S2 Module M2103 - Programmation Orientée Objet (POO)

EXERCICES 3 :
Encapsulation, attributs static, toString

Attributs et constructeurs

Vous devez écrire un programme permettant de gérer une banque.
Un compte courant est représenté par un numero, un solde et un seuil de découvert autorisé.
Chaque compte a un client qui est propriétaire de ce compte. Un compte appartient à un seul client.
Un client peut avoir plusieurs comptes.

1. Dessiner un schéma UML représentant les relations entre les classes Banque, Client et CompteCourant. Dans les questions suivantes vous devez implémenter ce schéma UML en Java. Les classes seront définies dans un package appelé "banque".
2. Créez la classe *CompteCourant* **en l'encapsulant**. Mettez les attributs de la classe *CompteCourant* en private. Utilisez le menu d'Eclipse Source > generate Getters and Setters pour générer automatiquement les accesseurs pour tous les attributs de la classe *CompteCourant*.
3. Définir un constructeur avec paramètres (un des paramètres aura le même nom qu'un des attributs)
4. Ajouter un test dans le setter de l'attribut solde : si on essaye d'y mettre une nouvelle valeur qui est supérieure à une constante SEUIL_SECURITE (que vous fixerez à 1000), afficher un message « ATTENTION tentative d'affectation suspecte d'un nouveau solde : compte no … » et ne modifiez alors pas cet attribut. Ce test doit aussi être fait lors de l'initialisation d'un compte.
5. Créez une classe *Banque* dans laquelle vous mettrez un main (). Dans ce main(), déclarez et créez quelques instances de la classe *CompteCourant*. Initialisez les attributs de ces comptes.
6. Essayez de modifier le solde de ces comptes en accédant directement à leurs attributs.
   Est-ce possible ?...
   Si ce n'est pas possible : Est-ce une erreur de compilation ou d'exécution ?...
   Quel est le message d'erreur ? ……
7. Vous devez maintenant gérer le fait qu'une banque a plusieurs comptes bancaires. Vous devez par exemple pouvoir ajouter un nouveau compte aux comptes existants.
8. Ecrire une méthode main () dans laquelle vous créez deux banques, des comptes que vous ajoutez à l'une ou l'autre banque.
9. Ecrire une méthode qui parcourt tous les comptes d'une banque et affiche les informations sur chaque compte. Est-ce mieux d'écrire une méthode de classe (static) ou une méthode d'objet ?
   ……
   ……
10. Ecrire les méthodes qui permettent d'afficher les informations sur tous les comptes d'un client spécifié par son nom (on supposera pour simplifier qu'il n'y a pas d'homonymes).
11. Ecrire une méthode qui, à partir d'un numéro de compte, affiche toutes les informations sur le propriétaire de ce compte.

12. Utilisez maintenant un attribut static pour initialiser automatiquement le numéro des comptes à l'aide d'un compteur spécifique à chaque classe.
13. Les attributs static peuvent servir à représenter une variable partagée par toutes les instances d'une même classe pour autre chose que de compter le nombre d'instances crées. Définissez un attribut static tauxRemuneration dont la valeur dépend des bénéfices **de toutes les banques au niveau national**. Fixez le à 1% au départ. Lors d'un dépôt (méthode crediter), ajouter ce pourcentage multiplié par le montant déposé. Prévoyez une méthode qui permet de changer ce taux. Dans le main() de la classe Banque, créez quelques comptes et faites des opérations en appelant les méthodes crediter et debiter. Augmenter le taux de rémunération à 2%. Faites de nouveaux quelques opérations et vérifiez que ce nouveau taux est bien pris en compte.
14. Un compte d'épargne est représenté par un numero, un solde, un taux d'interet (qui varie selon chaque compte) et une référence vers le client propriétaire de ce compte. Créez la classe CompteEpargne. Ajoutez la gestion d'un tableau de comptes d'épargne dans la banque.
15. Prévoir dans chacune de ces classes une méthode toString() :

```
// Méthode retournant une chaine de caractères décrivant la valeur des attributs de cet objet
public String toString () {
 String s = "" ;
 s = s + attribut ;
 ...
 return s ;
}
```

Cette méthode toString() est appelée ensuite de manière implicite lorsque vous affichez un objet avec println comme : System.out.println (clientDurant) ; // A UTILISER
est équivalent à : System.out.println (clientDurant.toString()) ; // A NE PAS UTILISER

16. Dupliquer votre package banque en un package banqueAL. Nous allons maintenant changer **l'implémentation** (les déclarations d'attributs et les instructions du corps des méthodes) de la classe Banque sans en changer l'**interface** (les entêtes des méthodes). => Remplacez les tableaux par des instances de la classe ArrayList (http://docs.oracle.com/javase/7/docs/api/index.html)

Est-ce que les entêtes des méthodes de la classe Banque ont changé ?
.....
.....

Est-ce que le main() de la classe Banque a changé ?
.....
.....

Est-ce que les développeurs qui utilisaient la version précédente de la classe Banque doivent changer leur programmes ?
.....
.....