TP N°2

Classe et Objet

Préparée par : RGUIBI Farah

Objectif :

Ce travail pratique vise à approfondir les concepts de la Programmation Orientée Objet (POO) en Java. À travers la création de classes métiers représentant des entités du monde réel, nous développons les compétences nécessaires pour maîtriser l'encapsulation, les constructeurs, les méthodes d'instance, et la manipulation d'objets.

Ce TP permet de comprendre comment modéliser des concepts abstraits (nombres complexes, points géométriques) et concrets (articles commerciaux, livres, rectangles) en utilisant les principes fondamentaux de la POO. L'objectif final est de créer des classes réutilisables, bien structurées, et respectant les bonnes pratiques de développement Java.

FIGURE 1 – Java

1 Les exercices :

1.1 Exercice 3 : La classe livre

1.1.1 Idée de l'exercice :

Créer une classe **Livre** modélisant un livre de bibliothèque avec validation des données. Cet exercice introduit la notion de contrôle de l'intégrité des données lors de la modification des attributs, particulièrement pour le prix qui ne doit pas être négatif.

1.1.2 Approche adoptée :

J'ai développé une classe simple avec trois attributs : `Titre`, `Auteur` et `prix`. La classe propose un constructeur paramétré unique, des getters et setters pour tous les attributs. Le setter `setprix()` intègre une validation refusant les valeurs négatives, ce qui garantit la cohérence des données. La méthode `afficher()` présente toutes les informations du livre.

1.1.3 Points clés de l'implémentation :

- Encapsulation avec attributs privés de types String et double
- Constructeur paramétré pour initialisation complète à la création
- Getters simples retournant les valeurs des attributs
- Setters standards pour titre et auteur
- Setter `setprix()` avec **validation conditionnelle** : accepte uniquement prix >= 0
- Affichage d'un message "erreur" en cas de tentative d'assignation de prix négatif
- Méthode `afficher()` présentant titre, auteur et prix

1.1.4 Diagramme de la classe Livre :

Voilà le diagramme de la classe Livre :

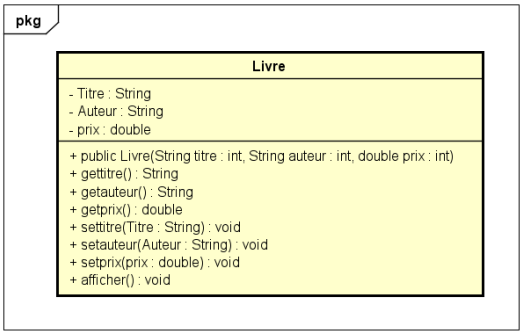

FIGURE 2 – Diagramme de la classe livre

1.1.5 Résultat :

Le programme crée et affiche correctement les informations du livre. La validation du prix assure qu'aucune valeur négative ne peut être assignée après la création de l'objet.

```java
package TP2;

public class Livre {
 private String Titre,Auteur;
 private double prix;

 public Livre(String titre,String auteur,double prix){
     this.Titre=titre;
     this.Auteur=auteur;
     this.prix=prix;
 }
    public String gettitre() {
        return Titre;}
    public String getauteur() {
        return Auteur;}
    public double getprix() {
        return prix;}
    public void settitre(String titre) {
        this.Titre=titre;}
    public void setauteur(String auteur) {
        this.Auteur=auteur;}
    public void setprix(double prix) {
        if(prix>=0) {
            this.prix=prix;
        }else {
            System.out.println("erreur");
        }}
    public void afficher() {
        System.out.println("le titre est " + Titre);
        System.out.println("l'auteur est " + Auteur);
        System.out.println("le prix est " + prix);
    }}
```

FIGURE 3 – La classe livre

```java
package TP2;

import java.util.Scanner;
public class MainLivre {

    public static void main (String []args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("donner le titre de livre");
        String titre=sc.next();
        System.out.println("donner l'auteur de livre ");
        String auteur=sc.next();
        System.out.println("donner le prix de livre");
        double prix=sc.nextDouble();
        Livre livre=new Livre(titre,auteur,prix);
        livre.afficher();
    }

}
```

FIGURE 4 – Main de la classe livre

```
donner le titre de livre
rguibi
donner l'auteur de livre
farah
donner le prix de livre
100000
le titre est rguibi
l'auteur est farah
le prix est 100000.0
```

FIGURE 5 – Resultat de la classe livre

1.2 Exercice 2 : La classe complexe

1.2.1 Idée de l'exercice :

Développer une classe **Complexe** représentant des nombres complexes de la forme z = a + bi, où a est la partie réelle et b la partie imaginaire. Cet exercice illustre comment modéliser des concepts mathématiques abstraits en POO et implémenter des opérations algébriques retournant de nouveaux objets.

1.2.2 Approche adoptée :

J'ai créé une classe avec deux attributs entiers : `reel` et `imaginaire`. La classe propose deux constructeurs (par défaut et paramétré), des getters et setters pour l'encapsulation, ainsi que deux méthodes d'opération : `plus()` pour l'addition et `Moins()` pour la soustraction. Ces méthodes créent et retournent un nouvel objet Complexe résultant de l'opération.

1.2.3 Points clés de l'implémentation :

- Encapsulation complète avec attributs privés et accesseurs publics
- Constructeur par défaut initialisant à 0 + 0i (élément neutre)
- Constructeur paramétré pour créer un complexe avec valeurs spécifiques
- Méthode `plus()` appliquant la règle : (a+bi) + (c+di) = (a+c) + (b+d)i
- Méthode `Moins()` appliquant la règle : (a+bi) - (c+di) = (a-c) + (b-d)i
- Les méthodes d'opération retournent un **nouvel objet** sans modifier les opérandes
- Méthode `afficher()` gérant l'affichage du signe selon la valeur de la partie imaginaire

1.2.4 Diagramme de la classe Complexe :

Voilà le diagramme de la classe Complexe :

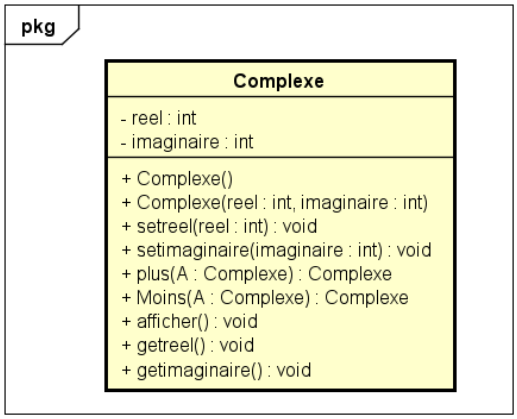

FIGURE 6 – Diagramme de la classe Complexe

1.2.5 Résultat :

Avec les entrées : z1 = 5 + 3i et z2 = 2 + 7i

Les calculs sont corrects : (5+3i) + (2+7i) = 7+10i et (5+3i) - (2+7i) = 3-4i. Le programme démontre la création d'objets retournés par les méthodes d'opération.

```java
package TP2;

public class Complexe {
    private int reel,imaginaire;

    public Complexe() {
        this.reel=0;
        this.imaginaire=0;}
    public Complexe(int reel,int imaginaire) {
        this.reel=reel;
        this.imaginaire=imaginaire;}
    public int getreel() {
        return reel;}
    public int getimaginaire() {
        return imaginaire;}
    public void setreel(int reel) {
        this.reel=reel;}
    public void setimaginaire(int imaginaire) {
        this.imaginaire=imaginaire;}
    public Complexe plus(Complexe A) {
        int re=this.reel+A.reel;
        int im =this.imaginaire+A.imaginaire;
        return new Complexe(re,im);}
    public Complexe Moins(Complexe A) {
        int re = this.reel - A.reel;
        int im = this.imaginaire - A.imaginaire;
        return new Complexe(re, im);}
    public void afficher() {
        if(reel >= 0) {
            System.out.println(reel +" +"+ imaginaire +"i");
        }else {
            System.out.println(reel +" - " + (-imaginaire) +"i");
        }
    }
```

FIGURE 7 – La classe complexe

```java
package TP2;

import java.util.Scanner;

public class MainComplexe {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Entrez la partie réelle du premier nombre : ");
        int re1 = sc.nextInt();
        System.out.println("Entrez la partie imaginaire du premier nombre : ");
        int im1 = sc.nextInt();
        Complexe z1 = new Complexe(re1, im1);

        System.out.println("Entrez la partie réelle du deuxième nombre : ");
        int re2 = sc.nextInt();
        System.out.println("Entrez la partie imaginaire du deuxième nombre : ");
        int im2 = sc.nextInt();
        Complexe z2 = new Complexe(re2, im2);

        System.out.print("Premier nombre complexe : ");
        z1.afficher();
        System.out.print("Deuxième nombre complexe : ");
        z2.afficher();

        Complexe somme =z1.plus(z2);
        System.out.println("Somme :");
        somme.afficher();

        Complexe moins =z1.Moins(z2);
        System.out.println("moins :");
        moins.afficher();
}}
```

FIGURE 8 – Main de la classe complexe

```
Entrez la partie réelle du premier nombre :
5
Entrez la partie imaginaire du premier nombre :
3
Entrez la partie réelle du deuxième nombre :
2
Entrez la partie imaginaire du deuxième nombre :
7
Premier nombre complexe : 5 +3i
Deuxième nombre complexe : 2 +7i
Somme :
7 +10i
moins :
3 +-4i
```

FIGURE 9 – Resultat de la classe complexe

1.3 Exercice 3 : La classe Rectangle

1.3.1 Idée de l'exercice :

Créer une classe **Rectangle** calculant les propriétés géométriques (périmètre, aire) et déterminant si le rectangle est un carré. Cet exercice combine calculs mathématiques, méthodes retournant différents types (double et boolean), et affichage conditionnel des résultats.

1.3.2 Approche adoptée :

J'ai développé une classe avec deux attributs : `Largeur` et `Longueur`. La classe propose deux constructeurs, des getters et setters, et quatre méthodes de calcul : `perimetre()` appliquant la formule 2(L+l), `aire()` calculant L×l, `EstCarre()` retournant un boolean selon l'égalité des dimensions, et `AfficherRectangle()` présentant toutes les informations avec un message conditionnel selon qu'il s'agisse d'un carré ou non.

1.3.3 Points clés de l'implémentation :

- Attributs de type double pour accepter des dimensions décimales
- Constructeur par défaut initialisant à zéro
- Constructeur paramétré pour créer un rectangle avec dimensions spécifiques
- Méthode `perimetre()` calculant 2 × (Longueur + Largeur)
- Méthode `aire()` calculant Longueur × Largeur
- Méthode `EstCarre()` retournant un **boolean** (Longueur == Largeur)
- Méthode `AfficherRectangle()` combinant affichage des données et des calculs
- Structure conditionnelle affichant "Il s'agit d'un carré" ou "Il ne s'agit pas d'un carré"

1.3.4 Diagramme de la classe Rectangle :

Voilà le diagramme de la classe Rectangle :

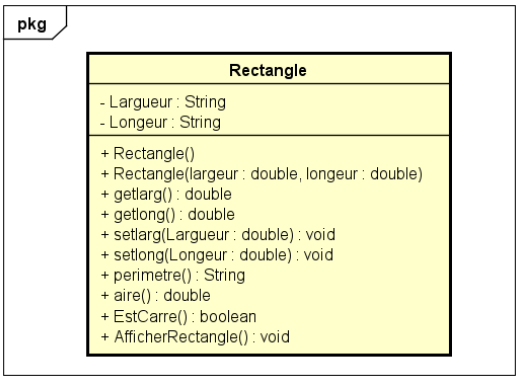

FIGURE 10 – Diagramme de la classe Rectangle

1.3.5 Résultat :

Cas 1 - Rectangle classique : largeur = 4.0, longueur = 9.0

Cas 2 - Carré : largeur = 7.0, longueur = 7.0

Les calculs sont corrects dans les deux cas. Pour le rectangle : périmètre = 2(9+4) = 26 et aire = 9×4 = 36. Pour le carré : périmètre = 2(7+7) = 28 et aire = 7×7 = 49. La détection du carré fonctionne correctement.

```java
package TP2;
import java.util.Scanner;
public class Rectangle {
    private double Largeur,Longueur;

    public Rectangle() {
        this.Largeur=0;
        this.Longueur=0;    }
    public Rectangle(double largeur,double longeur) {
        this.Largeur=largeur;
        this.Longueur=longeur;  }
    public double getlarg() {
        return Largeur;}
    public double getlong() {
        return Longueur;}
    public void setlarg(double largeur) {
        this.Largeur=largeur;}
    public void setlong(double longeur) {
        this.Longueur=longeur;}
    public double perimetre() {
        return 2*(Longueur+Largeur);}
    public double aire() {
        return Longueur*Largeur;}
    public boolean EstCarre() {
        return Longueur == Largeur;}
    public void AfficherRectangle() {
        System.out.println("la longeur du rectange est "+ Longueur);
        System.out.println("la Largeur du rectange est "+ Largeur);
        System.out.println("la perimetre du rectange est "+ perimetre());
        System.out.println("l'aire du rectange est "+ aire());
        if (EstCarre()) {
            System.out.println("Il s'agit d'un carré");
        } else {
            System.out.println("Il ne s'agit pas d'un carré");}}}
```

FIGURE 11 – La classe rectangle

```java
package TP2;

import java.util.Scanner;

public class MainRectangle {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("entrer la valeur de largeur");
        double largeur=sc.nextDouble();
        System.out.println("entrer la valeur de longueur");
        double longueur=sc.nextDouble();
        Rectangle R=new Rectangle(largeur,longueur);
        R.AfficherRectangle();

    }
}
```

FIGURE 12 – Main de la classe rectangle

```
entrer la valeur de largeur
4
entrer la valeur de longueur
9
la longeur du rectange est 9.0
la Largeur du rectange est 4.0
la perimetre du rectange est 26.0
l'aire du rectange est 36.0
Il ne s'agit pas d'un carré
```

FIGURE 13 – Resultat de la classe rectangle

1.4 Exercice 4 : La classe point

1.4.1 Idée de l'exercice :

Développer une classe **Point** représentant un point dans un plan cartésien 2D avec calcul de sa distance à l'origine. Cet exercice illustre l'application de formules mathématiques et l'utilisation de la classe Math de Java pour les calculs scientifiques.

1.4.2 Approche adoptée :

J'ai créé une classe avec deux attributs entiers : `x` et `y` représentant les coordonnées du point. La classe propose deux constructeurs (par défaut initialisant à l'origine, et paramétré), des getters et setters pour l'encapsulation. La méthode principale `Norme()` calcule la distance euclidienne du point à l'origine en appliquant la formule ($x^2 + y^2$) grâce aux méthodes `Math.pow()` et `Math.sqrt()`.

1.4.3 Points clés de l'implémentation :

- Modélisation d'un point géométrique avec coordonnées entières
- Constructeur par défaut plaçant le point à l'origine (0, 0)
- Constructeur paramétré pour créer un point à des coordonnées spécifiques
- Getters et setters permettant la modification des coordonnées après création
- Méthode `Norme()` retournant un double (résultat de la racine carrée)
- Utilisation de `Math.pow(x, 2)` pour calculer les carrés
- Utilisation de `Math.sqrt()` pour calculer la racine carrée

1.4.4 Diagramme de la classe Point :

Voilà le diagramme de la classe Point :

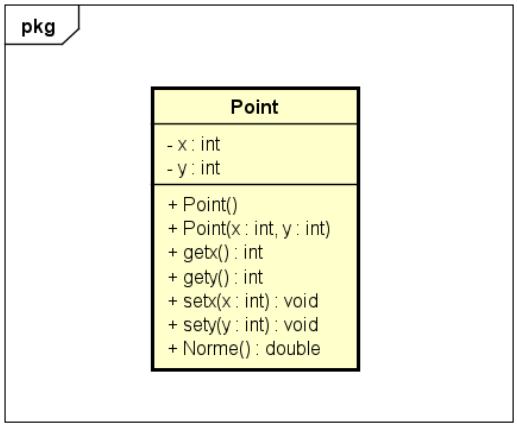

FIGURE 14 – Diagramme de la classe Point

1.4.5 Résultat :

Avec les entrées : x = 6 et y = 8

Le calcul est correct : $(6^2 + 8^2) = (36 + 64) = 100 = 10.0$. C'est un triplet pythagoricien classique (6, 8, 10) qui confirme la validité de l'implémentation.

```java
package TP2;
import java.util.Scanner;

public class Point {

private int x,y;

    public Point() {
        this.x=0;
        this.y=0;
    }
    public Point(int x,int y) {
        this.x=x;
        this.y=y;
    }
    public int getx() {return x;}
    public int gety() {return y;}
    public void setx(int x) {this.x=x;}
    public void sety(int y) {this.y=y;}

    public double Norme() {
            return Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2));
        }

}
```

FIGURE 15 – La classe point

```java
package TP2;

import java.util.Scanner;

public class MainPoint {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("entrer la valeur de x");
        int x=sc.nextInt();

        System.out.println("entrer la valeur de y");
        int y=sc.nextInt();
        Point p=new Point(x,y);
        System.out.println("La norme du point est : " + p.Norme());
    }

}
```

FIGURE 16 – Main de la classe point

```
entrer la valeur de x
6
entrer la valeur de y
8
La norme du point est : 10.0
```

FIGURE 17 – Resultat de la classe point

1.5 Exercice 5 : La classe article

1.5.1 Idée de l'exercice :

Créer une classe **Article** qui modélise un article commercial avec calcul automatique du prix TTC (Toutes Taxes Comprises). Cet exercice introduit la notion d'attribut statique partagé entre toutes les instances, ainsi que l'utilisation de plusieurs constructeurs pour différents scénarios d'initialisation.

1.5.2 Approche adoptée :

J'ai développé une classe avec quatre attributs : `reference`, `designation`, `prixHT` et `tauxTVA` (statique). La classe propose quatre constructeurs différents pour offrir une flexibilité maximale lors de la création d'objets. La méthode `calculprixttc()` applique la formule mathématique du calcul de TVA, et la méthode `afficher()` présente toutes les informations de l'article.

1.5.3 Points clés de l'implémentation :

- Déclaration de l'attribut `tauxTVA` comme **static** pour qu'il soit partagé par tous les articles
- Création d'un constructeur par défaut initialisant les valeurs à zéro ou chaînes vides
- Création d'un constructeur complet prenant tous les paramètres
- Création d'un constructeur partiel pour référence et désignation uniquement
- Création d'un constructeur par copie pour dupliquer un article existant
- Méthode `calculprixttc()` appliquant la formule : **PrixTTC = PrixHT + (PrixHT × TauxTVA / 100)**
- Méthode `afficher()` présentant référence, désignation, prix HT et prix TTC

1.5.4 Diagramme de la classe Article :

Voilà le diagramme de la classe Article :

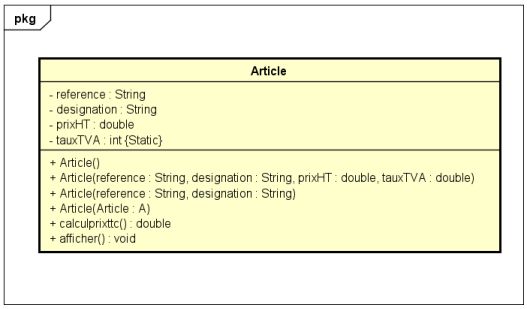

FIGURE 18 – Diagramme de la classe Article

1.5.5 Résultat :

Avec les entrées suivantes : référence = "arar", désignation = "Laptop", prix HT = 8000, taux TVA = 20

Le programme calcule correctement le prix TTC : 8000 + (8000 × 20/100) = 8000 + 1600 = 9600 DH. L'attribut statique `tauxTVA` permet d'appliquer le même taux à tous les articles créés.

```java
package TP2;

public class Article {

    private String reference;
    private String designation;
    private double prixHT;
    private static double tauxTVA;

    public Article() {
        this.reference = "";
        this.designation = "";
        this.prixHT = 0.0;}
    public Article(String reference, String designation, double prixHT, double tauxTVA) {
        this.reference = reference;
        this.designation = designation;
        this.prixHT = prixHT;
        Article.tauxTVA = tauxTVA;}

    public Article(String reference, String designation) {
        this.reference = reference;
        this.designation = designation;
        this.prixHT = 0.0;}

    public Article(Article A) {
        this.reference = A.reference;
        this.designation = A.designation;
        this.prixHT = A.prixHT;}
    public double  calculprixttc() {
        return prixHT+(prixHT*tauxTVA/100);}
    public void afficher() {
        System.out.println("la reference est = "+ reference);
        System.out.println("la designation est = "+ designation);
        System.out.println("le prixHT est = "+ prixHT);
        System.out.println("le prixttc est = "+ calculprixttc());}}
```

FIGURE 19 – La classe article

```java
package TP2;

import java.util.Scanner;
public class MainArticle {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("donner la reference");
        String reference =sc.next();
        System.out.println("donner la designation");
        String designation =sc.next();
        System.out.println("donner le prix HT");
        int prixht=sc.nextInt();
        System.out.println("donner le taux tva pour tous les articles");
        double tauxtva=sc.nextDouble();
        Article article=new Article(reference,designation,prixht,tauxtva);
        article.afficher();

    }

}
```

FIGURE 20 – Main de la classe article

```
donner la reference
arar
donner la designation
laptop
donner le prix HT
8000
donner le taux tva pour tous les articles
20
la reference est = arar
la designation est = laptop
le prixHT est = 8000.0
le prixttc est = 9600.0
```

FIGURE 21 – Resultat de la classe article

2 Synthèse

Ce travail pratique m'a permis de maîtriser les concepts fondamentaux de la Programmation Orientée Objet en Java :

- **Encapsulation** : Protection des données avec attributs privés et méthodes publiques d'accès
- **Constructeurs multiples** : Offrir différentes façons d'initialiser les objets selon les besoins
- **Méthodes d'instance** : Définir des comportements spécifiques à chaque objet
- **Attributs statiques** : Partager des données communes entre toutes les instances d'une classe
- **Validation des données** : Contrôler l'intégrité des informations via les setters
- **Retour d'objets** : Créer et retourner de nouveaux objets dans les méthodes
- **Méthodes de calcul** : Implémenter des formules mathématiques et géométriques
- **Utilisation de la classe Math** : Exploiter les fonctions mathématiques avancées de Java