TP N° : 2 Concept de l'orienté objet

Exercice 1 :

1. Créer un projet de type Java Project avec le nom **GestionEtudiants**.
2. Créer la classe **Etudiant** qui possède les attributs suivants :
   a. **Code** : int
   b. **Nom** : String
   c. **Prenom** : String
3. Ajouter les constructeurs suivants :
   a. Un constructeur par défaut
   b. Un constructeur d'initialisation qui accepte comme paramètre Nom.
   c. Un constructeur d'initialisation qui accepte comme paramètre Code, Nom et Prenom.
   d. Un constructeur par copie.
4. Ajouter les méthodes suivantes :
   a. **ToString()** : méthode qui retourne une chaine de caractères sous le format : **l'Etudiant dont le code est : « Code » son Nom est : « Nom », son Prénom est « Prénom ».**
5. Créer une autre classe **Test**, dans la méthode Main :
   a. Instancier 3 objets de la classe Etudiant. Utiliser pour chaque objet un constructeur différent.
   b. Demander à l'utilisateur de saisir les informations manquantes des étudiants créés par le constructeur par défaut, ainsi que par le deuxième qui accepte le Nom comme paramètre.
   c. Instancier un quatrième objet en utilisant le constructeur par copie.
   d. Afficher les informations de chaque objet.

Exercice 2 :

Voici le texte d'une classe représentant de façon sommaire un compte bancaire et les opérations bancaires courantes.

```
Class Compte {
    Int solde=0 ;
    Void deposer(int montant){
    Solde =solde+montant ;
    }
    Void retirer (int montant){
    Solde=solde-montant ;
    }
    Void virerVers(int montant, Compte destination){
    this.retirer(montant) ;
    Destination.deposer(montant) ;
    }
    Void afficher(){
    System.out.println('solde : '+solde) ; }
}
```

1- Comment fonctionne la méthode **virerVers**? Combien de comptes fait-elle intervenir? (répondez dans le code sous forme de commentaire)
2- Créez deux comptes que vous affecterez à deux variables.
3- Ecrivez le code correspondant aux opérations suivantes :
    a. dépot de 500 euros sur le premier compte.
    b. dépot de 1000 euros sur le second compte.
    c. retrait de 10 euros sur le second compte.
    d. virement de 75 euros du premier compte vers le second.
    e. affichage des soldes des deux comptes.
    f. Vous mettrez le code java correspondant à cette question dans la méthode main d'une nouvelle classe appelée **TestCompte**. Vous compilerez et testerez ce programme.
4- Complétez la classe Compte avec une information supplémentaire : le nom du titulaire du compte (type String). Modifiez la méthode d'affichage pour qu'elle affiche cette information.
5- Créez un constructeur pour la classe Compte. Ce constructeur doit prendre en paramètre le nom du titulaire du compte. Donnez le code de création d'un compte qui appelle ce constructeur.
6- Modifiez la méthode **retirer** pour empêcher le retrait quand le compte n'est pas suffisamment approvisionné.

Exercice 3 :

Créez une classe Ratio qui implémente les nombres rationnels. Pour représenter un nombre rationnel on utilise deux attributs entiers **numera** et **denomi** qui seront respectivement le numérateur et le dénominateur d'un rationnel. De plus le dénominateur doit être toujours différent de 0. Implémentez les méthodes publiques suivantes :

- Ratio(int numera , int denomi)
- Ratio produit(Ratio a)
- Ratio addition(Ratio a)
- boolean egale(Ratio a)
- boolean plusGrand(Ratio a)
- String toString()

Les méthodes produit et addition renvoient respectivement le produit et la somme de deux rationnels. La méthode **egale** renvoie true si les deux rationnels sont égaux (et renvoie false sinon) et la méthode **plusGrand** renvoie true si le rationel a est strictement plus grand que this. La méthode **toString** renvoie une représentation du rationnel, par exemple 13/7. Nous rappelons que a/b + c/d = (ad+bc)/bd , a/b × c/d = ac/bd , que a/b < c/d si et seulement si ad < bc et a/b = c/d si et seulement si ad = bc.

Testez votre classe.

Exercice 4 :

On souhaite disposer d'une classe permettant d'effectuer des conversions (dans les deux sens) entre nombre sexagésimaux (durée exprimée en heures, minutes, secondes) et des nombres décimaux (durée exprimée en heures décimales). Pour ce faire, on réalisera une classe permettant de représenter une durée. Elle comportera :

- un constructeur recevant trois arguments de type int représentant une valeur sexagésimale (heures, minutes, secondes).
- Un constructeur recevant un argument de type double représentant une durée en heures ;
- Une méthode getDec fournissant la valeur en heures décimales associée à l'objet,
- Des méthodes getH, getM et getS fournissant les trois composantes du nombre sexagésimal associé à l'objet.

Proposez deux solutions :

1. Avec un champ (privé) représentant la valeur décimale,
2. Avec des champs (privés) représentant la valeur sexagésimale

Testez votre programme (les deux solutions).