Eurydice Ruggieri
Thomas von Ascheberg

Explication de la hiérarchie de classe

Notre projet se décompose en 26 classes fonctionnelles (hors tests et exceptions). Comme on le voit bien sur le diagramme de classe, il y a 4 parties bien distinctes dans le projet : les produits (avec les catégories de produit), les différentes offres, les personnes (et leur catégorie) et les différentes alertes. On fait communiquer le tout via la classe ServiceMarketing qui lie tout le monde. Le ServiceMarketing est l'élément central de notre projet. A chaque création de personne, d'offre, d'alerte ou de produit, le ServiceMarketing s'occupe de vérifier que tout le monde est à jour (vérification que les paniers contiennent les offres applicables sur ceux-ci, préviens les alertes pour qu'elles examinent les paniers déjà créés, ajout des alertes comme listener des paniers, etc.).
Pour modéliser ce système, nous avons reconnu et implémenté plusieurs "Design Pattern":

- **1** pattern **State** : pour les catégories de client (classes Categorie, Adherent, Personnel et Visiteur). En effet, chaque catégorie a des propriétés propres et l'on doit pouvoir changer dans le temps de catégorie (en se connectant/déconnectant) : c'est exactement le design pattern State
- **1 faux State** : pour les catégories de produit (classes CategorieProduit, Livre, Spectacle, HighTech). Chaque catégorie de produit s'occupe de redéfinir l'égalité et l'affichage pour la catégorie produit. Chaque catégorie de produit a ses propriétés propres et son comportement propre, notamment face aux offres (la seule différence avec un vrai State est qu'elles ne peuvent pas changer dynamiquement).
- **1** pattern **Observable/Observer** : toutes les classes qui doivent être informées lors d'une mise à jour du panier (classes qui écoutent le panier via l'interface PanierListener : Alerte, AlertePrix, AlerteProduit, AlerteCombinaisonProduit, CarteFidelite).
- **2** patterns **Singleton** : Le premier (classe BaseLog) représente la base de données que l'on aurait derrière l'application pour gérer les connexions, le stockage des mots de passe, etc. Le deuxième singleton est le service marketing (classe ServiceMarketing). Il gère les liens à établir lors de la création des objets et les vérifications à effectuer avant de faire ces liens. Il est notamment responsable de la vérification des offres avant de les ajouter aux paniers où elles s'appliquent. Il gère aussi la vérification des alertes à la création de celles-ci (notamment de les ajouter comme listener du panier de chaque client).

Demeure une petite entorse à la modularité pour préserver l'encapsulation. Pour le ServiceMarketing, on est obligé de créer un traitement approprié à chaque type d'offre (avec des instanceof) pour l'ajout au panier. (En effet, on ne souhaite pas que les offres s'ajoutent d'elle-même au panier ni ne connaissent la structure ou le fonctionnement du panier : c'est le rôle du ServiceMarketing d'abstraire cela).

Nos tests fonctionnels sont présents dans la classe _Application.java

Description des tests fonctionnels

Afin de pouvoir vérifier le bon fonctionnement de notre application, nous avons mis en œuvre un ensemble de tests fonctionnels que nous allons détailler ici:

Test 1 :

Le but de ce test est de montrer que certaines catégories de produit (comme les livres) ne peuvent pas recevoir de réductions. Ici, on dépasse le seuil de points de notre carte fidélité en achetant 2 livres. Ainsi, une offre fidélité est générée et tente de s'appliquer au panier mais échoue. En résulte que l'on paye les livres à leur prix (sans réduction).

Test 2 :

Le but de ce test est de montrer que l'on peut combiner plusieurs types d'offre différents sur un même produit. Ici, on génère 3 offres sur le produit "ecran" (Une offreAdherent (25%), une offreFidélité (10%) car il dépasse le seuil de points de sa carte et une offreProduit (50%)). Notre client qui est un adhérent accumule ainsi les deux offres et paye son produit en multipliant les réductions. Ainsi au lieu de payer son "ecran" 300€ il ne paye que 101,25€ (il a obtenu 66,225% de réduction).

Test 3 :

Le but de ce test est de montrer le fonctionnement du système d'alerte. Ici, on génère 4 alertes (deux qui sont des AlerteCombinaisonProduit, une AlerteProduit et une AlertePrix). On veut montrer que l'alerte prend en compte le contenu du panier (en termes de quantité). Ainsi, la 2e alerte ne doit pas se lever car le panier ne contient pas tous les produits (2/3). Ici comme attendu, les 3 alertes se lèvent correctement au moment de l'ajout des produits (alertes portant sur les produits) ou au moment de payer le panier (alerte sur le prix du panier).

Test 4 :

Le but ici est juste de montrer que les alertes ne perturbent pas le comportement des produits face aux offres. Ainsi, on ne peut toujours pas appliquer d'offre sur le livre malgré la présence d'une alerte sur ce même livre. Ainsi on n'applique qu'une seule des offres ici. Au moment de payer le client n'obtient un rabais que sur la TV (pas sur le livre).

Test 5 :

Le but de ce test est de montrer que le fonctionnement imbriqué des offres flash et des alertes. Ici, on génère une offre flash qui s'applique sur exactement les mêmes produits que l'alerte non levée lors du test3 (2 livres et la TV). Lorsque l'on ajoute au panier les bons produits, l'alerte se lève et l'offre est vérifiée. Ainsi lorsque l'on paye, elle s'applique (mais que sur la TV car l'on ne peut pas avoir de réduction sur les livres). De plus on montre que l'on peut cumuler l'offre flash avec les autres offres (en effet l'offre produit sur la TV s'applique toujours). Ainsi on paye la TV à 25% de son prix initial (50% par offre) et 40€ pour les 2 livres.

Test 6 :

Le but de ce test est de montrer que certaines offres ne sont valables que pour certains clients qui appartiennent aux bonnes catégories. Ici, on va ajouter le même spectacle (spectacleTragique) au panier d'un client adhérent (client1), d'un client membre du personnel (client2) et d'un client visiteur (client3). Ainsi on génère sur ce spectacle une offreAdhérent (réservée aux adhérents) et une offrePersonnel (réservée aux membres du personnel). Comme attendu, lorsque les clients payent, les offres s'appliquent bien mais seulement aux clients de la bonne catégorie.