Concepts objet
Exercices

SOMMAIRE

INTRODUCTION

Nous allons travailler sur les explications du grossiste désirant vendre des produits.

Nous commencerons simplement et progressivement afin d'inclure au fur et à mesure les concepts objets.

1. DÉFINITION DU PRODUIT

Objectif : Manipulation des Classes, variables, méthodes et Objets.

Utilisation de l'encapsulation.

Notre grossiste vend des produits. Il désire mémoriser :

- le code, sous forme de chaine de caractères,
- la dénomination, sous forme de chaine de caractères,
- le prix, sous forme d'entiers.

Il faut donc créer des méthodes permettant :

- La création du produit en passant en argument le code du produit.
- La mise à jour de la dénomination, en passant en argument une chaîne de caractères,
- La mise à jour du prix, en passant en argument un entier,
- La consultation du code, on retourne une chaîne de caractères,
- La consultation de la dénomination, on retourne une chaîne de caractères,
- La consultation du prix, on retourne un entier.

1.1. FAIRE LA REPRÉSENTATION UML

1.2. Définir le code java pour la classe Produit

On utilisera l'encapsulation.

1.3. Définir le code java pour tester la classe Produit

On crée deux produits :

- produit1 de type Produit ayant :

  un code = « produit100 »

  une démonination = «crayon de papier »

  un prix = 1

- produit2 de type Produit ayant :

  un code = « produit200 »

  une démonination = «crayon de couleur»

  un prix = 2

On affiche :

Le code et la dénomination du produit1.

Le code, le prix du produit2 et sa dénomination.

2. AGREGATION OU COMPOSITION

Objectif : Manipulation d'un lien de composition.

Notre grossiste décide d'avoir des informations plus judicieuses sur la dénomination du produit. Il souhaite avoir :

- Un libellé long,
- Un libellé court.

On doit donc créer une classe Denomination et l'utiliser dans la classe Produit.

2.1. FAIRE LA REPRÉSENTATION UML

2.2. DÉFINIR LA CLASSE DENOMINATION

2.3. MODIFIER LA CLASSE PRODUIT

Ajout d'une variable d'instance de type Denomination.

Modification du constructeur de Produit, afin d'initialiser la variable d'instance.

Ajout d'une méthode afficheLibelle retournant une chaîne de caractères contenant le libellé court un espace et le libellé long.

Ajout d'une méthode modifLibelle attendant deux chaînes de caractères contenant le libellé court et le libellé long afin de renseigner la dénomination.

2.4. MODIFIER LA CLASSE TESTPRODUIT

Pour tester nos produits et nos dénominations, on utilise les méthodes modifLibelle et afficheLibelle.

On utilise toujours nos deux produits : produit1 et produit2.

3. HERITAGE SIMPLE

Objectif : Manipulation des liens de généralisations et de spécialisations.

Notre grossiste décide de vendre des produits saisonniers (bonnets, écharpes, parapluies,...) et des produits périsables (produits laitiers, viandes,...). Ce sont des sortes de produits.

Pour le produit saisonnier, il souhaite mémoriser :

- la saison, sous forme de chaîne de caratères,
- la remise de fin de saison, sous forme d'entier.
- On doit pouvoir récupérer et manipuler la saison et la remise.

Pour le produit périssable, il souhaite mémoriser :

- la durée de conservation, sous forme d'entier.
- la date de fabrication, sous forme de date. (On utilise la classe java.util.Date en créant une instance de Date on obtient la date système.)
- On doit pouvoir récupérer et manipuler la durée et la date.

3.1. FAIRE LE DIAGRAMME DE CLASSES

3.2. DÉFINIR LES CLASSES

3.2.1. PRODUITPERISSABLE

3.2.2. PRODUITSAISONNIER

3.3. Créer la classe TestProduits

Pour tester nos produits, on utilise la classe TestProduits. On reprend la classe TestProduit :

En utilisant toujours nos deux produits : produit1 et produit2 de type Produits.

On ajoute deux objets supplémentaires, pour lesquels on manipulera leurs différentes méthodes:

- produit3 de type ProduitPerissable ;
- produit4 de type ProduitSaisonnier.

On enregistre les produits dans un tableau de Produit.

Pour chaque poste du tableau, on applique la méthode getPrix().

Informations: Manipulation de tableaux

Définition d'un tableau de Produit nommé liste:

```
Produit liste[] = new Produit[n]
// n représente le nombre d'éléments que l'on désire mémoriser.
```

Initialisation d'un poste/d'un élément dans le tableau liste défini précédemment :

```
liste[n] = produit1 ;
// n représente l'élément que l'on désire mémoriser, on débute à 0.
```

Récupération d'un poste/d'un élement dans le tableau liste :

```
Produit  p = liste[n];
```

Nombre de postes/d'éléments dans le tableau liste, on applique la variable length au tableau :

```
liste.length;
```

4. POLYMORPHISME

Objectif : Redéfinition de la méthode retournant le prix.

Notre grossiste décide :

- Pour la classe ProduitSaisonnier : d'appliquer la remise. Le traitement à faire est :

  prix - remise

  le prix est celui défini dans la classe mère

  la remise est celle de la classe ProduitSaisonnier.

4.1. MODIFIER LA CLASSE PRODUITSAISONNIER

4.2. CLASSE TESTPRODUITS

Que faut-il faire ?

5. HÉRITAGE MULTIPLE

Objectif : Manipulation des interfaces et surcharge des méthodes.

Notre grossiste décide de vendre des huîtres, il s'agit d'un produit de type saisonnier et de type périssable. Etant donné qu'il n'y a pas pas d'héritage multiple en Java, il faut manipuler une interface.

5.1. CRÉATION DE L'INTERFACE SAISON

Cette interface déclare des méthodes permettant de récupérer et manipuler la saison, la remise et le prix. (On utilise les conventions d'appellation : get/set).

5.2. CRÉATION DE LA CLASSE PRODUITFUGACE

Cette classe implémente l'interface Saison , hérite de ProduitPerissable et définit les méthodes héritées de l'interface permettant de récupérer et manipuler la saison, la remise et le prix.

5.3. MODIFICATION DE LA CLASSE PRODUITSAISONNIER

Cette classe implémente l'interface Saison.

5.4. MODIFICATION DE LA CLASSE TESTPRODUITS

Manipulation d'objets de type Produit, ProduitSaisonnier, ProduitPerissable, ProduitFugace.

5.5. CRÉATION DE LA CLASSE TESTSAISONS

Manipulation d'objets de type ProduitSaisonnier et ProduitFugace, afin de tester les méthodes getSaison et getPrix.

6. COMPLÉMENTS (OPTIONNEL)

Objectif : Manipulation des concepts objets.

Notre grossiste doit prévoir de renouveller son stock. Il doit donc enregistrer les produits qu'il doit commander.

Comment représenter cette fonctionnalité, quels sont les traitements à prévoir.