ENSA de Kenitra
Module : Programmation Java
2020/2021
Série 1 : Basic/Classe/Objet & UML

Exercice 1 :

Partie 1

Réaliser une classe ***Point*** permettant de représenter un point sur un axe.

Chaque point sera caractérisé par un nom (de type char) et une abscisse (de type double). On prévoira :

- un constructeur recevant en arguments le nom et l'abscisse d'un point,
- une méthode affiche imprimant (en fenêtre console) le nom du point et son abscisse,
- une méthode translate effectuant une translation définie par la valeur de son argument.

1. Écrire un petit programme utilisant cette classe pour créer un point, en afficher les caractéristiques, le déplacer et en afficher à nouveau les caractéristiques.

2. Ajouter un constructeur par copie, adapter le main.

Partie 2

Soit la nouvelle classe point :

```
class Point
{ public Point (int abs, int ord)      { x = abs ;  y = ord ; }
  public void deplace (int dx, int dy) { x += dx ;  y += dy ; }
  public void affiche ()
  { System.out.println ("Je suis un point de coordonnees " + x + " " + y) ;
  }
  private double x ;   // abscisse
  private double y ;   // ordonnee
}
public class TstPnt
{ public static void main (String args[])
  { Point a ;
    a = new Point(3, 5) ;        a.affiche() ;
    a.deplace(2, 0) ;            a.affiche() ;
    Point b = new Point(6, 8) ;  b.affiche() ;
  }
}
```

1. Modifier la définition de la classe Point dans la partie 1 et en supprimant la méthode affiche et en introduisant deux méthodes d'accès nommées abscisse et ordonnée fournissant respectivement l'abscisse et l'ordonnée d'un point. Adapter la méthode main en conséquence.
2. Ajouter un constructeur par copie, adapter la méthode main.

3. Lui ajouter une méthode maxNorme déterminant parmi deux points lequel est le plus éloigné

de l'origine et le fournissant en valeur de retour. On donnera deux solutions :

• maxNorme est une méthode statique de Point,

• maxNorme est une méthode usuelle de Point.

4. Ajouter un attribut adresse (chaine de caractère) à la classe "Point", pour stocker l'adresse physique sur une carte, associés au point.

- Modifier les constructeurs

- Ajouter la méthode "localiser", pour afficher l'adresse du point.

Partie 3

Ajouter les fonctionnalités nécessaires à la classe Point, pour réaliser une classe Segment permettant de manipuler des segments d'un plan et disposant des méthodes suivantes :

- **segment (Point origine, Point extremite)**
- **segment (double xOr, double yOr, double xExt, double yExt)**
- **double longueur() ;**
- **void deplaceOrigine (double dx, double dy)**
- **void deplaceExtremite (double dx, double dy)**
- **void affiche()**

Exercice 2 :

1. Définir une classe Voiture avec les attributs suivants : Id, Marque, Vitesse, Puissance.
2. Définir un constructeur permettant d'initialiser les attributs d'un objet voiture par des valeurs passées en paramètre. Sachant que Id doit être auto-incrément.
3. Définir les accesseurs aux différents attributs de la classe.
4. Définir la méthode toString ( ) permettant d'afficher les informations d'une voiture.
5. Écrire un programme testant la classe Voiture.

Exercice 3 :

La classe Eleve posséde trois attributs privés :

- son nom, nommé nom, de type String,
- un ensemble de notes, nommé listeNotes, qui sont des entiers rangés dans un ArrayList<Integer>
- une moyenne de type double, nommée moyenne, qui doit toujours être égale à la moyenne des notes contenues dans l'attribut listeNotes. Un élève sans aucune note sera considéré comme ayant une moyenne nulle.

La classe Eleve possède un constructeur permettant uniquement d'inialiser le nom de l'élève.

La classe Eleve possède aussi cinq méthodes publiques :

- Un getter pour la moyenne de l'élève c'est-à-dire une méthode d'en-tête
  public double getMoyenne()
  renvoie la valeur de l'attribut moyenne ;

- Un getter pour le nom de l'élève c'est-à-dire une méthode d'en-tête
  public String getNom()
  renvoie le nom de l'élève ;
- Un getter pour la liste des notes de l'élève c'est-à-dire une méthode d'en-tête
  public ArrayList<Integer> getListeNotes()
  renvoie la liste des notes de l'élève ;
- La méthode d'en-tête
  public void ajouterNote(int note)
  ajoute la note reçue en paramètre à listeNotes ; si la note reçue en paramètre est négative, la note introduite est 0 ; si la note reçue en paramètre est supérieure à 20, la note introduite est 20 ; la méthode actualise en conséquence l'attribut moyenne ; l'actualisation est faite à temps constant, et non pas en un temps proportionnel au nombre de notes déjà enregistrées.
- La méthode d'en-tête
  public String toString()
  qui retourne une description de l'élève considéré (par exemple : "Sophie (12.25)").

Questions :

1- Donner le diagramme de classe et le diagramme des cas d'utilisation

2- Implémenter les classes

3- Ecraire la classe de Test avec le main pour tester l'application

Exercice 4 :

On souhaite réaliser une application selon le cahier de charge ci-dessous :

1. "Cahier des charges"

Il s'agit de définir une classe JAVA permettant de modéliser des comptes bancaires. Cette classe (Compte) doit permettre à une application de créer et utiliser autant de comptes bancaires que nécessaires, chaque compte étant un objet, instance (ou exemplaire) de la classe Compte.

Un compte bancaire est identifié par un numéro de compte. Ce numéro de compte est un entier positif permettant de désigner et distinguer sans ambiguïté possible chaque compte géré par l'établissement bancaire. Chaque compte possède donc un numéro unique. Ce numéro est attribué par la banque à l'ouverture du compte et ne peut être modifié par la suite. Dans un souci de simplicité (qui ne traduit pas la réalité) on adoptera la politique suivante pour l'attribution des numéros de compte : les comptes sont numérotés de 1 à n, n étant le nombre de comptes qui ont été créés. Lorsque un nouveau compte est créé, le numéro qui lui est attribué est n+1.

Un compte est associé à une personne (civile ou morale) titulaire du compte, cette personne étant décrite par son nom. Une fois le compte créé, le titulaire du compte ne peut plus être modifié.

La somme d'argent disponible sur un compte est exprimée en Dirham. Cette somme est désignée sous le terme de solde du compte. Ce solde est un nombre décimal qui peut être positif, nul ou négatif.

Le solde d'un compte peut être éventuellement (et temporairement) être négatif. Dans ce cas, on dit que le compte est à découvert. Le découvert d'un compte est nul si le solde du compte est positif ou nul, il est égal à la valeur absolue du solde si ce dernier est négatif.

En aucun cas le solde d'un compte ne peut être inférieur à une valeur fixée pour ce compte. Cette valeur est définie comme étant - (moins) le découvert maximal autorisé pour ce compte. Par exemple pour un compte dont le découvert maximal autorisé est 2000 dh, le solde ne pourra pas être inférieur à -2000 dh. Le découvert maximal autorisé peut varier d'un compte à un autre, il est fixé arbitrairement par la banque à la création du compte et peut être ensuite révisé selon les modifcations des revenus du titulaire du compte.

Créditer un compte consiste à ajouter un montant positif au solde du compte.

Débiter un compte consiste à retirer un montant positif au solde du compte. Le solde résultant ne doit en aucun cas être inférieur au découvert maximal autorisé pour ce compte.

Lors d'une opération de retrait, un compte ne peut être débité d'un montant supérieur à une valeur désignée sous le terme de débit maximal autorisé. Comme le découvert maximal autorisé, le débit maximal autorisé peut varier d'un compte à un autre et est fixé arbitrairement par la banque à la création du compte. Il peut être ensuite révisé selon les modifications des revenus du titulaire du compte.

Effectuer un virement consiste à débiter un compte au profit d'un autre compte qui sera crédité du montant du débit.

Lors de la création d'un compte seul le nom du titulaire du compte est indispensable. En l'absence de dépôt initial le solde est fixé à 0. Les valeurs par défaut pour le découvert maximal autorisé et le débit maximal autorisé sont respectivement de 800 dh et 1000 dh. Il est éventuellement possible d'attribuer d'autres valeurs à ces caractéristiques du compte lors de sa création.

Toutes les informations concernant un compte peuvent être consultées : numéro du compte, nom du titulaire, montant du découvert maximal autorisé, montant du débit maximal autorisé, situation du compte (est-il à découvert ?), montant du débit autorisé (fonction du solde courant et du débit maximal autorisé).

Travail demandé

1. **A partir du "cahier des charges" précédent élaborer une spécification d'une classe Java modélisant un compte bancaire.**

   Il s'agira en analysant le texte ci-dessus de :

   - définir les attributs (variables d'instance, variables de classe) de la classe Compte,
   - d'identifier les méthodes publiques proposées par la classe Compte. Pour chaque méthode on prendra soin, outre la définition de sa signature, de spécifier son comportement sous la forme d'un commentaire documentant.
   - de proposer un ou plusieurs constructeurs pour la classe Compte. Là aussi on complétera la donnée de la signature de chaque constructeur avec un commentaire documentant détaillant son utilisation.

   - Donner le diagramme de classe et le diagramme des cas d'utilisation

2. **Réaliser une implémentation en langage Java de la classe précédemment spécifiée.**

3. **Ecrire un programme de test** permettant de :

- Créer un compte c1, au nom de Driss avec un solde initial de 1 000 dh

- Créer un compte c2, au nom de Driss avec un solde inital de 50 000 dh, un debit maximal autorisé de 6000 dh et un découvert maximal autorisé de 5000 dh.
- D'afficher les caractéristiques des comptes c1 et c2 (c'est à dire les informations suivantes : numéro du compte, nom du titulaire, découvert maximal autorisé, débit maximal autorisé, solde du compte et si le compte est à découvert un message le signalant explicitement).
- Retirer 300 dh du compte c1.
- Retirer 600 dh du compte c2.
- Déposer 500 dh sur le compte c1.
- D'afficher les caractéristiques des comptes c1 et c2.
- Virer 1000 dh du compte c2 vers le compte c1.
- D'afficher les caractéristiques des comptes c1 et c2.

4. Reprendre le code précédent, en y intégrant la gestion des exceptions sur l'ensemble des méthodes susceptible de générer des erreurs (saisie des montants, calcule du solde, découvert, débit, crédit…)

EXERCICE 5 :

Un concessionnaire des voitures de luxe, souhaite disposer d'un programme Java pour la gestion de son atelier de maintenance des voitures.

Chaque voiture « Voiture » est définit par :

- Identifiant, entier (qui n'est pas propre à une voiture en particulier et il s'incrémente à chaque fois)
- Marque, chaine de caractère
- Matricule, entier
- Date de production, chaine de caractère
- Prix unitaire HT, décimale

Pour réaliser une opération de maintenance, un technicien doit d'abord accueillir le client, et saisir ses informations :

- Identifiant, entier (qui n'est pas propre à un client en particulier et il s'incrémente à chaque fois)
- Code client, entier
- Nom et prénom, chaine de caractère
- Date de naissance, chaine de caractère
- Genre, caractère

Ensuite, l'agent de maintenance doit saisir les informations sur l'opération de maintenance :

- Code client
- Matricule de sa voiture
- Date d'achat de la voiture
- Date dernière visite à l'atelier
- Nature de l'opération de maintenance (Vidange, Visite, Echange de piece)
- Montant TTC de l'opération de maintenance, décimale (le taux tva appliqué est 20%)
- Mode de paiement, chaine de caractère (le modalité possible : CASH, CARTE, CHEQUE)
- Date de sortie de l'atelier, chaine de caractère
- Nom de l'agent de maintenance
- Code de l'agent de maintenance

Cette application, va être utiliser aussi par le directeur de l'agence pour consulter le tableau de bord des ventes.

1- Proposer un diagramme des cas d'utilisation et un diagramme de classe modélisant cette application (les classes possibles : Voiture, Client et Maintenance).
2- Ecrire les constructeurs des classes :

- sans argument
- avec argument
- de copie

3- Ecrire les méthodes pour les classes prévues :

- « getter & setter », pour la lecture et l'écriture.
- « affiche », pour afficher les informations

4- Pour permettre au directeur de l'agence de suivre les opérations de maintenance, améliorer la classe "Maintenance", en ajoutant les méthodes ci-dessous :

- "MCheck", pour le calcule les montants des opérations par jour, par mois et par année
- "QCheck", pour calculer la quantité des opérations par jour, par mois, par année
- "Prime", pour calculer la prime à attribuer aux agents de maintenance en fonction des opérations (le meilleur agent qui fait le maximum des opérations par mois, perçoit une prime de 2000 DH en plus à son salaire)

> Ecrire une classe de test main(), pour tester ce programme