NFA032 – TP 5 et 6 (Objets partie 3 : objets dans les objets, un peu de conception OO)

12 mars 2019

1 Préliminaires

Lancez Eclipse et créez un nouveau projet de nom `Tp5-032`. Vous aurez besoin de la classe `Livre` (devéloppée par vous) lors d'un tp précédent. Nous traiterons ce sujet sur 2 séances.

2 Exercice 1 : classe StockLivres

Créez un paquetage `stocklivres` où vous copierez la classe `Livre` (voir Tp4). Vous allez créer une nouvelle classe `StockLivres` chargée de gérer le catalogue de tous les livres (au sens du Tp4) qui sont en stock dans une librairie. Vous pourrez enregistrer la liste de livres en stock, soit dans un tableau, soit dans un arraylist. **Aucune autre structure Java est autorisé dans ce Tp, en particulier vous ne devez pas employer d'autres collections (autre que les arraylist).** A la création, la liste des livres sera vide : on pourra ensuite y ajouter des nouveaux livres, plus d'exemplaires d'un livre déjà existant, ou en retirer. Un stock de livres doit respecter la *propriété de cohérence interne* suivante :

deux objets `Livre` du stock ne peuvent avoir le même titre ET le même auteur.

En revanche, plusieurs livres peuvent avoir le même titre et auteurs différents, ou le même auteur et différents titres. Cette propriété doit être maintenue à tout moment pour un stock : chaque opération doit la préserver.

2.1 Question 1 : conception de la classe

Dans cette question il s'agit de prendre le rôle du concepteur de l'application de gestion du stock *au sein d'une application plus vaste qui gère les services offerts par une librairie*. Il faut garder à l'esprit que la classe `StockLivre` doit :

(a) modéliser les données du stock ;

(b) fournir un ensemble de méthodes qui serviront (dans un prochain Tp) à programmer les méthodes d'une future classe Librairie. Ainsi par exemple, sur un futur objet librairie on voudra rechercher la disponibilité d'un ouvrage, réaliser un achat si l'ouvrage et disponible et dans ce cas, obtenir le prix à payer puis mettre à jour sa disponibilité en stock, etc. L'objet stock devra donc offir des méthodes pour interroger sa liste de livres (faire des recherches), pour y ajouter des données, retirer des exemplaires. Le but final est de *contribuer à l'implantation de services à offrir par la librairie* tout en gardant séparés les traitements sur les données de chaque classe. La classe `StockLivre` fournira les méthodes de gestion du stock, la classe Librairie celle de la librairie (nous y renviendrons dans un prochain Tp).

Pour concevoir la classe `StockLivre` **vous devez établir** :

1. Quelles sont les variables d'instance de la classe et quels sont leurs types ? Comment les initialiser ?
2. Quelles sont les conditions précises que doivent satisfaire ces variables pour valider la cohérence interne mentionnées plus haut ? Comment garantir que les valeurs initiales données aux variables sont cohérentes ?
3. Quelle est la liste de profils de méthodes (nom, type de retour, types des paramètres) que l'on pourra appliquer sur un objet stock, et quel sera le comportement de chacune de ces méthodes décrit précisement :
   — Décrire chaque cas différent, y compris les éventuelles exceptions lévées.
   — On vous conseille fortement d'établir un *squelette de classe* où vous mettrez les profils de vos méthodes (sans leur corps), **accompagnés d'un commentaire au format javadoc** (quelques exemples sont donnés plus bas).

2.2 Question 2 : coder une ébauche de solution

Voici une proposition de conception pour cette classe. Comparez cette solution à la votre, puis écrivez le code de la classe en suivant la propostion donné ici.

1. Côté variables, les livres du stock seront modélisés par un tableau ou par une arraylist d'objets `Livre`. Un constructeur permettra de créer une liste (ou tableau) vide.
2. `affiche()` et `afficheTousAvecTitre()` : affichent respectivement tous les livres du stock et tous les livres ayant un titre passé en paramètre.
3. Écrire une méthode qui retourne le nombre de livres se trouvant au catalogue et une autre méthode qui retourne le nombre d'exemplaires tous livres confondus. Par exemple, s'il y a trois livres différents dans la liste de livres, avec respectivement 2, 3 et 1 exemplaires, la première méthode doit retourner 3 et la deuxième 6.
4. Ecrivez une méthode qui implante la spécification javadoc suivante. Notez que cette méthode est déclarée privée, car on veut que seule les méthodes de la classe StockLivre y aient accès.

```
/**
 * Retourne l'indice d'un livre se trouvant au stock, avec titre et
 * auteur passés en parametre et -1 si non trouvé.
 * @param tit titre du livre recherché
 * @param aut auteur du livre  recherché
 */
 private int indiceTitreAuteur(String tit, String aut)
```

5. Utilisez **impérativement** la méthode `indiceTitreAuteur` afin d'écrire les méthodes suivantes. `existeLivre` teste si un livre avec un titre et auteur donnés existe au catalogue, et `chercheParTitreAuteur` qui pour un titre et auteur donnés retourne l'objet livre trouvé ou null si non trouvé.
6. Écrivez une version surchargée pour les deux méthodes précédentes : elles prendront en paramètre un livre, et la recherche se fera uniquement sur les champs titre et auteur du livre passé en paramètre.
7. Ecrivez une méthode qui prend un livre l en paramètre et l'ajoute au stock si aucun livre de même titre et auteur n'est déjà présent. Autrement, additionne le nombre d'exemplaires de l à ceux du livre déjà en stock.

8. On souhaite écrire une méthode correspondant au retrait du stock d'un certain nombre d'exemplaires d'un ouvrage (par exemple, pour finaliser un achat). La méthode doit retourner le prix total à payer, ou léver l'exception `NbExemplairesInsuffisant` si le nombre d'exemplaires en stock est insuffisant ou encore léver l'exception `OuvrageInexistant` si cet ouvrage n'existe pas au catalogue. S'il y a un nombre d'exemplaires suffisant la méthode met à jour le stock en retirant le nombre d'exemplaires retirés.

2.3 Question 3 : coder une mini application de gestion du stock

Ecrivez un programme qui propose un menu d'opérations sur le stock. Concevez le de manière à :

— tester toutes les fonctionnalités d'un stock : créer un stock vide, y ajouter des nouveaux livres, des exemplaires pour des livres déjà existants, retirer des exemplaires et réaliser des recherchez et affichages. Vous devez veiller à ce que la propriété de cohérence interne d'un stock soit toujours respectée.

— Traiter les principales erreurs pouvant apparaître pendant l'exécution qu'elles soientt liées aux erreurs de saisie, ou aux exceptions pouvant être levées. N'oubliez pas qu'on attend désormais que vos applications soient robustes et ne plantent pas à la moindre erreur.