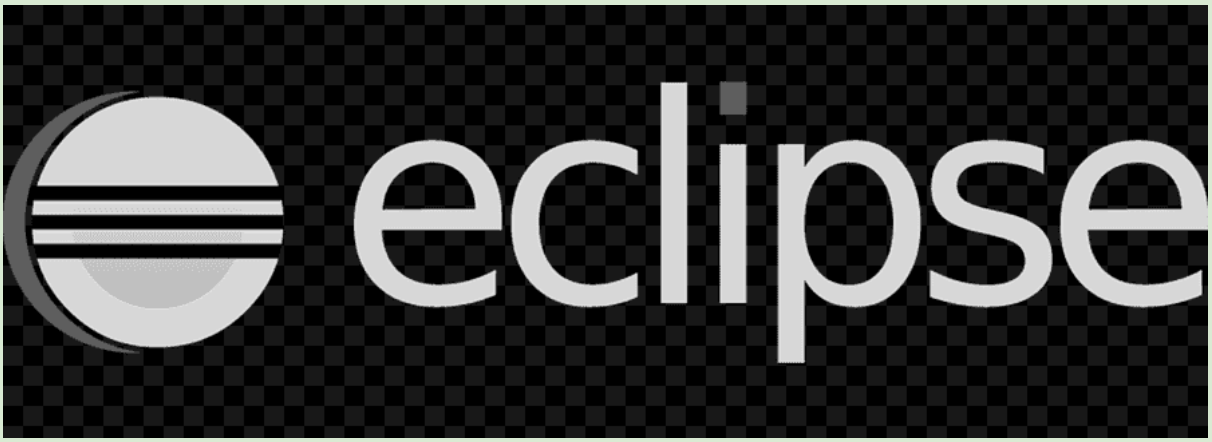
Compte Rendu TP - Les Classes (Livre et Carte Bancaire)



Introduction :
Dans ce TP nous allons devoir créer un programme concernant les livres et un système bancaire avec comptes clients. L'objectif est de maîtriser les concepts fondamentaux de la POO.

Exercice 1 : La Classe Livre

Question 1

Nous allons d'abord devoir d'abord définir une classe "Livre" avec les attributs (titre, auteur et prix).

```java
public class Livre {
    // Attributs
    private String titre;
    private String auteur;
    private double prix;
    private static Scanner scanner = new Scanner(System.in);
```

Question 2

Nous allons ensuite définir un constructeur permettant l'initialisation des attributs entrés par l'utilisateur.

```java
// Constructeur
public Livre() {
    System.out.print("Donner le titre: ");
    this.titre = scanner.nextLine();

    System.out.print("Donner l'auteur: ");
    this.auteur = scanner.nextLine();

    System.out.print("Donner le prix: ");
    this.prix = scanner.nextDouble();
    scanner.nextLine();
}
```

Question 3

Nous définirons les méthodes d' accès aux différents attributs grâce aux "getters" et aux "setters"

```java
// Getters et Setters
public String getTitre() {
    return titre;
}

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

public String getAuteur() {
    return auteur;
}

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

Question 4

Nous allons afficher toutes les infos concernant le livre en rappelant les attributs du livre vu précédemment.

```java
// Méthode Afficher
public void Afficher() {
    System.out.println("Titre: " + titre + ", Auteur: " + auteur + ", Prix: " + prix);
}
```

Question 5

Dans ce programme nous rappeler les classes que nous avons faites précédemment, il rappelle l'auteur, le titre et le prix.

```java
// Programme principal de test
public static void main(String[] args) {
    // Test avec livre
    System.out.println("Livre 1:");
    Livre livre = new Livre();

    // Test des getters
    System.out.println("Le titre est " + livre.getTitre());
    System.out.println("L'auteur est " + livre.getAuteur());
    System.out.println("Le prix est " + livre.getPrix());

    // Affichage avec la méthode Afficher
    livre.Afficher();
```

```
Livre 1:
Donner le titre: CCCCCC
Donner l'auteur: kiloo
Donner le prix: 20
Le titre est CCCCCC
L'auteur est kiloo
Le prix est 20.0
Titre: CCCCCC, Auteur: kiloo, Prix: 20.0
```

Exercice 2 : La Classe Compte

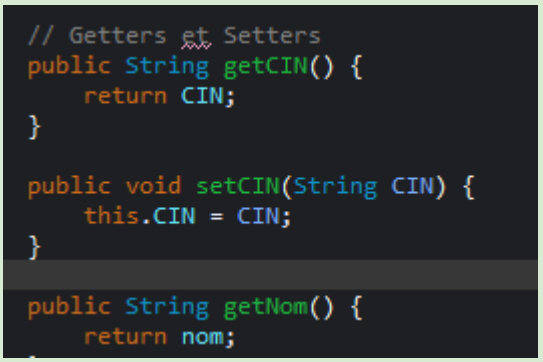
```java
public class exo2 {
        // Classe Client interne
        static class Client {
            private String CIN;
            private String nom;
            private String prenom;
            private String tel;
```

Question 1

On déclare 4 attributs qui seront présent dans la classe Client: CIN, Nom, Prénom, Tél.

Question 2

On permet l'accès contrôlé aux attributs privés depuis l'extérieur de la classe en implémentant des getters et des setters .



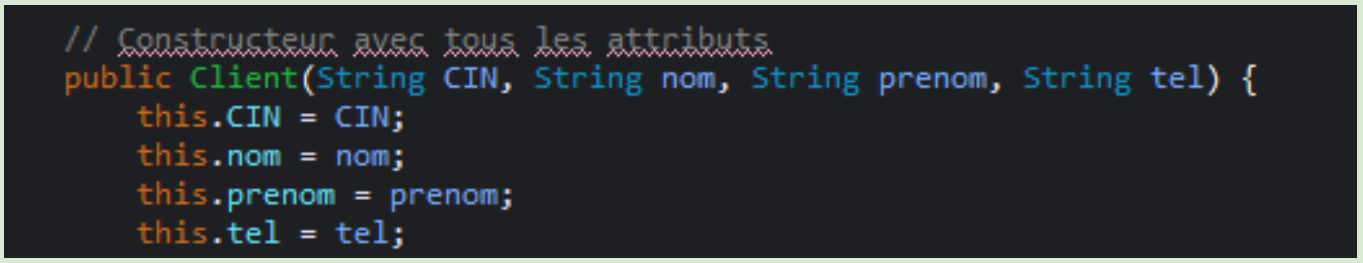
```java
// Getters et Setters
public String getCIN() {
    return CIN;
}

public void setCIN(String CIN) {
    this.CIN = CIN;
}

public String getNom() {
    return nom;
```

Question 3

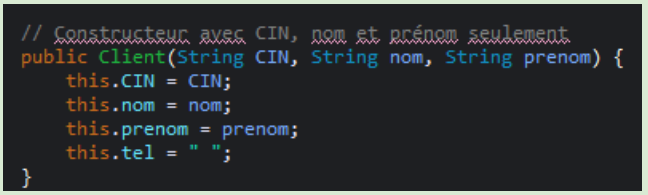
On crée un constructeur qui prend 4 paramètres et on utilise this.attribut, de plus il permet l'initialisation des attributs.



```java
// Constructeur avec tous les attributs
public Client(String CIN, String nom, String prenom, String tel) {
    this.CIN = CIN;
    this.nom = nom;
    this.prenom = prenom;
    this.tel = tel;
```

Question 4

On crée une alternative pour créer un client sans le numéro de téléphone en créant un autre constructeur, avec seulement 3 paramètres (CIN, nom, prénom), le téléphone est initialisé avec une valeur par défaut.



```java
// Constructeur avec CIN, nom et prénom seulement
public Client(String CIN, String nom, String prenom) {
    this.CIN = CIN;
    this.nom = nom;
    this.prenom = prenom;
    this.tel = " ";
}
```

Question 5

On doit afficher les informations du client.

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

Question 6

On doit créer la structure pour représenter un compte bancaire :

```java
// Classe Compte
static class Compte {
    private static int compteurCode = 0;

    private int code;
    private double solde;
    private Client proprietaire;
```

Question 7

On restreint l'accès en écriture pour certains attributs sensibles, on met un getter pour le code et le solde mais pas de setter, cela empêche la modification directe du solde et du numéro de compte.

```java
// Getters (lecture seule pour code et solde)
public int getCode() {
    return code;
}

public double getSolde() {
    return solde;
}

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

```java
    // Constructeur
    public Compte(Client proprietaire) {
        compteurCode++;
        this.code = compteurCode;
        this.solde = 0;
        this.proprietaire = proprietaire;
    }
```

Question 8

On initialise un nouveau compte avec son propriétaire.

Question 9

- Créditer(somme en paramètre)
  Ajoutez de l'argent au compte :
  Vérifie que le montant est positif puis l'ajoute au solde.
- Créditer(somme et compte en paramètre)
  Effectuer un virement entrant depuis un autre compte :
  Débite le compte source et crédite le compte courant.
- Débiter(somme en paramètre)
  Retirer de l'argent du compte :
  Vérifie le solde suffisant et la validité du montant avant retrait.
- Débiter(somme et compte en paramètre)
  Effectue un virement sortant vers un autre compte :
  Débite le compte courant et crédite le compte à destination.

```java
// Méthode Crediter avec montant et compte source
public void Crediter(double montant, Compte compteSource) {
    if (montant > 0) {
        compteSource.Debiter(montant);
        this.solde += montant;
    } else {
        System.out.println("Montant invalide");
    }
}

// Méthode Debiter avec un montant
public void Debiter(double montant) {
    if (montant > 0 && montant <= this.solde) {
        this.solde -= montant;
        System.out.println("Opération bien effectuée");
    } else if (montant <= 0) {
        System.out.println("Montant invalide");
    } else {
        System.out.println("Solde insuffisant");
    }
}
```

Question 10

Ce programme crée un compte client, effectue des dépôts et retraits, puis affiche les soldes après chaque opération. Il valide ainsi le bon fonctionnement de toutes les méthodes implémentées dans un scénario pratique.

```java
// Création du premier compte
System.out.println("Compte 1:");
Client client1 = creerClient(scanner);
Compte compte1 = new Compte(client1);

System.out.println("Détails du compte:");
compte1.AfficherResume();

// Test de dépôt
System.out.print("Donner le montant à déposer : ");
double montantDepot = scanner.nextDouble();
compte1.Crediter(montantDepot);

System.out.println("Après dépôt:");
compte1.AfficherResume();

// Test de retrait
System.out.print("Donner le montant à retirer :");
double montantRetrait = scanner.nextDouble();
compte1.Debiter(montantRetrait);


System.out.println("Après retrait:");
compte1.AfficherResume();

scanner.close();
```

Conclusion:

Ce TP de programmation m’a permis de concevoir et implémenter des classes Java modélisant des livres et un système bancaire. J’ai pu appliquer des principes fondamentaux comme les constructeurs et les méthodes.