POO – Programmation Orientée Objet en Java

Fiche de TP numéro 2

On s'occupe de la gestion d'un bibliobus.

La classe Livre

Dans un premier temps, vous devez implémenter une classe `Livre` qui répond au cahier des charges suivant. Un livre est ici à comprendre comme un livre dans un catalogue, pas comme un exemplaire physique d'un livre.

1. Un livre est décrit par cinq informations : son titre, le nom de son auteur, son éditeur, le nombre d'exemplaires présents dans la bibliothèque, et son genre. Le genre peut prendre sa valeur uniquement parmi *littérature française*, *littérature jeunesse*, *littérature étrangère*, *policier*, *politique*, *sciences*, *sciences humaines*, *non spécifié* ;
2. un livre a toujours un titre, un auteur et un éditeur qui ne peuvent pas changer ;
3. le nombre d'exemplaires ne peut pas être négatif. On peut ajouter soit un seul exemplaire à la fois, soit plusieurs exemplaires ;
4. la perte d'un livre étant assez rare, on ne peut signaler que la perte d'un exemplaire à la fois ;
5. on veut savoir si un livre est réellement présent dans le bibliobus, i.e. s'il y en a au moins un exemplaire ;
6. un livre est capable de s'identifier, en affichant les informations qui le caractérisent ;
7. on veut savoir si deux livres sont représentatifs de la même oeuvre (même titre, même auteur, même éditeur) ;
8. lorsqu'un livre change d'éditeur, on doit créer un nouveau livre avec le même titre, le même auteur, le même genre et le nouvel éditeur (et un seul exemplaire).

Exercice 1 : Classe Livre, classe DemoLivre

Commencez par créer deux fichiers contenant respectivement une classe `Livre` et une classe `DemoLivre`. Dans la classe `Livre`, écrivez tous les attributs nécessaires. Dans la classe `DemoLivre` implémentez seulement une méthode `main`. **Après chaque écriture d'une nouvelle méthode ou d'un constructeur pour la classe `Livre`, vous devrez tester votre implémentation dans la classe `DemoLivre`.**

Exercice 2 : Constructeurs

Écrivez un constructeur pour vérifier la première partie du point 2). Tant que le genre d'un livre n'est pas précisé, il contient la valeur *"Non spécifié"*. Utilisez une énumération pour le genre d'un livre.

Comment permettre de spécifier le genre d'un livre à sa création ?

Exercice 3 : Accesseurs

Comment être sûr que ni le titre, ni l'auteur ni l'éditeur du livre ne peuvent être modifiés de l'extérieur ? Implémentez tous les accesseurs en lecture et en écriture nécessaires (et uniquement ceux-là).

Exercice 4 : Achat d'exemplaires, perte d'un livre

Implémentez les méthodes `ajouteExemplaires()`, `ajouteExemplaires(int nb)` pour l'ajout d'un ou de `nb` exemplaires d'un livre, et la méthode `supprimeExemplaire()`.

Exercice 5 : Donner ses caractéristiques
Implémentez la méthode `toString()` qui retourne une chaîne de caractères. Cette méthode construit la chaîne de caractères qu'on souhaiterait voir si on veut afficher toutes les caractéristiques d'un livre.

Exercice 6 : Est présent dans le bibliobus ?
Implémentez la méthode `estPresent()` qui retourne un booléen. Cette méthode retourne vrai s'il y a au moins un exemplaire de ce livre.

Exercice 7 : Même livre
On dira que deux livres sont égaux s'ils ont le même titre, le même auteur et le même éditeur. Implémentez la méthode `equals(Livre l)` qui retourne un booléen. Cette méthode retourne vrai si le livre passé en paramètre et l'instance courante (celui qui exécute la méthode) sont les mêmes, faux sinon.

Exercice 8 : Changement d'éditeur
Implémentez la méthode `nouvelEditeur(String unEditeur)` qui retourne une nouvelle instance de `Livre` ayant le même titre, le même auteur et le même genre que le livre qui exécute cette méthode.

Le Bibliobus

On désire maintenant gérer le bibliobus. Dans un premier temps, on se limitera aux opérations suivantes :

- retourner les caractéristiques d'un livre dont on connaît l'identifiant ;
- afficher les caractéristiques d'un livre dont on connaît l'identifiant ;
- afficher la liste des livres du catalogue du bibliobus (i.e., les livres connus du bibliobus) ;
- afficher la liste des livres réellement présents dans le bibliobus ;
- afficher la liste des livres d'un genre donné ;
- indiquer si un livre donné est connu du bibliobus ou non ;
- indiquer si un livre donné est réellement présent dans le bibliobus ou non (à partir de son identifiant, ou d'une instance d'un livre) ;
- donner le nombre d'exemplaires d'un livre particulier (à partir de son identifiant, ou d'une instance d'un livre) ;
- faire sortir un livre définitivement du catalogue ;
- ajouter un livre au bibliobus.

Le nombre maximal de livres que peut contenir le bibliobus est fixé. L'ajout d'un livre ne peut se faire que s'il reste de la place. Pour le bibliothécaire, chaque livre est repéré par son indice dans les rayons : on utilisera un tableau pour stocker les livres dans la bibliothèque. Cet indice sera l'identifiant du livre. Lors de la sortie définitive d'un livre, on pensera à *resserrer* les livres, par exemple en rangeant le dernier livre dans le *trou*. Les nouveaux livres seront ainsi ajoutés toujours en fin de tableau.

Comme on peut avoir plusieurs bibliobus sur une commune, chaque bibliobus possède un nom (qui ne peut pas changer).

Exercice 9 : classe Bibliobus, classe DemoBibliobus
Commencez par créer deux fichiers contenant respectivement une classe `Bibliobus` et une classe `DemoBibliobus`. Dans la classe `Bibliobus`, écrivez tous les attributs nécessaires. Dans la classe `DemoBibliobus` implémentez seulement une méthode `main`. **Après chaque écriture d'une nouvelle méthode ou d'un constructeur pour la classe `Bibliobus`, vous devrez tester votre implémentation dans la classe `DemoBibliobus`.**

Exercice 10 : Constructeurs
Par défaut, un bibliobus peut contenir 100 livres dans son catalogue. Mais il doit être possible de définir

à la création du bibliobus la taille du catalogue. Le nom du bibliobus est toujours indiqué à sa création. Implémentez les constructeurs nécessaires.

Exercice 11 : Un livre appartient-il au bibliobus ?
Implémentez les méthodes `int indiceLivre(String titre, String auteur, String editeur)` et `boolean appartient(String titre, String auteur, String editeur)`. La première retourne l'indice du livre dans le bibliobus s'il y est (-1 si le livre n'existe pas), la seconde renvoie juste un boolean qui indique si le livre est répertorié dans le bibliobus.

Exercice 12 : Ajout d'un livre
Implémentez les méthodes `boolean ajouteLivre(String titre, String auteur, String editeur, Genre genre)` et `boolean ajouteLivre(String titre, String auteur, String editeur, Genre genre, int nbreEx)` qui permettent d'ajouter un livre au catalogue, dans la limite des places disponibles et si le livre n'existe pas. Si le livre est déjà présent, alors on augmente son nombre d'exemplaires.

Exercice 13 : Afficher le catalogue
Implémentez la méthode `afficheCatalogue()` qui affiche sur la console le nom du bibliobus, et toutes les informations sur son catalogue : identifiant de chaque livre et caractéristiques du livre.

Exercice 14 : Accès aux caractéristiques d'un livre
Implémentez les méthodes `getTitre(int identifiant)`, `getAuteur(int identifiant)`, `getEditeur(int identifiant)`, `getGenre(int identifiant)` et `getNbExemplaires(int identifiant)` qui retournent les caractéristiques d'un livre indiqué par son identifiant.

Exercice 15 : Afficher les caractéristiques d'un livre
Implémentez la méthode `afficheLivre(int identifiant)` qui affiche sur la console toutes les caractéristiques d'un livre indiqué par son identifiant.

Exercice 16 : À vous !
Implémentez toutes les autres méthodes du cahier des charges. À vous de déterminer leur signature !

Exercice 17 : Fusion de stocks
Implémentez la méthode `fusion(Bibliobus bib)` qui fusionne le stock de `bib` avec le stock du bibliobus courant. Les livres sont progressivement supprimés de `bib`. Comment faire pour pousser les murs et accepter plus de livres qu'actuellement (pas d'inquiétude, vous avez du budget pour de nouvelles étagères) ? On considèrera que si vous avez besoin de plus de places qu'actuellement, vous ajoutez 100 places à votre dimension actuelle.

Exercice 18 : Comparer deux livres
Dans la classe `Livre`, implémentez une méthode `boolean compareTo(Livre unLivre)` qui indique si le livre courant est plus petit ou plus grand, en un certain sens, que le livre `unLivre`. On dira qu'un livre est plus petit qu'un autre si son genre est placé avant dans l'énumération, puis, pour deux mêmes genres, si le nom de son auteur est inférieur dans l'ordre alphabétique que l'auteur de `unLivre`, puis vous comparerez les titres. La méthode retourne `-1` si le livre courant est plus petit, `1` s'il est plus grand, `0` s'ils sont équivalents. On accède à la position d'un type énuméré par sa méthode `ordinal()` (qui retourne un entier).

Cette méthode `compareTo` existe déjà pour comparer deux chaînes de caractères. Vous aurez besoind e l'utiliser dans votre méthode.

Exercice 19 : Trier les livres
Implémentez la méthode de tri de votre choix pour ranger le bibliobus.