CHELLE Raymond 01/12/2025

Rapport de TP :

Sommaire :

I. Introduction

Dans le cadre de notre apprentissage du langage Java, nous cherchons à comprendre en profondeur le fonctionnement et la structure d'une classe. Pour cela, nous allons réaliser deux exercices pratiques centrés sur la création et la manipulation de classes. Ces exercices seront effectués à l'aide des environnements de développement BlueJ ou Eclipse, qui offrent des outils visuels et interactifs facilitant l'observation du comportement des objets, la compréhension des concepts de base tels que les attributs, les méthodes, les constructeurs, ainsi que les notions d'encapsulation et d'accès aux données.

L'objectif de cette démarche est de mettre en pratique les notions théoriques vues en cours, notamment la définition d'une classe, l'utilisation des getters et setters, la gestion des constructeurs et la création d'interactions entre plusieurs objets. À travers ces exercices, nous pourrons consolider notre compréhension du paradigme orienté objet, tout en nous familiarisant avec des outils professionnels de développement Java. Ce compte rendu détaillera les étapes suivies, les principes mis en œuvre et les conclusions tirées de cette exploration.

II. Le travail à réaliser

Exercice 1 :

1) Pour définir une classe *Livre* accessible depuis n'importe quel autre package du projet, il faut la déclarer avec le mot-clé **public**.
A l'intérieur de celle-ci il faut déclarer les attributs Titre, Auteur et Prix dans un accès privé :

```
public class Livre {
    private String titre;
    private String auteur;
    private int prix;
```

2) Pour que l'utilisateur puisse faire des entrées, il faut importer le Scanner avec ce code : import java.util.Scanner;
Ensuite, nous ajoutons le code du Scanner :

```
public Livre() {
    Scanner sc = new Scanner(System.in);
```

Et nous attribuons les entrées utilisateurs aux constructeurs :

```
public Livre() {
    Scanner sc = new Scanner(System.in);

    System.out.print("Entrez le titre du livre : ");
    this.titre = sc.nextLine();

    System.out.print("Entrez l'auteur du livre : ");
    this.auteur = sc.nextLine();

    System.out.print("Entrez le prix du livre (en euros) : ");
    this.prix = sc.nextInt();
}
```

3) Pour permettre l'accès aux différents attributs, il est nécessaire de créer un *getter* et un *setter* pour chacun d'eux.
Un getter renvoie la valeur de l'attribut (return this.nomAttribut), tandis qu'un setter permet de la modifier (this.nomAttribut = nomAttribut).

Nous allons donc créer ces méthodes pour **Titre**, **Auteur** et **Prix**. Nous ferons en sorte que les setters ne renvoient aucune valeur (grâce au mot-clé void) et qu'ils reçoivent en paramètre la nouvelle valeur destinée à l'attribut correspondant

```
public String getTitre() {
    return this.titre;
}

public String getAuteur() {
    return this.auteur;
}

public int getPrix() {
    return this.prix;
}


public void setTitre(String titre) {
    this.titre = titre;
}

public void setAuteur(String auteur) {
    this.auteur = auteur;
}

public void setPrix(int prix) {
        this.prix = prix;
    }
}
```

4)

```
public void Afficher() {
    System.out.println("Titre du livre en cours : " + this.titre);
    System.out.println("Auteur du livre en cours : " + this.auteur);
    System.out.println("Prix du livre en cours : " + this.prix);

}
```

5) Pour cela, nous allons utiliser les setters et faire appel à la classe *Afficher* afin d'afficher l'auteur, le titre et le prix.

```
public static void main(String[] args) {
    Livre monLivre = new Livre();


    monLivre.setTitre("1984");
    monLivre.setAuteur("George Orwell");
    monLivre.setPrix(20);


    monLivre.Afficher();
}
```

Exercice 2 :

1) Définition d'une classe *Client* contenant les attributs suivants : **CIN**, **Nom**, **Prénom** et **Téléphone**

```
public class Client {
    private String cin;
    private String nom;
    private String prenom;
    private String tel;
```

2) Définir, à l'aide des propriétés, les méthodes d'accès (getters et setters) aux différents attributs de la classe.

```
public String getCin() {
    return cin;
}

public String getNom() {
    return nom;
}

public String getPrenom() {
    return prenom;
}

public String getTel() {
    return tel;
}

public void setCin(String cin) {
    this.cin = cin;
}

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

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

public void setTel(String tel) {
    this.tel = tel;
}
```

3) Définir un constructeur qui initialise l'ensemble des attributs de la classe.

```
public Client(String cin, String nom, String prenom, String tel) {
    this.cin = cin;
    this.nom = nom;
    this.prenom = prenom;
    this.tel = tel;
}
```

4) Définissons un constructeur permettant d'initialiser le **CIN**, le **nom** et le **prénom**

```
public Client(String cin, String nom, String prenom) {
    this.cin = cin;
    this.nom = nom;
    this.prenom = prenom;
}
```

5) Définissons la méthode *Afficher()* afin qu'elle affiche les informations du client courant.

```
public void Afficher() {
    System.out.println("CIN: " + this.cin);
    System.out.println("Nom: " + this.nom);
    System.out.println("Prénom: " + this.prenom);
    System.out.println("Tél: " + this.tel );
}
```

6) Créons une classe *Compte* caractérisée par son solde, par un code qui s'incrémente automatiquement à chaque nouvelle création de compte, ainsi que par son propriétaire, représenté par un objet de type *Client*.

On initialise le dernier code à 0 et on le déclare en *static* afin qu'il soit partagé entre toutes les instances de la classe.

```
class Compte {
    private static int dernierCode = 0;
    private int code;
    private int solde;
    private Client proprietaire;
}
```

L'incrémentation du code devra ensuite être effectuée lors de la création d'un nouveau compte, ce qui signifie qu'elle doit être placée dans le constructeur. Cependant, cela n'est pas encore possible à ce stade, donc cette étape sera ajoutée dans l'exercice 8

7) Définissons, à l'aide de propriétés, les méthodes d'accès aux différents attributs de la classe, uniquement en lecture (donc sans setters).

```
public int getCode() {
    return code;
}

public int getSolde() {
    return solde;
}

public Client getProprietaire() {
    return proprietaire;
}
```

8) Définissons un constructeur permettant de créer un compte en spécifiant son propriétaire.

```
public Compte(Client proprietaire) {
    dernierCode = dernierCode+1;
    this.code = dernierCode;
    this.solde = 0;
    this.proprietaire = proprietaire;
    System.out.println("Compte n°" + this.code + " créé pour " + proprietaire.getPrenom() + " " + proprietaire.getNom());
}
```

III. Conclusion

Ce travail pratique nous a permis de mieux comprendre les principes fondamentaux de la programmation orientée objet en Java. En réalisant les exercices, nous avons appris à créer et structurer des classes, à gérer les attributs et leurs méthodes d'accès, ainsi qu'à utiliser des constructeurs pour initialiser correctement les objets. L'utilisation de BlueJ ou Eclipse a facilité l'observation du fonctionnement des classes et des interactions entre objets.

Au final, ce TP nous a aidés à relier la théorie à la pratique et à renforcer nos bases en Java, ce qui sera essentiel pour la suite de notre apprentissage.