Travaux Pratiques N°5

Problème A :

On désire programmer une mini-application de gestion de petit commerce. Pour cela, on va écrire une classe **Produit** et une classe **Magasin**.

Exercice 1 : Première version du magasin

Dans notre commerce, chaque produit est caractérisé par son nom, son prix d'achat, son prix de vente, le nombre d'exemplaires en stock et sa description.

A la création du produit, on fixe son nom, son prix d'achat et son prix de vente, c'est-à-dire que ces données sont fournies en argument au constructeur de la classe. Par défaut, la description du produit est «Pas de description», et la quantité de produits en stock est nulle.

La classe **Produit** dispose d'un certain nombre de méthodes, qui lui permettent d'afficher la description du produit, de l'éditer, d'augmenter ou de diminuer le nombre d'exemplaires en stock, ainsi que d'obtenir les valeurs des différents attributs.

Question 1.1: Écrivez la classe Produit.

Un magasin se caractérise par son solde et son stock de produits. Le stock de produit est représenté par un tableau d'objets «**Produit**».

Avant de pouvoir acheter ou vendre un produit, il faut l'avoir ajouté dans le stock. Pour cela, la classe **Magasin** dispose d'une méthode publique:

void ajouterProduit (String nom, float prixAchat, float prixVente)

L'indice de la case du tableau dans laquelle on a stocké le produit devient alors la référence de ce produit. Pour acheter ou vendre ce produit, on utilise alors la référence comme argument des méthodes : **void acheterProduit (int referenceProduit, int nombreExemplaires)**

void vendreProduit (int referenceProduit, int nombreExemplaires)

La classe **Magasin** dispose également des méthodes habituelles d'accès à ses attributs, et d'une méthode Bilan qui permet d'afficher un bilan du magasin.

Question 1.2: Écrivez la classe Magasin. Testez-la.

Exercice 2: Différents types de produits

Jusque-là, les produits disposent d'une description qui est une simple chaîne de caractère. Cela peut être insuffisant dans certains cas. Par exemple, un livre est un produit particulier qui est identifié par son auteur et son éditeur. De même, un produit alimentaire est caractérisé par sa date de fabrication et sa date d'expiration.

Question 2.1 : Écrivez la classe Livre et la classe Aliment qui héritent de la classe Produit. Certaines méthodes doivent être surchargées (c'est-à-dire réécrites).

Il va maintenant falloir faire les modifications nécessaires dans la classe **Magasin**, mais elles sont minimes. En effet, **Livre** étant une sous-classe de **Produit**, les objets de type **Livre** sont également de type **Produit**, donc le tableau de produits peut également contenir des livres et des aliments. C'est à la création des objets que les choses changent. En plus de la méthode **ajoueterProduit**, la classe **Magasin** doit également contenir une méthode **ajouterLivre** et une méthode **ajouterAliment**.

Question 2.2: Modifier la classe Magasin pour tenir compte des nouveaux objets.

Exercice 3: Recherche par mot

Dans les parties précédentes, pour acheter, vendre, éditer ou afficher la description d'un produit existant, il fallait le spécifier par sa référence, c'est-à-dire par son indice dans le tableau. Ce n'est pas un moyen très pratique d'aller chercher les produits, surtout s'il y en a un très grand nombre.

Nous allons écrire les méthodes :

- **Produit rechercheProduit (String nom)** qui renvoie un objet Produit dont le nom est spécifie en argument,
- **Produit rechercheProduitParMot (String mot)** qui renvoie un objet Produit dont la description contient le mot spécifié en argument.

Pour écrire ces méthodes, vous allez avoir besoin d'un outil capable de comparer deux chaines de caractères, et d'un outil capable de dire si une chaine est incluse dans une autre.

Question 3.1: Écrivez les méthodes rechercherProduit et rechercherProduitParMot.

Problème B :

On désire programmer une mini-application de gestion des ports de pêche. Pour des raisons de simplicité, on se limite à une description sommaire des barques (petits bateaux) et des ports. Une barque est caractérisée par son nom, son numéro d'autorisation ainsi que par une méthode **toChaine()** qui décrit ses propriétés. Il existe aussi des barques dotées de moteurs de puissances différentes. La puissance, exprimée en nombre entier, est supposée l'information unique qu'il faut représenter pour le moteur d'une barque. Une méthode **getPuissance()** est prévue pour chaque barque dotée de moteur. Un port de pêche est caractérisé, entre autres, par un nom et une capacité d'accueil de barques. Une taxe journalière est prévue pour chaque barque se trouvant dans le port. Le montant de cette taxe est approprié à chaque port. Une

majoration de la taxe est prévue pour les barques avec moteur à raison de 10% si la puissance du moteur est < 5 et de 20% si non. Il faut connaître, à chaque moment, le nombre actuel de barques dans un port donné. Aussi, il faut gérer les opérations d'entrée et de sortie de barques par des méthodes appropriées. En plus d'une méthode de description **toChaine()** qui décrit aussi les caractéristiques des barques se trouvant dans le port, il faut prévoir une méthode qui calcule le montant d'une recette de taxe pour la journée en cours. Utilisez un seul vecteur pour représenter toutes les barques dans un port (de type Barque et BarqueMoteur à la fois).

Questions :

1. Définir les classes Barque, BarqueMoteur et Port.
2. Définir une classe TestPort qui se charge de la création d'un objet Port. D'un objet BarqueMoteur. Cette fonction doit faire entrer les deux barques au port et calculer le montant de taxe.