CNAM Bretagne **2005-2006**

TP Algorithmique programmation

TP 5 : Objets (2)

On souhaite écrire un logiciel permettant la gestion des rayons d'une librairie.

Exercice 1 (La classe Livre).

1. Un livre est caractérisé par un titre, son auteur et le nombre de pages qu'il comporte. Écrire une classe `Livre` comportant :
   - les constructeurs qui vous semblent pertinents
   - un accesseur pour chaque attribut
   - un modificateur pour le nombre de pages : il ne devra changer le nombre de pages que si on lui passe en paramètre un nombre positif, et ne rien faire sinon, en affichant un message d'erreur.
2. Ajoutez une méthode `afficheToi()` qui affiche une description du livre (auteur, titre et nombre de pages).
3. Écrivez une classe `TestLivre`.
4. Dans `TestLivre`, ajoutez l'instruction `System.out.println(livre)` où `livre` désigne un des livres que vous avez créés. Qu'est ce qui s'affiche ? Ajoutez une méthode `toString()` dans la classe `Livre` qui renvoie une chaîne de caractères qui décrit le livre. Donnez à la méthode `toString()` la même signature que la méthode de même nom de la classe `java.lang.Object` (cherchez dans les API du JDK). Relancez le test. Quel affichage obtenez-vous cette fois par l'instruction `System.out.println(livre)` ? Modifiez la méthode `afficheToi()` pour utiliser `toString()`.
5. Ajoutez un prix aux livres ainsi qu'un accesseur et un modificateur pour ce nouvel attribut. Ajoutez au moins un constructeur qui prend le prix en paramètre. Si le prix d'un livre n'a pas été donné, la description du livre (`toString()`) devra indiquer "Prix pas encore donné". Attention, un livre peut être gratuit (0 est une valeur possible pour un prix) ; la valeur -1 indiquera que le prix n'a pas encore été donné.
6. On souhaite bloquer complètement les prix : un prix ne peut être fixé qu'une seule fois et ne peut être modifié ensuite (une tentative pour changer le prix ne fait qu'afficher un message d'erreur). Ajoutez donc une variable booléenne `prixFixe` (pour "prix fixé") qui indiquera que le prix ne peut plus être modifié. Récrivez le modificateur pour l'attribut prix (et autre chose si besoin est).
7. Dans la classe `Livre`, écrivez une méthode d'instance `compare` pour comparer 2 livres par rapport à leur nombre de pages : `compare` prend un livre en paramètre et elle renvoie 0 si le livre a le même nombre de pages que l'instance courante ("`this`"), 1 si l'instance courante a plus de pages que le paramètre et -1 sinon.

Exercice 2 (La classe Etagere). Créez une classe `Etagere` pour représenter une étagère qui peut contenir **au plus** 15 livres. Vous utiliserez pour cela un tableau de `Livre`. Les méthodes existantes seront les suivantes :

- Donner le nombre de livres que peut contenir l'étagère et le nombre de livres qu'elle contient vraiment.
- Ajouter un livre ("`void ajouter(Livre)`"). Vous ajouterez les livres "à la fin" de l'étagère. Il devra être impossible d'ajouter des livres dans une étagère pleine.
- Récupérer un livre dont on donne la position sur l'étagère (le livre reste sur l'étagère, on récupère simplement une référence sur le livre). La méthode renverra une instance de `Livre`. La position du premier livre d'une étagère devra être 1 (et pas 0, bien que le livre soit rangé à la première position du tableau, qui est d'indice 0). La signature de la méthode sera "`Livre getLivre(int)`".
- Chercher sur l'étagère un livre repéré par son titre et son auteur. La méthode renverra la position du livre dans l'étagère (ou 0 si le livre n'y est pas). La signature de la méthode sera "`int chercher(String, String)`". S'il y a plusieurs livres avec le même titre et le même auteur, la méthode renvoie le premier trouvé.
- Avoir une fonctionnalité semblable à la précédente, mais la méthode renvoie un tableau de positions s'il y a plusieurs livres qui ont ce titre et cet auteur. On souhaite appeler cette méthode "chercher" ; est-ce possible ? Le tableau aura pour taille le nombre de livres trouvés (0 si aucun livre n'a été trouvé).
- Enlever des livres (on "tassera" les livres vers le début). Vous écrirez 2 méthodes `supprimerLivre` pour enlever un livre :
  - une méthode qui repérera le livre par sa position (1 pour le premier livre) dans l'étagère (de signature "`Livre enleverLivre(int)`"),
  - une méthode qui repérera le livre par son auteur et son titre, et qui utilisera la méthode `chercherLivre` (de signature "`Livre enleverLivre(String, String)`").

  Les 2 méthodes renverront le livre supprimé (ou null si le livre n'a pas été trouvé).
- écrire une méthode `toString`

Les méthodes écrites seront bien sûr testées au fur et à mesure dans un programme de test.

Exercice 3. Écrire la documentation de ces deux classes et la générer grâce à l'outil javadoc.

Remarque : en temps normal, la documentation se rédige au fur et à mesure de l'écriture des méthodes et pas "à la fin" comme c'est proposé ici.