Université A. Belkaïd Tlemcen
Faculté des Sciences
Département d'Informatique

Examen de rattrapage de Programmation Objet

Année universitaire 2014-2015
2e année licence informatique
Programmation Objet

Lundi 15 juin 2015

Documents non autorisés **Durée : 1h30**

Une société enregistre les niveaux de stock des produits qu'elle vend.

Chaque produit est représenté par une instance de la classe **`Produit`**, qui enregistre :

- son identité : un numéro de référence de type entier,
- son nom : une dénomination en lettres,
- sa quantité : le nombre d'unités en stock.

L'instanciation d'un nouveau produit nécessite des informations sur son identité et son nom. La quantité en stock initiale est toujours de zéro.

La classe **`Produit`** définit :

- la méthode **`augmenterQuantite()`** pour enregistrer les augmentations du stock d'un produit selon la valeur qui lui est passée en paramètre. Si cette valeur est négative, cette méthode affiche un message d'erreur et ne change rien.
- la méthode **`vendreUn()`** qui enregistre la vente d'une unité en diminuant la quantité du produit de 1. Elle affiche une erreur si le produit est épuisé.

1) De quoi est constitué **l'état** d'un objet de type **`Produit`** ?

2) De quoi est constitué son **comportement** ?

3) Écrire les instructions Java qui permettent de définir la classe **`Produit`** avec les contraintes suivantes :

- Respecter les principes de **l'encapsulation** de la manière la plus stricte.
- L'état d'un produit ne pourra être modifié que par les deux méthodes précédemment citées.
- Prévoir un moyen d'obtenir les valeurs de chaque attribut d'un produit.

4) Soit **`p`** un objet de type **`Produit`**.

a) Qu'affiche l'instruction suivante ? **`System.out.println(p) ;`**

b) Pourquoi ?

c) Écrire les instructions qu'il faut ajouter à la classe **`Produit`** pour que **`System.out.println(p)`** affiche l'état détaillé du produit **`p`**.

Un *gestionnaire de stock* est représenté par une instance de la classe **`GestionStock`**. Il stocke les objets **`Produit`** dans une **`ArrayList`**, qui est vide au moment de sa création. Il définit les méthodes suivantes :

- **`ajouterProduit()`** permet d'ajouter un objet Produit à la collection.
- **`afficherDetailsProduits()`** affiche une liste avec les états détaillés de tous les produits stockés.
- **`trouverProduit()`** cherche un produit en stock grâce à son numéro d'identité. Si elle le trouve elle arrête sa recherche et renvoie le produit, sinon elle retourne la valeur **`null`**.
- **`nombreEnStock()`** retourne à partir d'un numéro d'identité la quantité du produit correspondant. Elle renvoie zéro si le produit est épuisé ou absent de la liste.
- **`livraison()`** utilise un numéro d'identité et la quantité livrée pour augmenter la quantité en stock du produit s'il est présent dans la liste.

5) Écrire les instructions Java qui permettent de définir la classe **`GestionStock`**.

6) On veut définir une deuxième méthode **`trouverProduit()`** pour faire la même chose que la première mais en utilisant le nom d'un produit (chaîne) au lieu de son numéro d'identité (entier).

a) Est-il possible de le faire avec le même nom que la première ou faut-il que leurs noms soit différents ?

b) Pourquoi ?

c) Écrire la définition de cette méthode.

d) Qu'utilise-t-on pour comparer l'égalité des chaînes de caractères dans cette méthode ? Expliquer pourquoi.

7) La société évolue et commercialise à présent des produits frais, qui peuvent se périmer et ont donc une date de péremption avec jour, mois et année. Pour une meilleure gestion, il est obligatoire d'ajouter ces informations pour ces produits (mais pas pour les autres).

a) Écrire les instructions Java qui permettent de définir la classe **`ProduitFrais`** en évitant les répétitions.

b) Que doit-on changer dans la classe **`Produit`** ? Expliquer.

c) Que doit-on changer dans la classe **`GestionStock`** ? Expliquer.