CNAM

M-NFA032 Algorithmique et programmation : bibliothèques et patterns

Cinquième partie

Travail à effectuer

Rappels

Quelques éléments importants avant de commencer le fil rouge :

- à terme, votre programme comportera une interface graphique ; cette interface ne vous permettra pas de lire au clavier ni d'écrire à l'écran ; donc, ne pas mettre d'ordre de lecture ou d'écriture dans vos classes ; si vous souhaitez valider vos classes, prévoyez alors une petite bibliothèque temporaire pour les entrées-sorties.
- Dans vos classes, mettez systématiquement vos champs private ou, à la limite, protected ; vous pourrez ensuite donner des accès à vos variables d'instances via des getters et des setters.

Exercice 1 : Equals et hashCode

Dans cet exercice, nous allons redéfinir les méthodes equals et hashCode de quelques classes qui pourront être stockées dans des collections.

Pour toutes les classes dans le package produit, il n'y a rien à faire puisque ces méthodes ont été redéfinies dans Produit et que toutes les autres classes de ce package en héritent.

Catalogue, il n'y en aura qu'une seule instance. Il n'est pas nécessaire de le faire.

LignePanier doit déjà avoir été prévue.

Redéfinir les méthodes pour Panier est sans intérêt.

1. Pour la classe Client, deux clients seront identiques si ils ont le même e-mail qui sert d'identifiant sur le futur site.
2. Pour LigneCommande, on pourra se contenter de comparer les produits.
3. Pour Commande, le numéro de commande suffira à distinguer deux commandes.

Exercice 2 : Sérialisation des classes

La notion de sérialisation étant vue, nous pouvons préparer toutes les classes en vue de leur enregistrement dans un fichier. Il n'est pas nécessaire de le faire pour les énumérations.

Compteur

1. Ouvrez la classe Compteur dans le package utilitaires.
2. A la suite de `public class Compteur` ajoutez `implements Serializable` pour indiquer que les instances de cette classe pourront être enregistrées.
3. Serializable doit être souligné en rouge car nous n'avons pas importé la classe. Faites Ctrl-Maj-O pour corriger le problème.
4. Maintenant, le nom de la classe « Compteur » doit être souligné en jaune car nous n'avons pas déclaré la constante serialVersionUID. Pour corriger ceci, amenez le curseur de la souris sur Compteur souligné en jaune et, après quelques secondes, choisissez

« Add default serial version ID ». Ceci doit ajouter une ligne

```
private static final long serialVersionUID = 1L;
```

à la classe. C'est le numéro de série que vous attribuez à cette classe et qui doit être modifié chaque fois que vous modifiez la classe. 1L signifie 1 codé en long.

Produit

1. Faites les mêmes opérations pour la classe Produit.
2. Les cinq classes héritières de Produit doivent maintenant apparaître avec un triangle jaune car elles attendent qu'on définisse la constante serialVersionUID. On pourrait ne pas le faire et laisser le compilateur Java le faire pour nous, mais il vaut mieux rester maître de ce numéro de série. Nous allons appliquez l'étape 4 de Compteur à chacune de ces 5 classes en une seule manipulation.
   1. Dans le menu Window, Cliquez sur Show View > Problems
   2. Cliquez sur la petite flèche « > » à gauche de « Warnings » pour voir les items
   3. Repérez une ligne commençant par « The serializable class Cahier does not declare a static final ... » et cliquez dessus avec le bouton droit de la souris.
   4. Choisissez « Quick fix »
   5. Choisissez maintenant « Add default serial version ID »
   6. Cliquez sur Select all
   7. Finish. Toutes les classes doivent avoir été corrigées.

Les autres

1. Ouvrez la classe Compteur et faites une copier de « `implements Serializable` »
2. Ouvrez maintenant Catalogue et faites un coller au bon endroit. Eclipse insère de lui-même la ligne
   ```
   import java.io.Serializable;
   ```
3. Répétez cette opération pour les classes suivantes : Client, Commande et LigneCommande. Panier et LignePanier ne seront jamais enregistrées donc, pas utile de les définir Serializable.
4. Corrigez les avertissements concernant ces 4 classes comme il a été fait au point 2 de Produit.

Exercice 3 : Carnet de commande

Nous allons créer une structure de données CarnetCommande pour conserver toutes les commandes de chaque Client et celui du magasin. Pour les conserver par ordre chronologique et les retrouver facilement par leur numéro, nous utiliserons une LinkedHashMap avec un Integer comme clé et une Commande comme valeur. Bien sûr, les carnets de commande seront sauvegardés ! Donc, prévoyez !

1. Créez cette classe. Pour l'instant, ne prévoyez qu'une méthode d'ajout et une autre pour récupérer une commande par son numéro. Cette classe sera complétée au fur et à mesure des besoins.
2. Dans la classe Client, ajoutez un champ de type CarnetCommande et les méthodes d'ajout et de recherche par numéro qui délégueront au carnet de commande. Prévoyez de créer un carnet de commande à la création du client.

Exercice 4 : Carnet de Commande Magasin

Construire une classe CarnetCommandeMagasin appliquant le patron de conception Singleton (Cf. classe Catalogue). Cette classe encapsulera uniquement un carnet de commande. Pensez à d'ores et déjà déléguer les deux services de CarnetCommande.

Exercice 5 : Carnet clientèle

Créez une classe CarnetClientele qui permette de retrouver efficacement un client par son email. Appliquez dessus le patron singleton. Pour l'instant, prévoyez une méthode d'ajout d'un client et une méthode de recherche par email.

Nous en avons fini avec les classes du modèle. Il restera à les compléter au fil des besoins.