Cours de PHP orienté objet

Créer une classe Point en PHP et initialiser les variables d'instance

1) Testez le code ci-dessous. Il définit une classe Point puis crée une instance de Point dans la variable $pt. Remarquez la ressemblance avec les classes Java.

```php
<?php
// Oblige PHP à vérifier les types de données
declare(strict_types=1);

// Définition de la classe Point
class Point
{
   public float $x;
   public float $y;
}

// Création d'un objet de type Point
$pt = new Point;
$pt->x = 2.5;
$pt->y = 5.4;
var_dump($pt->x);
var_dump($pt->y);
?>
```

---

2) Tout comme en Java, nous allons rendre les attributs de Point private. Le code ci-dessous génère donc une erreur “Cannot access private property”.

```php
<?php
// Oblige PHP à vérifier les types de données
declare(strict_types=1);

// Définition de la classe Point
class Point
{
   private float $x;
   private float $y;
}

// Création d'un objet de type Point
$pt = new Point;
$pt->x = 2.5;
$pt->y = 5.4;
var_dump($pt->x);
```

```
?>
```

---

3) Nous ajoutons des getters et des setters

```php
<?php
// Oblige PHP à vérifier les types de données
declare(strict_types=1);

// Définition de la classe Point
class Point
{
   private float $x;
   private float $y;
   public function distance(): float {
       return sqrt(pow($this->x,2) + pow($this->y,2));
   }
   public function getX(): float {
      return $this->x;
   }
   public function getY(): float {
      return $this->y;
   }
   public function setX(float $pos_x): void
   {
       if ($pos_x < 0) {
           trigger_error(
               'Le x doit être supérieur ou égal à zéro',
               E_USER_ERROR
           );
       }

       $this->x = $pos_x;
   }

   public function setY(float $pos_y): void
   {
       if ($pos_y < 0) {
           trigger_error(
               'Le y doit être supérieur ou égal à zéro',
               E_USER_ERROR
           );
       }

       $this->y = $pos_y;
   }
}

// Création d'un objet de type Point
```

```php
$pt = new Point;
$pt->setX(2.5);
$pt->setY(5.4);
var_dump($pt->getX());
var_dump($pt->getY());
?>
```

---

4) Nouvelle méthode : distance qui retourne la distance à l'origine du repère, le type de retour est float (il est indiqué après les deux points :)

```php
<?php
// Oblige PHP à vérifier les types de données
declare(strict_types=1);

// Définition de la classe Point
class Point
{
   private float $x;
   private float $y;

   public function distance(): float
   {
       return sqrt(pow($this->x,2) + pow($this->y,2));
   }

   public function getX(): float {
      return $this->x;
   }
   public function getY(): float {
      return $this->y;
   }
   public function setX(float $pos_x): void
   {
       if ($pos_x < 0) {
           trigger_error(
               'Le x doit être supérieur ou égal à zéro',
               E_USER_ERROR
           );
       }

       $this->x = $pos_x;
   }

   public function setY(float $pos_y): void
   {
       if ($pos_y < 0) {
           trigger_error(
               'Le y doit être supérieur ou égal à zéro',
```

```
                E_USER_ERROR
            );
        }

        $this->y = $pos_y;
    }
}

// Création d'un objet de type Point
$pt = new Point;
$pt->setX(2.5);
$pt->setY(5.4);
var_dump($pt->getX());
var_dump($pt->getY());


?>
```

---

5) Ajout d'un constructeur à deux paramètres

```
<?php
// Oblige PHP à vérifier les types de données
declare(strict_types=1);

class Point
{
   private float $x;
   private float $y;

   public function __construct($pos_x, $pos_y){
      $this->x = $pos_x;
      $this->y = $pos_y;
   }
   public function distance(): float
   {
       return sqrt(pow($this->x,2) + pow($this->y,2));
   }

   public function getX(): float {
      return $this->x;
   }
   public function getY(): float {
      return $this->y;
   }
   public function setX(float $pos_x): void
   {
       if ($pos_x < 0) {
           trigger_error(
```

```
                'Le x doit être supérieur ou égal à zéro',
                E_USER_ERROR
            );
        }

        $this->x = $pos_x;
    }

    public function setY(float $pos_y): void
    {
        if ($pos_y < 0) {
            trigger_error(
                'Le y doit être supérieur ou égal à zéro',
                E_USER_ERROR
            );
        }

        $this->y = $pos_y;
    }
}

$pt = new Point(2.5,5.4);
var_dump($pt->getX());
var_dump($pt->getY());
echo "La distance du point à l'origine du repère est de " .
$pt->distance();
?>
```

---

6) Variable de classe : nombrePoints

- Remarquez que pour accéder à une variable de classe il faut la précéder de *self::*
- Pour appeler une variable ou une méthode de classe en dehors de la classe il faudra utiliser de nom de la classe donc *Point::*

```
<?php
// Oblige PHP à vérifier les types de données
declare(strict_types=1);

// Définition de la classe Point
class Point
{
    private static int $nombrePoints = 0;
    private float $x;
    private float $y;

    public function __construct($pos_x, $pos_y){
        $this->x = $pos_x;
        $this->y = $pos_y;
```

```
        self::$nombrePoints++;
    }

    public function distance(): float
    {
        return sqrt(pow($this->x,2) + pow($this->y,2));
    }

    public function getX(): float {
        return $this->x;
    }
    public function getY(): float {
        return $this->y;
    }
    public function setX(float $pos_x): void
    {
        if ($pos_x < 0) {
            trigger_error(
                'Le x doit être supérieur ou égal à zéro',
                E_USER_ERROR
            );
        }

        $this->x = $pos_x;
    }

    public function setY(float $pos_y): void
    {
        if ($pos_y < 0) {
            trigger_error(
                'Le y doit être supérieur ou égal à zéro',
                E_USER_ERROR
            );
        }

        $this->y = $pos_y;
    }
}

// Création d'un objet de type Point
$pt = new Point(2.5,4.6);
echo "La distance du point à l'origine du repère est de " .
$pt->distance();

?>
```

Exercices

Exercice 1 :

Ajoutez d'autres points, un getter static pour récupérer nombrePoint et affichez le nombre d'objets Point.

Exercice 2 :

Ajoutez une variable d'instance z de type float et modifiez le constructeur pour qu'il prenne un troisième paramètre pour pos_z.

Connexion à la base de données et exploitation des données en PHP Orienté Objet

- Importez le fichier
  coordonnees_db.sql
  dans une base de données coordonnees_db.
- Au BTS Blanc il y aura un code de connexion à la base de données avec PDO ainsi que des requêtes. Vous ne devez donc pas apprendre à l'écrire par cœur mais vous devez le comprendre dans ses grandes lignes.

---

7) Ci-dessous le code PHP permettant de se connecter à la base de données à l'aide du PDO (PHP Data Object).

- Remarquez que cela ressemble beaucoup à la version procédurale mysqli_connect vue précédemment.
- Ce code affiche le contenu des lignes de manière peu lisible. Voir le point 7) pour produire un meilleur affichage
- Remplacez 'SELECT * from Point;' par la requête de votre choix pour faire des tests.

```
try {
   $user="root";
   $pass="";
   $dbh = new PDO('mysql:host=localhost;dbname=coordonnees_db', $user,
$pass);
   foreach($dbh->query('SELECT * from Point;') as $row) {
       print_r($row);
   }
   $dbh = null;
} catch (PDOException $e) {
   print "Erreur !: " . $e->getMessage() . "<br/>";
   die();
}
```

---

8) Affichage amélioré

- Ici $row est un tableau associatif récupéré automatiquement grâce au foreach et à l'appel à query

```
try {
   $user="root";
   $pass="";
   $dbh = new PDO('mysql:host=localhost;dbname=coordonnees_db', $user,
$pass);
   foreach($dbh->query("SELECT * from figure;") as $row) {
      echo "<h2>" . $row["forme"] . " " . $row["couleur"] . "</h2>";
   }
   $dbh = null;
} catch (PDOException $e) {
   print "Erreur !: " . $e->getMessage() . "<br/>";
   die();
}
```

---

9) Nous allons parcourir les figures en utilisant une autre méthode : les requêtes préparées. Vous comprendrez l'utilité de cette méthode un peu plus longue dans le point n°10.

```
try {
   $user="root";
   $pass="";
   $dbh = new PDO('mysql:host=localhost;dbname=coordonnees_db', $user,
$pass);
   $req = "select * from figure;";
   $res = $dbh->prepare($req);
   $res->execute();
   $lesFigures = $res->fetchAll();
   foreach($lesFigures as $uneFigure) {
      echo "<h2>" . $uneFigure["forme"] . " " . $uneFigure["couleur"] .
"</h2>";
   }

   $dbh = null;
} catch (PDOException $e) {
   print "Erreur !: " . $e->getMessage() . "<br/>";
   die();
}
```

---

10) Il est possible de lier (bind) des paramètres à la requête SQL. Il faut pour cela utiliser les deux-points :

- Je ne souhaite afficher que les figures de forme triangle

- Donc on ajoute un WHERE dans la requête SQL. Plutôt que d'écrire WHERE forme='triangle', je préfère rendre le code plus dynamique. On donne un nom précédé des deux-points, exemple avec *:maforme* . Puis on utilise la fonction bind pour dire à php de remplacer *:maforme* par le contenu de la variable $forme_choisie dans la requête.

```
try {
   $user="root";
   $pass="";
   $dbh = new PDO('mysql:host=localhost;dbname=coordonnees_db', $user,
$pass);
  /* foreach($dbh->query("SELECT * from figure;") as $row) {
      echo "<h2>" . $row["forme"] . " " . $row["couleur"] . "</h2>";
   }
   */
   $forme_choisie = "triangle";
   $req = "select * from figure where forme=:maforme;";
   $res = $dbh->prepare($req);
   $res->bindParam(':maforme', $forme_choisie);
   $res->execute();
   $lesFigures = $res->fetchAll();
   foreach($lesFigures as $uneFigure) {
      echo "<h2>" . $uneFigure["forme"] . " " . $uneFigure["couleur"] .
"</h2>";
   }
   $dbh = null;
} catch (PDOException $e) {
   print "Erreur !: " . $e->getMessage() . "<br/>";
   die();
}
```

Exercices

Exercice 3 :

- Reprendre le code précédent et faites la même chose pour afficher la liste des points. N'afficher que les points qui ont un x supérieur à 10 grâce à une requête préparée.

Exercice 4 :

- Reprendre le code du point **9)** et pour chaque figure, faire une requête qui permet d'afficher tous les points de cette figure.

CORRECTION

```
try {
   $user="root";
   $pass="";
   $dbh = new PDO('mysql:host=localhost;dbname=coordonnees_db', $user,
$pass);
   $req = "select * from figure;";
   $res = $dbh->prepare($req);
   $res->execute();
   $lesFigures = $res->fetchAll();
   foreach($lesFigures as $uneFigure) {
      echo "<h2>" . $uneFigure["forme"] . " " . $uneFigure["couleur"] .
"</h2>";
      $req = "select distinct point.* from point join appartient on
point.id=appartient.id_point join figure on figure.id=appartient.id_figure
where figure.id=:id;" ;
      $res = $dbh->prepare($req);
      $res->bindParam(':id', $uneFigure['id']);
      $res->execute();
      $lesPoints = $res->fetchAll();
      echo "<ul>";
      foreach($lesPoints as $unPoint) {
         $nouveauPoint = new Point($unPoint['x'], $unPoint['y'],
$unPoint['z']);
         echo "<li>(".$nouveauPoint->getX() . ", ".
$nouveauPoint->getY().")</li>";
      }
      echo "</ul>";
   }
   $dbh = null;
} catch (PDOException $e) {
   print "Erreur !: " . $e->getMessage() . "<br/>";
   die();
}
```