Cours Complet – Programmation Orientée Objet (POO)

E LEARNING BTS

Pour Débutants – Septembre 2025

Contents

1 Introduction à la Programmation Orientée Objet

Qu'est-ce que la POO ?

La **Programmation Orientée Objet (POO)** est un paradigme de programmation qui organise le code autour d'objets plutôt que d'actions, et de données plutôt que de logique.

Avantages de la POO :

- **Réutilisabilité** : Les classes peuvent être réutilisées dans différents projets
- **Modularité** : Le code est organisé en modules indépendants
- **Maintenabilité** : Plus facile à maintenir et à faire évoluer
- **Abstraction** : Cache la complexité et expose seulement ce qui est nécessaire

1.1 Les Concepts Fondamentaux

1. La Classe

Une **classe** est un modèle (ou un plan) qui définit la structure et le comportement d'objets.
Elle contient :

- **Attributs** (variables) : Les propriétés de l'objet
- **Méthodes** (fonctions) : Les actions que l'objet peut effectuer
- **Constructeur** : Une méthode spéciale pour initialiser l'objet

Analogie : Une classe est comme un plan architectural d'une maison. Elle décrit comment la maison doit être construite, mais ce n'est pas la maison elle-même.

2. L'Objet

Un **objet** est une instance (réalisation concrète) d'une classe.
Analogie : Si la classe est le plan architectural, l'objet est la maison réelle construite à partir de ce plan. On peut construire plusieurs maisons (objets) à partir du même plan (classe).

Exemple :

- Classe : `Voiture`
- Objets : `maVoiture`, `voitureDePaul`, `voitureRouge`

3. Les Attributs

Les **attributs** sont les variables qui stockent les données de l'objet.
Visibilité des attributs :

- **private** : Accessible uniquement à l'intérieur de la classe
- **public** : Accessible de partout

- **protected** : Accessible dans la classe et ses classes dérivées

Bonne pratique : Déclarez toujours vos attributs en `private` et utilisez des getters/setters pour y accéder (principe d'encapsulation).

4. Les Méthodes

Les **méthodes** sont des fonctions définies dans une classe qui décrivent les comportements de l'objet.
Types de méthodes :

- **Constructeur** : Initialise l'objet lors de sa création
- **Getters** : Retournent la valeur d'un attribut
- **Setters** : Modifient la valeur d'un attribut
- **Méthodes métier** : Effectuent des traitements spécifiques

5. Le Constructeur

Le **constructeur** est une méthode spéciale appelée automatiquement lors de la création d'un objet.
Rôle : Initialiser les attributs de l'objet avec des valeurs par défaut ou des valeurs passées en paramètres.
Caractéristiques :

- Porte le même nom que la classe (en Java)
- S'appelle `__construct()` en PHP
- Ne retourne aucune valeur

6. L'Encapsulation

L'**encapsulation** consiste à cacher les détails internes d'une classe et à n'exposer que ce qui est nécessaire.
Principe :

- Attributs en `private`
- Accès contrôlé via getters/setters
- Protection des données contre les modifications non autorisées

Avantage : Permet de modifier l'implémentation interne sans affecter le code qui utilise la classe.

2 Syntaxe en PHP

Structure d'une Classe en PHP

```php
<?php
class NomDeLaClasse {
    // Attributs
    private $attribut1;
    private $attribut2;

    // Constructeur
    public function __construct() {
        $this->attribut1 = valeur1;
        $this->attribut2 = valeur2;
    }

    // M thodes
    public function nomMethode() {
        // code
    }

    // Getters et Setters
    public function getAttribut1() {
        return $this->attribut1;
    }

    public function setAttribut1($valeur) {
        $this->attribut1 = $valeur;
    }
}
?>
```

Points importants :

- `$this` fait référence à l'objet courant
- `->` est l'opérateur d'accès aux membres
- Les méthodes doivent spécifier leur visibilité (public, private, protected)

Instanciation et Utilisation en PHP

```php
<?php
// Instanciation (cr ation d'un objet)
$monObjet = new NomDeLaClasse();

// Appel d'une m thode
$monObjet->nomMethode();

// Utilisation d'un getter
$valeur = $monObjet->getAttribut1();

// Utilisation d'un setter
$monObjet->setAttribut1(nouvelleValeur);
?>
```

3 Syntaxe en Java

Structure d'une Classe en Java

```java
public class NomDeLaClasse {
    // Attributs
    private int attribut1;
    private String attribut2;

    // Constructeur
    public NomDeLaClasse() {
        this.attribut1 = 0;
        this.attribut2 = "";
    }

    // M thodes
    public void nomMethode() {
        // code
    }

    // Getters et Setters
    public int getAttribut1() {
        return this.attribut1;
    }

    public void setAttribut1(int valeur) {
        this.attribut1 = valeur;
    }
}
```

Points importants :

- Le constructeur porte le même nom que la classe
- `this` fait référence à l'objet courant
- Les types doivent être explicitement déclarés

Instanciation et Utilisation en Java

```java
// Instanciation (cr ation d'un objet)
NomDeLaClasse monObjet = new NomDeLaClasse();

// Appel d'une m thode
monObjet.nomMethode();

// Utilisation d'un getter
int valeur = monObjet.getAttribut1();

// Utilisation d'un setter
monObjet.setAttribut1(nouvelleValeur);
```

4 Exemple Complet : Classe Rectangle

4.1 Analyse du Problème

Cahier des Charges

Objectif : Créer une classe Rectangle qui permet de gérer un rectangle.
Attributs nécessaires :

- Longueur (lg)
- Largeur (lr)

Méthodes nécessaires :

- Saisir les dimensions
- Afficher les dimensions
- Calculer la surface
- Calculer le périmètre
- Getters et Setters pour chaque attribut

4.2 Implémentation en PHP

Classe Rectangle en PHP (rectangle.class.php)

```php
<?php
class Rectangle {
    // Attributs
    private $lg, $lr;

    // Constructeur
    public function __construct() {
        // Initialise les attributs
        $this->lg = 0;
        $this->lr = 0;
    }

    // M thodes
    public function saisir($tab) {
        // $tab --> $_POST du formulaire
        $this->lg = $tab['lg'];
        $this->lr = $tab['lr'];
    }

    public function afficher() {
        return "<br>Longueur : " . $this->lg .
               "<br>Largeur : " . $this->lr;
    }

    public function surface() {
        return $this->lg * $this->lr;
    }

    public function perimetre() {
```

```
30          return 2 * ($this->lg + $this->lr);
31      }
32
33      // Getters et Setters
34      public function getLg() {
35          return $this->lg;
36      }
37
38      public function setLg($lg) {
39          $this->lg = $lg;
40      }
41
42      public function getLr() {
43          return $this->lr;
44      }
45
46      public function setLr($lr) {
47          $this->lr = $lr;
48      }
49  }
50  ?>
```

Page HTML/PHP d'utilisation (index.php)

```php
<?php
require_once("rectangle.class.php");
?>
<!DOCTYPE html>
<html>
<head>
    <title>Rectangle</title>
    <meta charset="utf-8">
</head>
<body>
<center>
    <h1>Gestion de la classe Rectangle</h1>
    <form method="post">
        Donnez la longueur : <br>
        <input type="text" name="lg"> <br>
        Donnez la largeur : <br>
        <input type="text" name="lr"> <br>
        <input type="submit" name="Valider" value="Valider">
    </form>

    <?php
    if(isset($_POST['Valider'])) {
        // Instancier la classe Rectangle
        $unRectangle = new Rectangle();

        // Appel de la methode saisir
        $unRectangle->saisir($_POST);

        // Appel de la methode afficher
        echo $unRectangle->afficher();

        // Appel de la methode surface
        echo "<br>La surface est de : " .
             $unRectangle->surface();

        // Appel de la methode perimetre
        echo "<br>Le perimetre est de : " .
             $unRectangle->perimetre();

        // Afficher juste la longueur
        echo "<br>La longueur est de : " .
             $unRectangle->getLg();

        // Changer la valeur de la largeur en 15
        $unRectangle->setLr(15);
        echo "<br>Nouvelle largeur : " .
             $unRectangle->getLr();
    }
    ?>
</center>
</body>
</html>
```

4.3 Implémentation en Java

Classe Rectangle en Java (Rectangle.java)

```java
import java.util.Scanner;

public class Rectangle {
    // Attributs
    private float lg, lr;

    // Constructeur
    public Rectangle() {
        // Initialiser les attributs
        this.lg = 0;
        this.lr = 0;
    }

    // M thodes
    public void saisir() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Donner la longueur : ");
        this.lg = sc.nextFloat();
        System.out.print("Donner la largeur : ");
        this.lr = sc.nextFloat();
    }

    public void afficher() {
        System.out.println("La longueur est de : " + this.lg);
        System.out.println("La largeur est de : " + this.lr);
    }

    public float surface() {
        return this.lg * this.lr;
    }

    public float perimetre() {
        return 2 * (this.lg + this.lr);
    }

    public void menu() {
        int choix = 0;
        Scanner sc = new Scanner(System.in);

        do {
            System.out.println("___ Menu Rectangle ___");
            System.out.println("1- Saisir les cotes");
            System.out.println("2- Afficher les cotes");
            System.out.println("3- Surface");
            System.out.println("4- Perimetre");
            System.out.println("0- Quitter");
            System.out.print("Votre choix : ");
            choix = sc.nextInt();

            switch(choix) {
                case 1: this.saisir(); break;
                case 2: this.afficher(); break;
                case 3:
                    System.out.println("Surface : " +
                                       this.surface());
```

```
                    break;
                case 4:
                    System.out.println("Perimetre : " +
                                       this.perimetre());
                    break;
                case 0:
                    System.out.println("Au revoir !");
                    break;
                default:
                    System.out.println("Erreur de saisie");
                    break;
            }
        } while(choix != 0);
    }

    // Getters et Setters
    public float getLg() {
        return lg;
    }

    public void setLg(float lg) {
        this.lg = lg;
    }

    public float getLr() {
        return lr;
    }

    public void setLr(float lr) {
        this.lr = lr;
    }
}
```

Classe de Gestion en Java (Gestion.java)

```java
public class Gestion {
    public static void main(String[] args) {
        // Instancier la classe Rectangle
        Rectangle unRectangle = new Rectangle();

        // Appel    la m thode menu
        unRectangle.menu();
    }
}
```

Explications :

- La méthode `main` est le point d'entrée du programme
- On crée un objet Rectangle
- On appelle la méthode `menu()` qui gère toute l'interaction

5 Exercices Pratiques

5.1 Exercice 1 : Classe Cercle

Énoncé - Classe Cercle

Objectif : Créer une classe `Cercle` qui permet de gérer un cercle.
Attributs :

- `rayon` : le rayon du cercle (type : réel)

Méthodes à implémenter :

- `saisir()` : demande le rayon à l'utilisateur
- `afficher()` : affiche le rayon
- `surface()` : calcule et retourne la surface ($\pi \times rayon^2$)
- `perimetre()` : calcule et retourne le périmètre ($2 \times \pi \times rayon$)
- Getters et Setters pour le rayon

Travail à faire : Implémentez cette classe en PHP et en Java avec une interface utilisateur complète.

Tips : Utilisez `M_PI` en PHP et `Math.PI` en Java pour la valeur de π.

Solution PHP - Classe Cercle

Fichier cercle.class.php :

```php
<?php
class Cercle {
    // Attribut
    private $rayon;

    // Constructeur
    public function __construct() {
        $this->rayon = 0;
    }

    // M thodes
    public function saisir($tab) {
        $this->rayon = $tab['rayon'];
    }

    public function afficher() {
        return "<br>Rayon : " . $this->rayon;
    }

    public function surface() {
        return M_PI * $this->rayon * $this->rayon;
    }

    public function perimetre() {
        return 2 * M_PI * $this->rayon;
    }

    // Getters et Setters
    public function getRayon() {
        return $this->rayon;
    }

    public function setRayon($rayon) {
        $this->rayon = $rayon;
    }
}
?>
```

Solution PHP - Page d'utilisation

Fichier index.php :

```php
<?php
require_once("cercle.class.php");
?>
<!DOCTYPE html>
<html>
<head>
    <title>Cercle</title>
    <meta charset="utf-8">
</head>
<body>
<center>
```

```
12      <h1>Gestion de la classe Cercle</h1>
13      <form method="post">
14          Donnez le rayon : <br>
15          <input type="text" name="rayon"> <br>
16          <input type="submit" name="Valider" value="Valider">
17      </form>
18
19      <?php
20      if(isset($_POST['Valider'])) {
21          $unCercle = new Cercle();
22          $unCercle->saisir($_POST);
23
24          echo $unCercle->afficher();
25          echo "<br>La surface est de : " .
26               number_format($unCercle->surface(), 2);
27          echo "<br>Le perimetre est de : " .
28               number_format($unCercle->perimetre(), 2);
29      }
30      ?>
31  </center>
32  </body>
33  </html>
```

Solution Java - Classe Cercle

```java
import java.util.Scanner;

public class Cercle {
    // Attribut
    private float rayon;

    // Constructeur
    public Cercle() {
        this.rayon = 0;
    }

    // M thodes
    public void saisir() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Donner le rayon : ");
        this.rayon = sc.nextFloat();
    }

    public void afficher() {
        System.out.println("Le rayon est de : " + this.rayon);
    }

    public double surface() {
        return Math.PI * this.rayon * this.rayon;
    }

    public double perimetre() {
        return 2 * Math.PI * this.rayon;
    }

    public void menu() {
        int choix = 0;
        Scanner sc = new Scanner(System.in);

        do {
            System.out.println("___ Menu Cercle ___");
            System.out.println("1- Saisir le rayon");
            System.out.println("2- Afficher le rayon");
            System.out.println("3- Surface");
            System.out.println("4- Perimetre");
            System.out.println("0- Quitter");
            System.out.print("Votre choix : ");
            choix = sc.nextInt();

            switch(choix) {
                case 1: this.saisir(); break;
                case 2: this.afficher(); break;
                case 3:
                    System.out.printf("Surface : %.2f\n",
                                      this.surface());
                    break;
                case 4:
                    System.out.printf("Perimetre : %.2f\n",
                                      this.perimetre());
                    break;
                case 0:
                    System.out.println("Au revoir !");
```

```
                    break;
                default:
                    System.out.println("Erreur de saisie");
                    break;
            }
        } while(choix != 0);
    }

    // Getters et Setters
    public float getRayon() {
        return rayon;
    }

    public void setRayon(float rayon) {
        this.rayon = rayon;
    }
}
```

5.2 Exercice 2 : Classe Compte Bancaire

Énoncé - Classe Compte

Objectif : Créer une classe `Compte` qui permet de gérer un compte bancaire.
Attributs :

- `nom` : le nom du titulaire
- `prenom` : le prénom du titulaire
- `numero` : le numéro de compte
- `solde` : le solde du compte (initialisé à 80)

Méthodes à implémenter :

- `ouvrir()` : saisit nom, prénom et numéro
- `consulter()` : affiche toutes les informations du compte
- `deposer()` : ajoute une somme au solde
- `retirer()` : retire une somme du solde (avec vérification)
- `menu()` : menu interactif
- Getters et Setters

Contraintes :

- On ne peut retirer que si le solde est suffisant
- Les montants doivent être positifs

Solution Java - Classe Compte

```java
import java.util.Scanner;

public class Compte {
    // Attributs
    private String nom, prenom;
    private int numero;
    private float solde;

    // Constructeur
    public Compte() {
        this.nom = "";
        this.prenom = "";
        this.numero = 0;
        this.solde = 80;
    }

    // M thodes
    public void ouvrir() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Nom : ");
        this.nom = sc.next();
        System.out.print("Prenom : ");
        this.prenom = sc.next();
        System.out.print("Numero : ");
        this.numero = sc.nextInt();
    }

    public void consulter() {
        System.out.println("Nom    : " + this.nom);
        System.out.println("Prenom : " + this.prenom);
        System.out.println("Numero : " + this.numero);
        System.out.println("Solde  : " + this.solde);
    }

    public void deposer() {
        Scanner sc = new Scanner(System.inlatex);
        System.out.print("Somme    d poser : ");
        float somme = sc.nextFloat();

        if (somme > 0) {
            this.solde += somme;
            System.out.println("Nouveau solde : " + this.solde);
        } else {
            System.out.println("Erreur de saisie");
        }
    }

    public void retirer() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Somme    retirer : ");
        float somme = sc.nextFloat();

        if (somme > 0 && somme <= this.solde) {
            this.solde -= somme;
            System.out.println("Nouveau solde : " + this.solde);
        } else {
            System.out.println("Montant invalide ou " +
```

```
                                "solde insuffisant.");
        }
    }

    public void menu() {
        Scanner sc = new Scanner(System.in);
        int choix = 0;

        do {
            System.out.println("\n____ Gestion Compte ____");
            System.out.println("1- Ouvrir le compte");
            System.out.println("2- Consulter le compte");
            System.out.println("3- D poser une somme");
            System.out.println("4- Retirer une somme");
            System.out.println("0- Quitter");
            System.out.print("Votre choix : ");
            choix = sc.nextInt();

            switch(choix) {
                case 1: this.ouvrir(); break;
                case 2: this.consulter(); break;
                case 3: this.deposer(); break;
                case 4: this.retirer(); break;
                case 0:
                    System.out.println("Au revoir !");
                    break;
                default:
                    System.out.println("Erreur de saisie");
                    break;
            }
        } while(choix != 0);
    }

    // Getters et Setters
    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getPrenom() {
        return prenom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public int getNumero() {
        return numero;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

```

```
116     public float getSolde() {
117         return solde;
118     }
119 
120     public void setSolde(float solde) {
121         this.solde = solde;
122     }
123 }
```

Solution Java - Classe Gestion

```java
public class Gestion {
    public static void main(String[] args) {
        // Instanciation de la classe Compte
        Compte unCompte = new Compte();

        // Appel de la m thode menu
        unCompte.menu();
    }
}
```

Solution PHP - Classe Compte

Fichier compte.class.php :

```php
<?php
class Compte {
    // Attributs
    private $nom, $prenom;
    private $numero;
    private $solde;

    // Constructeur
    public function __construct() {
        $this->nom = "";
        $this->prenom = "";
        $this->numero = 0;
        $this->solde = 80;
    }

    // M thodes
    public function ouvrir($tab) {
        $this->nom = $tab['nom'];
        $this->prenom = $tab['prenom'];
        $this->numero = $tab['numero'];
    }

    public function consulter() {
        return "<br>Nom : " . $this->nom .
               "<br>Prenom : " . $this->prenom .
               "<br>Numero : " . $this->numero .
               "<br>Solde : " . $this->solde;
    }

    public function deposer($somme) {
        if ($somme > 0) {
            $this->solde += $somme;
            return "Nouveau solde : " . $this->solde;
        } else {
            return "Erreur : montant invalide";
        }
    }

    public function retirer($somme) {
        if ($somme > 0 && $somme <= $this->solde) {
```

```php
            $this->solde -= $somme;
            return "Nouveau solde : " . $this->solde;
        } else {
            return "Erreur : montant invalide ou " .
                   "solde insuffisant";
        }
    }

    // Getters et Setters
    public function getNom() {
        return $this->nom;
    }

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

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

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

    public function getNumero() {
        return $this->numero;
    }

    public function setNumero($numero) {
        $this->numero = $numero;
    }

    public function getSolde() {
        return $this->solde;
    }

    public function setSolde($solde) {
        $this->solde = $solde;
    }
}
?>
```

Solution PHP - Page d'utilisation

Fichier index.php :

```php
<?php
require_once("compte.class.php");
session_start();

// Cr er le compte en session s'il n'existe pas
if (!isset($_SESSION['compte'])) {
    $_SESSION['compte'] = new Compte();
}

$monCompte = $_SESSION['compte'];
?>
<!DOCTYPE html>
<html>
<head>
    <title>Gestion Compte</title>
    <meta charset="utf-8">
    <style>
        body { font-family: Arial, sans-serif; }
        .container { width: 500px; margin: 50px auto; }
        button { margin: 10px; padding: 10px 20px; }
        input { padding: 5px; margin: 5px; }
        .result {
            background: #f0f0f0;
            padding: 20px;
            margin: 20px 0;
        }
    </style>
</head>
<body>
<div class="container">
    <h1>Gestion de Compte Bancaire</h1>

    <!-- Formulaire d'ouverture -->
    <h2>Ouvrir un compte</h2>
    <form method="post">
        <input type="text" name="nom" placeholder="Nom"
               required><br>
        <input type="text" name="prenom" placeholder="Prenom"
               required><br>
        <input type="number" name="numero"
               placeholder="Numero" required><br>
        <button type="submit" name="ouvrir">
            Ouvrir le compte
        </button>
    </form>

    <!-- Formulaire de d p t -->
    <h2>D poser une somme</h2>
    <form method="post">
        <input type="number" step="0.01" name="montantDepot"
               placeholder="Montant" required><br>
        <button type="submit" name="deposer">D poser</button>
    </form>

    <!-- Formulaire de retrait -->
    <h2>Retirer une somme</h2>
```

```
    <form method="post">
        <input type="number" step="0.01" name="montantRetrait"
               placeholder="Montant" required><br>
        <button type="submit" name="retirer">Retirer</button>
    </form>

    <!-- Consulter -->
    <form method="post">
        <button type="submit" name="consulter">
            Consulter le compte
        </button>
    </form>

    <!-- Affichage des r sultats -->
    <div class="result">
        <?php
        if (isset($_POST['ouvrir'])) {
            $monCompte->ouvrir($_POST);
            echo "Compte ouvert avec succ s !";
        }

        if (isset($_POST['deposer'])) {
            echo $monCompte->deposer($_POST['montantDepot']);
        }

        if (isset($_POST['retirer'])) {
            echo $monCompte->retirer($_POST['montantRetrait']);
        }

        if (isset($_POST['consulter'])) {
            echo $monCompte->consulter();
        }
        ?>
    </div>
</div>
</body>
</html>
```

5.3 Exercice 3 : Classe Étudiant

Énoncé - Classe Étudiant

Objectif : Créer une classe `Etudiant` pour gérer les informations d'un étudiant.
Attributs :

- `nom` : nom de l'étudiant
- `prenom` : prénom de l'étudiant
- `matricule` : matricule de l'étudiant
- `note1, note2, note3` : trois notes sur 20

Méthodes à implémenter :

- `saisir()` : saisit toutes les informations
- `afficher()` : affiche toutes les informations
- `moyenne()` : calcule et retourne la moyenne des 3 notes
- `mention()` : retourne la mention selon la moyenne :
  - Moyenne ≥ 16 : "Très Bien"
  - Moyenne ≥ 14 : "Bien"
  - Moyenne ≥ 12 : "Assez Bien"
  - Moyenne ≥ 10 : "Passable"
  - Moyenne < 10 : "Ajourné"
- Getters et Setters

Travail à faire : Implémentez cette classe en PHP et en Java.

Solution Java - Classe Étudiant

```java
import java.util.Scanner;

public class Etudiant {
    // Attributs
    private String nom, prenom;
    private int matricule;
    private float note1, note2, note3;

    // Constructeur
    public Etudiant() {
        this.nom = "";
        this.prenom = "";
        this.matricule = 0;
        this.note1 = 0;
        this.note2 = 0;
        this.note3 = 0;
    }

    // M thodes
    public void saisir() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Nom : ");
        this.nom = sc.next();
        System.out.print("Prenom : ");
        this.prenom = sc.next();
        System.out.print("Matricule : ");
        this.matricule = sc.nextInt();
        System.out.print("Note 1 : ");
        this.note1 = sc.nextFloat();
        System.out.print("Note 2 : ");
        this.note2 = sc.nextFloat();
        System.out.print("Note 3 : ");
        this.note3 = sc.nextFloat();
    }

    public void afficher() {
        System.out.println("\n--- Informations Etudiant ---");
        System.out.println("Nom : " + this.nom);
        System.out.println("Prenom : " + this.prenom);
        System.out.println("Matricule : " + this.matricule);
        System.out.println("Note 1 : " + this.note1);
        System.out.println("Note 2 : " + this.note2);
        System.out.println("Note 3 : " + this.note3);
        System.out.printf("Moyenne : %.2f\n", this.moyenne());
        System.out.println("Mention : " + this.mention());
    }

    public float moyenne() {
        return (this.note1 + this.note2 + this.note3) / 3;
    }

    public String mention() {
        float moy = this.moyenne();

        if (moy >= 16) {
            return "Tr s Bien";
        } else if (moy >= 14) {
```

```
            return "Bien";
        } else if (moy >= 12) {
            return "Assez Bien";
        } else if (moy >= 10) {
            return "Passable";
        } else {
            return "Ajourn ";
        }
    }

    public void menu() {
        Scanner sc = new Scanner(System.in);
        int choix = 0;

        do {
            System.out.println("\n___ Menu Etudiant ___");
            System.out.println("1- Saisir les informations");
            System.out.println("2- Afficher les informations");
            System.out.println("3- Calculer la moyenne");
            System.out.println("4- Afficher la mention");
            System.out.println("0- Quitter");
            System.out.print("Votre choix : ");
            choix = sc.nextInt();

            switch(choix) {
                case 1: this.saisir(); break;
                case 2: this.afficher(); break;
                case 3:
                    System.out.printf("Moyenne : %.2f\n",
                                      this.moyenne());
                    break;
                case 4:
                    System.out.println("Mention : " +
                                       this.mention());
                    break;
                case 0:
                    System.out.println("Au revoir !");
                    break;
                default:
                    System.out.println("Erreur de saisie");
                    break;
            }
        } while(choix != 0);
    }

    // Getters et Setters
    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getPrenom() {
        return prenom;
    }

```

```
116     public void setPrenom(String prenom) {
117         this.prenom = prenom;
118     }
119 
120     public int getMatricule() {
121         return matricule;
122     }
123 
124     public void setMatricule(int matricule) {
125         this.matricule = matricule;
126     }
127 
128     public float getNote1() {
129         return note1;
130     }
131 
132     public void setNote1(float note1) {
133         this.note1 = note1;
134     }
135 
136     public float getNote2() {
137         return note2;
138     }
139 
140     public void setNote2(float note2) {
141         this.note2 = note2;
142     }
143 
144     public float getNote3() {
145         return note3;
146     }
147 
148     public void setNote3(float note3) {
149         this.note3 = note3;
150     }
151 }
```

Solution PHP - Classe Étudiant

Fichier etudiant.class.php :

```php
<?php
class Etudiant {
    // Attributs
    private $nom, $prenom;
    private $matricule;
    private $note1, $note2, $note3;

    // Constructeur
    public function __construct() {
        $this->nom = "";
        $this->prenom = "";
        $this->matricule = 0;
        $this->note1 = 0;
        $this->note2 = 0;
        $this->note3 = 0;
    }

    // M thodes
    public function saisir($tab) {
        $this->nom = $tab['nom'];
        $this->prenom = $tab['prenom'];
        $this->matricule = $tab['matricule'];
        $this->note1 = $tab['note1'];
        $this->note2 = $tab['note2'];
        $this->note3 = $tab['note3'];
    }

    public function afficher() {
        return "<h3>Informations Etudiant</h3>" .
               "<br>Nom : " . $this->nom .
               "<br>Prenom : " . $this->prenom .
               "<br>Matricule : " . $this->matricule .
               "<br>Note 1 : " . $this->note1 .
               "<br>Note 2 : " . $this->note2 .
               "<br>Note 3 : " . $this->note3 .
               "<br>Moyenne : " .
               number_format($this->moyenne(), 2) .
               "<br>Mention : " . $this->mention();
    }

    public function moyenne() {
        return ($this->note1 + $this->note2 +
                $this->note3) / 3;
    }

    public function mention() {
        $moy = $this->moyenne();

        if ($moy >= 16) {
            return "Tr s Bien";
        } elseif ($moy >= 14) {
            return "Bien";
        } elseif ($moy >= 12) {
            return "Assez Bien";
        } elseif ($moy >= 10) {
            return "Passable";
```

```
57          } else {
58              return "Ajourn ";
59          }
60      }
61
62      // Getters et Setters
63      public function getNom() {
64          return $this->nom;
65      }
66
67      public function setNom($nom) {
68          $this->nom = $nom;
69      }
70
71      public function getPrenom() {
72          return $this->prenom;
73      }
74
75      public function setPrenom($prenom) {
76          $this->prenom = $prenom;
77      }
78
79      public function getMatricule() {
80          return $this->matricule;
81      }
82
83      public function setMatricule($matricule) {
84          $this->matricule = $matricule;
85      }
86
87      public function getNote1() {
88          return $this->note1;
89      }
90
91      public function setNote1($note1) {
92          $this->note1 = $note1;
93      }
94
95      public function getNote2() {
96          return $this->note2;
97      }
98
99      public function setNote2($note2) {
100         $this->note2 = $note2;
101     }
102
103     public function getNote3() {
104         return $this->note3;
105     }
106
107     public function setNote3($note3) {
108         $this->note3 = $note3;
109     }
110 }
111 ?>
```

6 Concepts Avancés (Introduction)

L'Héritage

L'**héritage** permet à une classe (classe fille) de récupérer les attributs et méthodes d'une autre classe (classe mère).
Avantages :

- Réutilisation du code
- Organisation hiérarchique
- Spécialisation des classes

Exemple conceptuel :

- Classe mère : `Personne` (nom, prénom, âge)
- Classes filles : `Etudiant`, `Professeur`
- L'étudiant hérite de nom, prénom, âge et ajoute matricule, notes

Syntaxe :

- PHP : `class Etudiant extends Personne`
- Java : `public class Etudiant extends Personne`

Le Polymorphisme

Le **polymorphisme** permet à des objets de classes différentes de répondre différemment à un même message (méthode).
Exemple :

- La méthode `afficher()` se comporte différemment selon qu'elle est appelée sur un Rectangle, un Cercle ou un Triangle
- Chaque classe définit sa propre version de `afficher()`

Types :

- **Redéfinition** : Une classe fille redéfinit une méthode héritée
- **Surcharge** : Plusieurs méthodes avec le même nom mais des paramètres différents

L'Abstraction

L'**abstraction** consiste à définir des classes ou méthodes abstraites qui servent de modèles.
Caractéristiques :

- Une classe abstraite ne peut pas être instanciée
- Elle sert de base pour d'autres classes
- Peut contenir des méthodes abstraites (sans implémentation)

Exemple :

- Classe abstraite : `Forme` avec méthode abstraite `surface()`
- Classes concrètes : `Rectangle`, `Cercle` qui implémentent `surface()`

7 Bonnes Pratiques en POO

Principes SOLID

Les principes SOLID sont 5 règles fondamentales pour bien concevoir en POO :

1. **Single Responsibility (Responsabilité Unique)**
   - Une classe = une seule responsabilité
   - Facilite la maintenance et les tests
2. **Open/Closed (Ouvert/Fermé)**
   - Ouvert à l'extension, fermé à la modification
   - On ajoute des fonctionnalités sans modifier le code existant
3. **Liskov Substitution (Substitution de Liskov)**
   - Une classe fille doit pouvoir remplacer sa classe mère
4. **Interface Segregation (Ségrégation des Interfaces)**
   - Préférer plusieurs interfaces spécifiques à une interface générale
5. **Dependency Inversion (Inversion de Dépendance)**
   - Dépendre d'abstractions plutôt que d'implémentations concrètes

Conventions de Nommage

Classes :

- PascalCase : `Rectangle`, `CompteBancaire`
- Noms significatifs et descriptifs

Attributs et Méthodes :

- camelCase : `nomClient`, `calculerSurface()`
- Les attributs sont des noms
- Les méthodes sont des verbes

Constantes :

- MAJUSCULES : `TVA_TAUX`, `MAX_TENTATIVES`

Conseils Pratiques

1. **Toujours encapsuler :**
   - Attributs en `private`
   - Accès via getters/setters

2. **Documenter le code :**
   - Commentaires pour les classes et méthodes complexes
   - Expliquer le "pourquoi", pas le "comment"
3. **Tester régulièrement :**
   - Testez chaque méthode individuellement
   - Vérifiez les cas limites
4. **Éviter la duplication :**
   - DRY (Don't Repeat Yourself)
   - Factoriser le code répétitif
5. **Garder les méthodes courtes :**
   - Une méthode = une tâche
   - Facilite la lecture et la maintenance

8 Exercice Récapitulatif

Projet Final - Gestion de Bibliothèque

Objectif : Créer un système de gestion de bibliothèque simplifié.
Classe Livre :

- Attributs : `titre`, `auteur`, `isbn`, `disponible` (booléen)
- Méthodes :
  - `saisir()` : saisit les informations du livre
  - `afficher()` : affiche les informations
  - `emprunter()` : marque le livre comme non disponible
  - `retourner()` : marque le livre comme disponible
  - Getters et Setters

Contraintes :

- On ne peut emprunter que si le livre est disponible
- On ne peut retourner que si le livre n'est pas disponible
- L'ISBN doit être unique (validation simple)

Interface utilisateur :

- Menu interactif avec toutes les opérations
- Messages clairs pour l'utilisateur
- Gestion des erreurs

Bonus (optionnel) :

- Ajouter un attribut `dateEmprunt`
- Calculer les jours de retard
- Gérer plusieurs livres dans un tableau

9 Résumé et Checklist

Ce que vous devez retenir

Concepts fondamentaux :

- Différence entre classe et objet
- Rôle des attributs et méthodes
- Utilité du constructeur
- Principe d'encapsulation
- Utilisation des getters/setters

Syntaxe :

- Déclarer une classe en PHP et Java
- Créer un constructeur
- Instancier un objet
- Appeler des méthodes
- Accéder aux attributs via `$this` (PHP) ou `this` (Java)

Bonnes pratiques :

- Toujours mettre les attributs en private
- Nommer clairement classes, méthodes et attributs
- Commenter le code
- Tester chaque méthode
- Respecter le principe de responsabilité unique

Glossaire des Termes

Classe : Modèle ou plan qui définit la structure d'objets
Objet : Instance concrète d'une classe
Attribut : Variable qui stocke les données d'un objet
Méthode : Fonction définie dans une classe
Constructeur : Méthode spéciale appelée lors de la création d'un objet
Encapsulation : Principe de masquer les détails internes d'une classe
Getter : Méthode qui retourne la valeur d'un attribut
Setter : Méthode qui modifie la valeur d'un attribut
this / $this : Référence à l'objet courant
Instanciation : Action de créer un objet à partir d'une classe
Visibilité : Niveau d'accès (private, public, protected)
Héritage : Mécanisme permettant à une classe d'hériter d'une autre
Polymorphisme : Capacité de répondre différemment au même message

Erreurs Courantes à Éviter

1. **Oublier $this ou this**
   - Incorrect : `lg = 10;`
   - Correct : `$this->lg = 10;` (PHP) ou `this.lg = 10;` (Java)
2. **Accéder directement aux attributs privés**
   - Incorrect : `$monObjet->attributPrive;`
   - Correct : `$monObjet->getAttributPrive();`
3. **Oublier le mot-clé new**
   - Incorrect : `$obj = Rectangle();`
   - Correct : `$obj = new Rectangle();`
4. **Confondre -> et ::**
   - `->` : Accès aux membres d'un objet
   - `::` : Accès aux membres statiques d'une classe
5. **Ne pas initialiser dans le constructeur**
   - Toujours donner des valeurs par défaut aux attributs
6. **Méthodes trop longues**
   - Décomposer en plusieurs méthodes simples
7. **Noms de variables non descriptifs**
   - Incorrect : `$x, $a, $tmp`
   - Correct : `$longueur, $prixTotal, $compteur`

10 Exercices Supplémentaires

10.1 Exercice 4 : Classe Produit

Énoncé - Classe Produit

Objectif : Créer une classe `Produit` pour gérer un produit dans un magasin.
Attributs :

- `reference` : référence du produit
- `designation` : nom du produit
- `prixHT` : prix hors taxes
- `quantiteStock` : quantité en stock
- `tauxTVA` : taux de TVA (par défaut 20%)

Méthodes à implémenter :

- `saisir()` : saisit les informations du produit
- `afficher()` : affiche toutes les informations
- `calculerPrixTTC()` : retourne le prix TTC
- `calculerMontantTVA()` : retourne le montant de la TVA
- `ajouterStock(quantite)` : ajoute une quantité au stock
- `retirerStock(quantite)` : retire une quantité (si possible)
- `estDisponible()` : retourne vrai si quantité > 0
- Getters et Setters

Formules :

- Prix TTC = Prix HT $\times$ (1 + Taux TVA / 100)
- Montant TVA = Prix HT $\times$ Taux TVA / 100

Indications

Points d'attention :

- Vérifier que la quantité à retirer n'excède pas le stock
- Les quantités doivent être positives
- Le prix HT doit être positif
- Le taux de TVA doit être entre 0 et 100

Tests à effectuer :

- Créer un produit et afficher ses informations
- Calculer le prix TTC et la TVA

- Ajouter et retirer du stock
- Tenter de retirer plus que le stock disponible
- Vérifier la disponibilité

10.2 Exercice 5 : Classe Voiture

Énoncé - Classe Voiture

Objectif : Créer une classe `Voiture` pour gérer une voiture.
Attributs :

- `marque` : marque de la voiture
- `modele` : modèle de la voiture
- `annee` : année de fabrication
- `kilometrage` : kilométrage actuel
- `carburant` : niveau de carburant (0 à 100)
- `vitesse` : vitesse actuelle (initialisée à 0)

Méthodes à implémenter :

- `saisir()` : saisit les informations de la voiture
- `afficher()` : affiche toutes les informations
- `demarrer()` : démarre la voiture (si carburant > 0)
- `accelerer(vitesse)` : augmente la vitesse
- `freiner()` : réduit la vitesse à 0
- `faireLePlein()` : remplit le réservoir à 100
- `rouler(kilometres)` : ajoute des km et consomme du carburant
- `calculerAge()` : retourne l'âge de la voiture
- Getters et Setters

Règles :

- Consommation : 5 litres par 100 km
- On ne peut rouler que si carburant > 0
- La vitesse ne peut pas dépasser 200 km/h

11 Comparaison PHP vs Java

Différences Principales

Aspect	PHP	Java
Typage	Faible (dynamique)	Fort (statique)
Variables	Préfixe $ obligatoire	Pas de préfixe
Constructeur	__construct()	Nom de la classe
Référence objet	$this->	this.
Déclaration type	Optionnelle	Obligatoire
Instanciation	new Classe()	new Classe()
Visibilité par défaut	Public	Package (default)
Héritage	extends	extends
Interface	implements	implements
Fichiers	Plusieurs classes possibles	1 classe publique par fichier

Exemple Comparatif

PHP :

```php
<?php
class Personne {
    private $nom;

    public function __construct() {
        $this->nom = "";
    }

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

$p = new Personne();
$p->setNom("Dupont");
?>
```

Java :

```java
public class Personne {
    private String nom;

    public Personne() {
        this.nom = "";
    }

    public void setNom(String nom) {
        this.nom = nom;
    }
}

Personne p = new Personne();
p.setNom("Dupont");
```

12 Ressources et Pour Aller Plus Loin

Concepts à Approfondir

Une fois les bases maîtrisées, vous pourrez explorer :

1. **Héritage et Polymorphisme**
   - Classes mères et filles
   - Redéfinition de méthodes
   - Mot-clé `parent` (PHP) ou `super` (Java)
2. **Classes Abstraites et Interfaces**
   - Définir des contrats
   - Implémenter plusieurs interfaces
3. **Méthodes et Attributs Statiques**
   - Membres partagés par toutes les instances
   - Accès via le nom de la classe
4. **Exceptions et Gestion d'Erreurs**
   - try-catch
   - Créer ses propres exceptions
5. **Design Patterns**
   - Singleton
   - Factory
   - Observer
   - MVC (Model-View-Controller)
6. **Namespaces et Autoloading**
   - Organisation du code
   - Éviter les conflits de noms

Projet Pratique Suggéré

Système de Gestion d'École

Créez un système complet avec les classes suivantes :

- `Personne` (classe mère)
  - Attributs : nom, prénom, dateNaissance
- `Etudiant` (hérite de Personne)
  - Attributs : matricule, classe, notes[]

- Méthodes : calculerMoyenne(), passerEnClasseSuperieure()

- `Professeur` (hérite de Personne)
  - Attributs : matiere, salaire
  - Méthodes : donnerCours(), corrigerCopies()

- `Classe`
  - Attributs : nom, niveau, etudiants[], professeur
  - Méthodes : ajouterEtudiant(), afficherListeEtudiants()

Ce projet vous permettra de pratiquer l'héritage, la composition et la gestion de collections d'objets.

13 Mini Quiz d'Auto-évaluation

Testez vos Connaissances

Question 1 : Quelle est la différence entre une classe et un objet ?

Question 2 : Pourquoi déclare-t-on les attributs en `private` ?

Question 3 : À quoi sert un constructeur ?

Question 4 : Quelle est la différence entre un getter et un setter ?

Question 5 : En PHP, comment accède-t-on à un attribut dans une méthode de la classe ?

Question 6 : Peut-on créer plusieurs objets à partir d'une même classe ?

Question 7 : Qu'est-ce que l'encapsulation ?

Question 8 : Comment instancie-t-on un objet en Java ?

Question 9 : Quelle est la visibilité recommandée pour les attributs ?

Question 10 : Combien de constructeurs peut avoir une classe ?

Réponses

Réponse 1 : Une classe est un modèle/plan, un objet est une instance concrète de cette classe.
Réponse 2 : Pour protéger les données et contrôler l'accès via des méthodes (encapsulation).
Réponse 3 : À initialiser les attributs d'un objet lors de sa création.
Réponse 4 : Un getter retourne la valeur d'un attribut, un setter modifie sa valeur.
Réponse 5 : Avec `$this->nomAttribut`
Réponse 6 : Oui, autant qu'on veut. Chaque objet est indépendant.
Réponse 7 : Le principe de cacher les détails internes d'une classe et de n'exposer que ce qui est nécessaire.
Réponse 8 : `NomClasse objet = new NomClasse();`
Réponse 9 : `private`
Réponse 10 : Un seul en PHP, plusieurs possibles en Java (surcharge).

14 Conseils pour Réussir vos Projets POO

Méthodologie de Développement

1. **Analyser avant de coder**
   - Identifiez les entités (futures classes)
   - Listez leurs propriétés (attributs)
   - Définissez leurs comportements (méthodes)
   - Dessinez un diagramme de classes si nécessaire
2. **Commencer simple**
   - Créez d'abord la structure de base
   - Ajoutez les fonctionnalités une par une
   - Testez après chaque ajout
3. **Tester régulièrement**
   - Créez des objets de test
   - Vérifiez chaque méthode
   - Testez les cas limites
4. **Refactoriser le code**
   - Améliorez la structure sans changer le comportement
   - Éliminez les duplications
   - Simplifiez les méthodes complexes
5. **Documenter**
   - Commentez les parties complexes
   - Expliquez les choix de conception
   - Créez une documentation utilisateur

Exemple de Plan de Développement

Pour créer une classe Compte :
Étape 1 : Créer la classe vide avec les attributs

```
class Compte {
    private $nom, $prenom, $numero, $solde;
}
```

Étape 2 : Ajouter le constructeur

```
public function __construct() {
    $this->solde = 80;
}
```

Étape 3 : Ajouter une méthode à la fois et tester
Étape 4 : Ajouter les getters/setters
Étape 5 : Créer l'interface utilisateur
Étape 6 : Tester l'ensemble
Étape 7 : Améliorer et optimiser

Félicitations !

Vous avez terminé ce cours sur la POO

Continuez à pratiquer pour maîtriser ces concepts !

E LEARNING BTS – Septembre 2025