Examen Licence Informatique - L3

Programmation par objets 2 (FLIN 606)

Session 1 - avril 2008 - 2h

Tous documents autorisés

Une chocolaterie souhaite mettre en place un logiciel de vente sur internet de ses produits. La figure 1 vous présente quelques premiers éléments de conception.

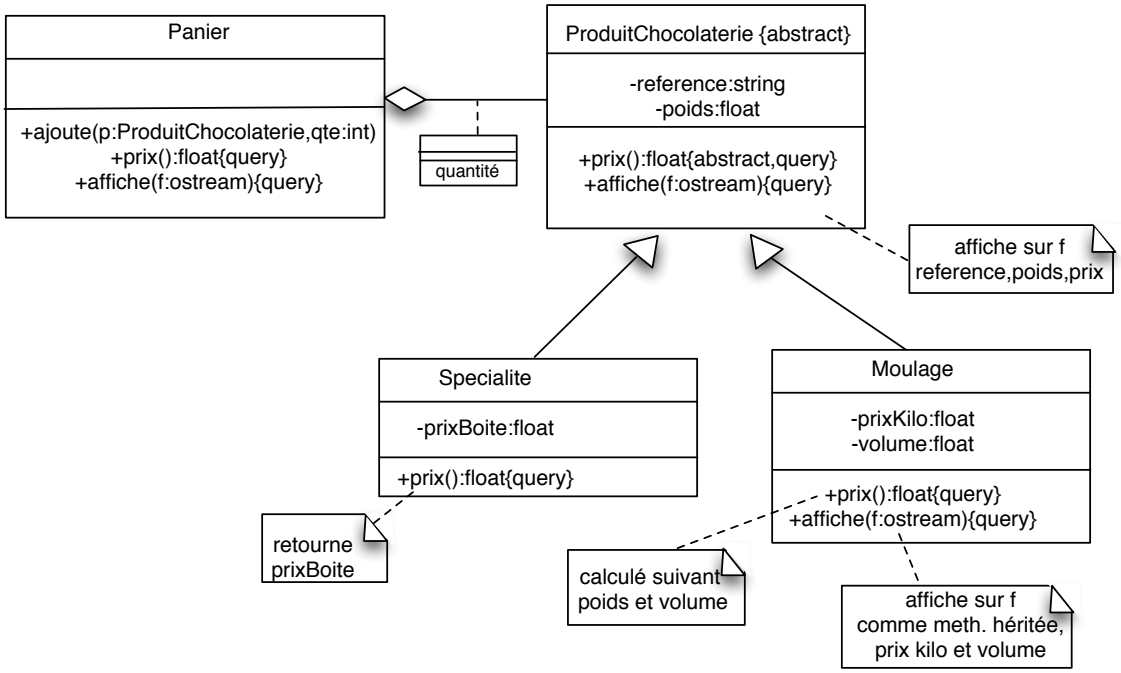

Fig. 1 – Une solution de conception pour la chocolaterie

1 Héritage simple

Dans cette question, nous nous intéressons uniquement à l'écriture en C++ des classes `ProduitChocolaterie`, `Specialite` et `Moulage`.

Un `ProduitChocolaterie` est décrit par une référence (chaîne de caractères) et un poids en kilo (nombre réel). On pourra calculer son prix et l'afficher.

Un `Moulage` est un sujet en chocolat (poule, lapin, œuf, etc.). On le décrit plus spécifiquement par son volume (en cm^3) et par le prix au kilo du chocolat qui le constitue. Son prix se calcule d'après (1) le prix du chocolat au kilo rapporté au poids du moulage auquel on ajoute (2) 3 euros pour chaque 10 cm^3 de volume. Son affichage se résume à sa référence, son poids, son prix, le prix au kilo du chocolat qui le constitue et son volume.

Une `Specialite` est un produit dont la marque a été déposée (calissons d'Aix, gavottes au chocolat, bouchon de champagne, etc.) qui ne se vend que par boîtes. On la décrit plus spécifiquement par le prix d'une boîte de cette spécialité. Son affichage se résume à sa référence, son poids et son prix.

Question 1.1 (classes) *N'écrivez dans cette question QUE l'interface et le cas échéant l'implémentation pour les trois classes des éléments suivants :*

- *l'entête des trois classes (dont les noms, liens de spécialisation, etc.),*
- *les attributs,*
- *les constructeurs (justifiez leur implémentation et le choix ou non de faire un constructeur par copie),*
- *les destructeurs (justifiez leur implémentation).*

On supposera pour la suite que les accesseurs en lecture et en écriture aux attributs existent.

Question 1.2 (`prix`)
Ecrivez pour les classes représentant les produits de la chocolaterie, la méthode ***`prix`****.*

Question 1.3 (`affiche` **et surcharge d'opérateur)**
Ecrivez pour les classes représentant les produits de la chocolaterie, la méthode ***`affiche`****. Proposez ensuite une surcharge de l'opérateur d'insertion dans un flot pour afficher les produits de la chocolaterie.*

2 Héritage multiple

Etudiez l'ajout d'une sous-classe `OursDeBerne` qui est à la fois une sous-classe de `Specialite` et une sous-classe de `Moulage`. Un ours de berne est décrit plus spécifiquement par le fait d'être ou non fourré à la crème (booléen).

Question 2.1 (Héritage virtuel)
Ecrivez l'interface et l'implémentation de la classe ***`OursDeBerne`*** *de manière à faire de l'héritage virtuel : constructeurs, destructeur, opérations* ***`affiche`*** *et* ***`prix`****. Pour la méthode* ***`prix`****, on prendra le maximum des prix calculés pour les deux superclasses. Pour l'affichage, faites votre propre proposition. Il ne doit pas y avoir de conflits.*

Question 2.2 (Héritage répété)
Indiquez ce qu'il faut changer dans vos classes afin de faire de l'héritage répété : vous devriez indiquer des modifications dans les entêtes (y compris des deux super-classes), les constructeurs, peut-être certaines méthodes. Justifiez vos modifications.

Question 2.3 (Ordre d'appel des constructeurs)
Indiquez, dans les deux cas d'héritage (virtuel et répété), l'ordre d'appel des constructeurs lors de la création d'une instance de ***`OursDeBerne`****.*

3 Utilisation de STL

Nous étudions à présent la classe `Panier` représentant les paniers d'achats. Un panier contient un certain nombre de produits achetés chacun dans une certaine quantité. Pour représenter le contenu du panier, il vous est imposé d'utiliser une `map` de la librairie STL : les clefs seront les produits de la chocolaterie, les valeurs seront les quantités achetées. Les éléments qui vous sont nécessaires ici pour manipuler les `map` sont tous illustrés dans le polycopié de cours : insertion avec l'opérateur [ ], parcours et accès avec un itérateur sur le même schéma que les `set` (sauf qu'on récupère une paire dans le cas des `map`), accès avec la méthode `find` et aux attributs `first` et `second` de la paire. Notez que la méthode `find` retourne l'itérateur sur la fin de la map (également retourné par `la méthode end()`) lorsque la clef passée en paramètre n'est pas trouvée.

Question 3.1 *Ecrivez l'intégralité de l'interface et l'implémentation de la classe* ***`Panier`*** *en respectant les méthodes proposées dans le schéma de conception de la figure 1.*

4 Généricité paramétrique

Question 4.1 *Proposez une version générique de la classe* ***`Panier`*** *paramétrée par le type des produits rangés à l'intérieur. Instanciez-la dans un programme* ***`main`*** *pour obtenir un panier de produits de la chocolaterie et appelez la méthode* ***`ajoute`*** *avec un produit que vous aurez créé préalablement.*

5 Signalement d'exception

Question 5.1

Nous approfondissons l'écriture de la méthode `ajoute(p :ProduitChocolaterie,qte :int)` *pour la classe* `Panier` *(version générique ou non cela n'importe pas pour cette question). Nous supposerons qu'essayer d'ajouter un produit déjà présent dans le panier est un cas d'erreur.*

(a) Proposez une classe d'exceptions pour représenter le cas où on cherche à ajouter un produit déjà présent dans le panier.

(b) Complétez la méthode `ajoute` *en prévoyant le signalement d'exception approprié.*

(c) Ecrivez également un petit programme qui se préoccupe de capturer l'exception liée à un ajout.