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 caractères,
- 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. DEFINIR LES CLASSES

3.2.1. PRODUITPERISSABLE

3.2.2. PRODUITSAISONNIER

3.3. CREER 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 :

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

Initialisation d'un poste 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 dans le tableau liste :

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

Nombre de postes dans le tableau liste, on applique la variable length au tableau :

```
liste.length;
```

4. POLYMORPHISME

Objectif : Surcharge 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. HERITAGE 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. CREATION 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. CREATION 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. CREATION DE LA CLASSE TESTSAISONS

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