

M1 Informatique PR - Ingénierie Logicielle

TD-TP No 5

Schémas de conception (*Design Patterns*) - Suite

1 TD : Locations, version 1, Double Dispatch

Considérons la modélisation objet d'un logiciel utilisé dans un magasin de location. Le magasin permet à ses abonnés de louer des produits : des DVDs et des livres (le prix hors taxe de location d'un livre est ajusté chaque semaine en fonction de l'évolution du prix de vente du même livre dans le commerce, la TVA pour un livre est 5%).

Pour louer, tout client doit posséder un compte. Le prix de la location d'un produit dépend du produit loué (DVD ou Livre) et du type de compte. Pour faire un système extensible, on fera donc en sorte que le produit et le compte soient consultés pour définir le prix de location du produit. Le protocole de calcul de prix est le suivant :

```
--> Envoi du message "calculPrixLocation(Produit p)" à un compte c,  
  --> Envoi du message "calculPrixLocation(c)" à p,  
    --> p calcule de son point de vue, son prix TTC et le passe à c  
    --> Envoi du message prixFinalLocation(float prixTTC) à c,  
      --> c reçoit le prixTTC et applique d'éventuelles réductions qui lui sont propres  
<-- en retour le prix final de la location.
```

On distingue les comptes normaux (`Compte` et les `CompteAvecReduction`). Un compte avec réduction est plus cher à la base mais donne ensuite droit à une réduction de 10% sur chaque produit loué.

Voici un exemple d'utilisation d'instances des classes représentant les concepts précédents.

```
Client cl = new Client("Dupont");  
Compte cmt = new Compte(cl, 1000);  
DVD lgv = new DVD("La grande vadrouille");  
cmt.calculPrixLocation(lgv);  
Compte cmt2 = new CompteAvecReduction(cl, 1000);  
cmt2.calculPrixLocation(lgv);
```

Vous ne vous occuperez pas dans ce sujet de distinguer produit et exemplaire de produit. Un DVD est simplement représenté par une instance de la classe DVD.

La classe `Compte` définit les méthodes suivantes :

```
- public float calculPrixLocation (Produit p)  
- package float prixFinalLocation(float f)
```

La classe `Produit` déclare ou définit au moins les méthodes suivantes :

```
- protected float prixHT()  
- protected float prixTTC()  
- package float calculPrixLocation (Compte c).
```

Questions :

1. Définissez les classes et méthodes utiles à et compatibles avec cet énoncé.
2. La conception précédente prétend définir, via un double envoi du message `calculPrixLocation`, réalisant un *double dispatch* un programme extensible du point de vue du calcul du prix. En effet vous avez constaté qu'il est possible de définir de nouveaux types de comptes (`CompteAvecReduction` sans modifier le code existant. Vérifiez qu'il est de même possible de définir de nouveaux types de produits (par exemple `ProduitSoldé`).

2 TD : Locations, version 2, utilisation du schéma “Décorateur”

Le gérant du magasin décide de varier son offre et de créer des comptes avec forfaits car le système précédent est un peu limité. Les forfaits sont potentiellement cumulables et peuvent être à tout moment ajoutés à, ou supprimés, d'un compte existant. On peut imaginer différentes sortes de forfaits mais limitons nous pour le principe aux deux suivants :

- f10 permet d'obtenir une réduction de 10% sur toute location. Il coûte 5 euros à l'achat.
- f20 permet d'obtenir une réduction de 20% sur toute location. Il coûte 10 euros à l'achat.

Pour programmer ce système, la solution précédente est insuffisante. Inspirez vous du schéma de conception “Decorateur” que vous avez étudié en cours pour proposer une solution logicielle adaptée au problème posé ici que vous illustrerez par un diagramme de classes UML. N'oubliez pas d'indiquer la signature des méthodes utiles à l'exemple.

Quels limites voyez vous à ce schéma de conception ? Imaginez un exemple de forfait incompatible avec un autre.

3 TP

Programmez les deux versions du système de location dans deux packages différents.