Parfois on serait obligé de créer une classe mère qui n'est pas instanciable, elle est utilisée juste pour regrouper des membres communs pour l'ensemble de classes filles,
- Si on prend le cas de notre école « ESSAI » on a trois types du personnel: Etudiant, Enseignant et administratif et ces trois ont des informations communes qu'on peut les regrouper dans une classe mère nommée Personnel, or ce statut n'existe pas dans l'école c'est pour ça on va choisir que la classe Personnel soit abstraite

LES CLASSES ABSTRAITES EN PHP

- Voici un exemple de conception: diagramme de classe pour une classe abstraite et sa fille qui l'hérite:



- Voici la traduction de cette conception en langage PHP:

LES CLASSES ABSTRAITES EN PHP

```php
<?php
abstract class Animal{
    private $id;
    private $lib;
    public function __construct($id,$lib)
    {
        $this->id=$id;
        $this->lib=$lib;
    }
    public function presenter(){
        return '<br>Je suis un animal identifié par '.$this->id.' mon nom est '.$this->lib;

    }
    abstract public function manger($nourrriture);

}

?>
```

- Une classe abstraite peut contenir des méthodes abstraites qui sont juste des signatures et les classes filles qui l'héritent cette doivent redéfinir ces méthodes

LES CLASSES ABSTRAITES EN PHP

- Voir l'exemple :

```php
<?php
class Herbivore extends Animal{
    private $regimeAlimentaire;
    public function __construct($id,$lib,$regimeAlimentaire)
    {
        parent::__construct($id,$lib,$regimeAlimentaire);
         $this->regimeAlimentaire =$regimeAlimentaire;

    }
    public function presenter(){
        return parent::presenter().'j\'ai un régime: '. $this->regimeAlimentaire;

    }
    public function manger($nourriture){
        return '<br> je mange de '.$nourriture;
    }

}
```

Les classes abstraites en PHP

- Exécution:

```php
<?php
    spl_autoload_register(function($name){

        require_once('src/'.$name.'.php');

    });
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <?php
      $a=new Herbivore(1,'gazelle','Végétarien');
      echo $a->presenter();
      echo $a->manger('herbe');

?>
</body>
</html>
```

Importation des classes

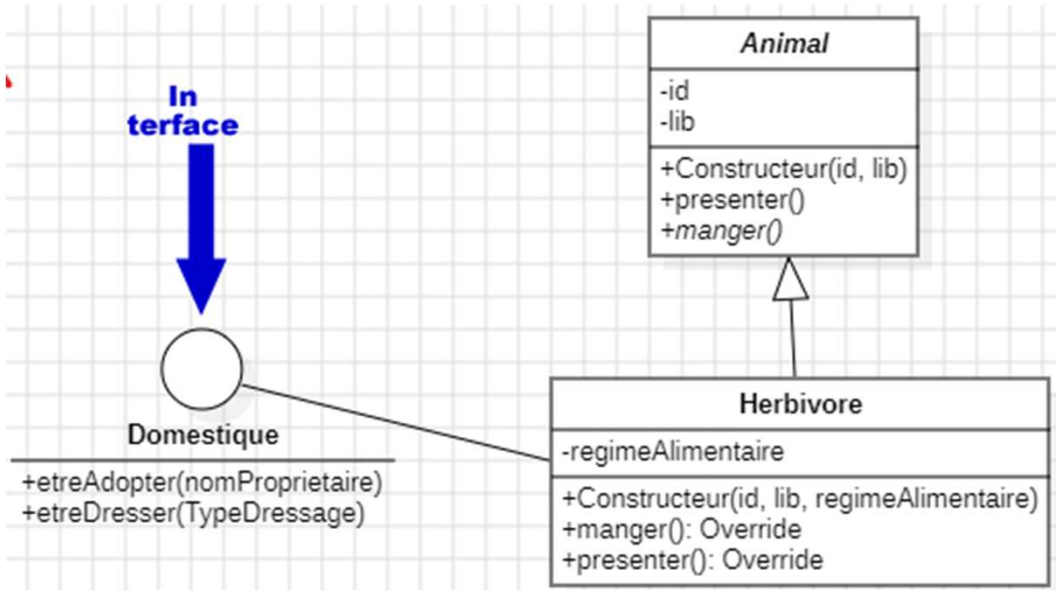
Création de l'objet

LES INTERFACE EN PHP

- Une interface est une forme particulière de classe où toutes les méthodes sont abstraites.
- Lorsqu'une classe implémente une interface, elle indique ainsi qu'elle s'engage à fournir une implémentation (c'est-à-dire un corps) pour chacune des méthodes abstraites de cette interface.

LES INTERFACES EN PHP

- Voici un exemple de conception: diagramme de classe pour l'implémentation d'une interface par une classe



- Voici l'implémentation de cette conception en langage PHP:

LES INTERFACES EN PHP

- Les classes et l'interface:

```php
<?php
class Herbivore extends Animal implements Domestique{
    private $regimeAlimentaire;
    public function __construct($id,$lib,$regimeAlimentaire)
    {
        parent::__construct($id,$lib,$regimeAlimentaire);
        $this->regimeAlimentaire =$regimeAlimentaire;

    }
    public function presenter(){
        return parent::presenter().'j\'ai un régime: '. $this->regimeAlimentaire;

    }
    public function manger($nourriture){
        return '<br> je mange de '.$nourriture;
    }
    public function etreAdopter($nomProprietaire){
        return '<br> je suis adepté(e)'.$nomProprietaire;
    }
    function etredresser($typeDressage){
        return '<br> je suis dressé(e) avec un type de dressage '.$typeDressage;
    }

}
```

```php
<?php
interface Domestique{
    public function etreAdopter($nomProprietaire);
    public function etredresser($typeDressage);
}
```

```php
<?php
abstract class Animal{
    private $id;
    private $lib;
    public function __construct($id,$lib)
    {
        $this->id=$id;
        $this->lib=$lib;
    }
    public function presenter(){
        return '<br>Je suis un animal identifié par '.$this->id.' mon nom est '.$this->lib;

    }
    abstract public function manger($nourrriture);

}

?>
```

LES INTERFACES EN PHP

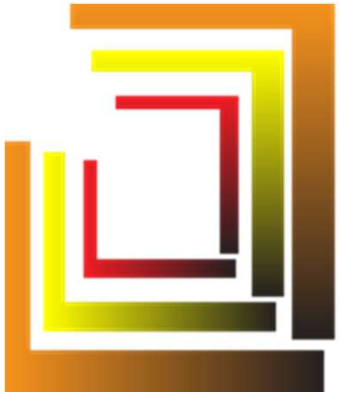
- Exécution:

```php
<?php
    spl_autoload_register(function($name){

        require_once('src/'.$name.'.php');

    });
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <?php
      $a=new Herbivore(1,'gazelle','Végétarien');
      echo $a->presenter();
      echo $a->manger('herbe');
      echo $a->etreAdopter('flen');
      echo $a->etredresser('Hard');

?>
</body>
</html>
```



FIJN

Mansour Sihem