Travail pratique #3

Ce travail doit être fait individuellement.

AUCUN RETARD ACCEPTÉ

Notions mises en pratique : le respect d'un ensemble de spécifications, les classes et les objets, et l'utilisation tableaux.

1. Spécifications

Ce travail consiste à définir la classe `Bibliotheque` en utilisant la classe `Livre` (fournie). La classe `Bibliotheque` modélise une collection de livres (une bibliothèque), et offre des méthodes permettant de gérer cette collection : y ajouter de livres, y supprimer des livres, y faire des recherches par catégorie(s), etc.

1.1 La classe Livre

Cette classe vous est fournie. Vous ne **DEVEZ PAS** la modifier, car elle sera utilisée telle quelle pour tester votre classe `Bibliotheque`. Veillez à bien lire cette classe, à comprendre les méthodes publiques qu'elle offre, car vous aurez à les utiliser pour implémenter votre classe Bibliotheque.

1.2 La classe Bibliotheque

Vous devez définir la classe `Bibliotheque` qui modélise une bibliothèque contenant une collection de livres représentée en mémoire par un tableau de `Livre`. De plus, l'attribut `nbrLivres` servira à conserver le nombre de livres effectivement dans cette bibliothèque.

Tous les livres de cette bibliothèque doivent en tout temps se trouver aux indices 0 à `nbrLivres` - 1 du tableau de livres. Une bibliothèque vide est une bibliothèque qui ne contient aucun livre (`nbrLivres` = 0).

Précisions quant à l'ajout de livres à la bibliothèque (au tableau de livres de cette bibliothèque) :

- Les livres sont ajoutés l'un à la suite de l'autre : les livres dans le tableau de livres sont stockés dans l'ordre où ils sont ajoutés. Par exemple, si l'on ajoute, dans une bibliothèque vide, les 3 livres l1, l2, et l3 dans cet ordre, ces 3 livres se trouveront aux indices 0 à 2, dans ce même ordre. Si l'on ajoute ensuite le livre l4, les livres l1, l2, l3, l4 occuperont les indices de 0 à 3 du tableau de livres, dans cet ordre, etc. On ajoute en fait toujours à l'indice `nbrLivres` (avant de l'incrémenter après l'ajout).
- La longueur initiale du tableau de livres, à la création d'une bibliothèque vide, est de 4.
- Chaque fois que le tableau de livres est plein, on l'agrandit de 4 cases supplémentaires lorsqu'on veut y ajouter un autre livre.

Précisions quant à la suppression de livres de la bibliothèque (dans le tableau de livres de cette bibliothèque) :

- Pour supprimer un livre à l'indice i, il faut décaler tous les livres du tableau de livres, qui viennent après i, d'une position vers la gauche.

***** Voir l'exemple de la p.19 des notes de cours sur les tableaux (qu'on a vu en classe). *****

Notez que dans le cadre de ce travail, vous n'avez pas à faire d'application, mais éventuellement, vous pourriez utiliser la classe `Bibliotheque` pour faire un programme de gestion d'une bibliothèque offrant des options de menu pour ajouter et supprimer des livres dans la bibliothèque, afficher les livres de la bibliothèque, faire des recherches par catégories dans la bibliothèque, etc.

Les sections suivantes décrivent les membres publics que vous devez implémenter. Utilisez AUTANT QUE POSSIBLE les membres publics contenus dans la classe `Livre`.

1.1.1 Constante de classe

Nom de l'attribut	Type	Description
`NBR_CASES`	`int`	**Doit être initialisé à 4**. Constante utilisée pour assigner la longueur initiale du tableau de livres, ainsi que pour agrandir ce tableau lorsque requis.

Notez que vous pouvez ajouter d'autres constantes de classe si vous le jugez pertinent.

1.1.2 Attributs d'instance

Nom de l'attribut	Type	Description
`livres`	`Livre[]`	Le tableau contenant les livres de cette bibliothèque.
`nbrLivres`	`int`	Le nombre de livres dans cette bibliothèque.

Respectez à la lettre le nom et le type donné pour les attributs, sinon les tests ne compileront peut-être pas, et vous risquez de perdre des points.

Notez que vous ne devez pas ajouter d'autres attributs d'instance (aucune autre variable globale n'est permise).

ATTENTION, pour les constructeurs et méthodes suivantes (sections 1.1.3 et 1.1.4) :

- Respectez à la lettre le nom donné à chaque méthode.
- Lorsqu'il y a plus d'un paramètre, votre implémentation doit respecter l'ordre d'énumération dans les tableaux décrivant ces paramètres.

1.1.3 Constructeurs (2)

Constructeur sans paramètre

Ce constructeur construit une bibliothèque vide (ne contenant aucun livre). L'attribut `livres` est initialisé avec une longueur égale à `NBR_CASES`.

Constructeur à un paramètre

***NOTE** : Codez les méthodes **`ajouterLivre`**`(Livre)` et **`livreExiste`**`(Livre)` décrites plus bas AVANT de coder ce constructeur.*

Paramètre	Type	Description
`lesLivres`	`Livre[]`	Un tableau contenant les livres à ajouter à cette bibliothèque.

Ce constructeur initialise d'abord l'attribut `livres` avec une longueur de `NBR_CASES` et l'attribut `nbrLivres` à 0. Il ajoute ensuite chaque livre non `null` contenu dans le tableau `lesLivres` à cette bibliothèque, un par un. Attention, ce constructeur ne doit ajouter aucun doublon à cette bibliothèque, il faut donc vérifier, avant l'ajout d'un livre, que celui-ci n'existe pas déjà dans cette bibliothèque. N'oubliez pas d'ajuster l'attribut `nbrLivres` s'il y a lieu.

Précisions :

- Pour ajouter les livres, un par un, à cette bibliothèque, utilisez la méthode `ajouterLivre(Livre)` décrite plus bas. Cette méthode s'occupe déjà de ne pas ajouter les livres `null` ou qui existent déjà dans cette bibliothèque, et d'incrémenter l'attribut `nbrLivres`, s'il y a lieu.
- Les livres (non `null` et non-doublon) doivent être ajoutés à cette bibliothèque, l'un à la suite de l'autre, dans l'ordre où ils se présentent dans le tableau `lesLivres`.
- Lorsque le tableau `livres` est plein, et qu'il reste des livres à ajouter, il faut l'agrandir de `NBR_CASES` avant de poursuivre l'ajout (la méthode `ajouterLivre(Livre)` s'occupe déjà d'agrandir le tableau si nécessaire).

1.1.4 Méthodes d'instance publiques

Nom méthode : `getNbrLivres (getter)`
Type retour : `int`
Paramètre : `Aucun`

Retourne le nombre de livres dans cette bibliothèque.

Nom méthode : `livreExiste`
Type retour : `boolean`

Paramètre	Type	Description
`livre`	`Livre`	Le livre dont on veut tester l'existence.

Retourne vrai si le livre donné en paramètre existe déjà dans cette bibliothèque, faux sinon. Autrement dit, on veut savoir si un livre égal au livre donné en paramètre appartient déjà à cette bibliothèque. Un livre est considéré comme égal à un autre livre s'il a exactement le même titre (peu importe la casse), le même nom d'auteur (peu importe la casse), et la même date de publication. Utilisez la méthode `estEgal(Livre)` de la classe `Livre` pour tester l'égalité entre deux livres.

Nom méthode : `ajouterLivre`
Type retour : `boolean`

Paramètre	Type	Description
`livre`	`Livre`	Le livre à ajouter à cette bibliothèque.

Ajoute, si possible, le livre donné en paramètre à cette bibliothèque. Si l'ajout est effectué, la méthode retourne `true`, sinon elle retourne `false`.

Précisions:

- Si le livre donné en paramètre est `null` ou s'il existe déjà dans cette bibliothèque, il n'est pas ajouté.
- Si le tableau `livres` est plein avant l'ajout, et que l'ajout peut être effectué, la méthode doit agrandir le tableau `livres` de `NBR_CASES` avant d'y ajouter le livre.
- Ne pas oublier d'incrémenter l'attribut `nbrLivres` lorsqu'il y a lieu.

Nom méthode : `ajouterLivre`
Type retour : `boolean`

Paramètres	Type	Description
`titre`	`String`	Le titre du livre à ajouter à cette bibliothèque.
`auteur`	`String`	Le nom de l'auteur du livre à ajouter à cette bibliothèque.
`anneePub`	`int`	L'année de publication du livre à ajouter à cette bibliothèque.
`numCategories`	`int[]`	La liste des numéros des catégories associées au livre à ajouter à cette bibliothèque.

Ajoute, si possible, le livre ayant les informations données en paramètres à cette bibliothèque. Si l'ajout est effectué, la méthode retourne `true`, sinon elle retourne `false`.

Précisions:

- Vous n'avez pas à vérifier la validité des paramètres `titre`, `auteur`, et `anneePub` avant de créer le livre à ajouter, car ceux-ci seront traités par le constructeur de la classe `Livre`.

- Si le livre à ajouter existe déjà dans cette bibliothèque, il n'est pas ajouté.
- Les numéros de catégorie donnés dans le tableau `numCategories` doivent être associés au livre à ajouter à cette bibliothèque : seuls les numéros de catégories valides doivent être associés au livre, et les numéros de catégories associés au livre ne doivent pas contenir de doublons.

Nom méthode : `obtenirLivre`
Type retour : `Livre`

Paramètre	Type	Description
`iemeLivre`	`int`	L'indice du livre à retourner dans le tableau `livres` de cette bibliothèque.

Retourne le livre à la position `iemeLivre` dans le tableau `livres` de cette bibliothèque. Si `iemeLivre` n'est pas un indice valide, la méthode retourne `null`.

Précision : un indice valide se trouve entre 0 et `nbrLivres` - 1 inclusivement.

Nom méthode : `supprimerLivre`
Type retour : `Livre`

Paramètre	Type	Description
`iemeLivre`	`int`	L'indice du livre à supprimer dans le tableau `livres` de cette bibliothèque.

Supprime le livre à la position `iemeLivre` dans le tableau `livres` de cette bibliothèque. Si `iemeLivre` est un indice valide, la suppression a lieu, et le livre supprimé est retourné. Sinon, la méthode retourne `null` pour signaler que la suppression n'a pas eu lieu.

Précisions:
- Un indice valide se trouve entre 0 et `nbrLivres` - 1 inclusivement.
- N'oubliez pas de décrémenter l'attribut `nbrLivres` s'il y a lieu.

Nom méthode : `rechercherParConjonctionDeCategories`
Type retour : `Livre[]`

Paramètre	Type	Description
`numCategories`	`int[]`	Un tableau contenant les numéros de catégorie à considérer pour la recherche.

Retourne un tableau contenant les livres retournés par la recherche par conjonction (ET) des numéros de catégorie valides présents dans `numCategories`. Autrement dit, les livres retournés doivent être ceux qui sont associés à tous les numéros de catégorie valides présents dans `numCategories`.

Précisions:
- Le tableau retourné doit être **de longueur minimale**. Par exemple, si le résultat de la recherche contient 3 livres, le tableau retourné doit être de longueur 3, si la recherche ne retourne aucun livre, le tableau retourné doit être de longueur 0, etc.
- Seuls les numéros de catégorie valides présents dans le tableau `numCategories` doivent être considérés pour la recherche.
- Si `numCategories` est `null` ou ne contient aucun numéro de catégorie valide, la recherche retourne tous les livres contenus dans cette bibliothèque.
- Les livres dans le tableau de livres retourné par la méthode doivent être dans le même ordre que dans le tableau `livres` de cette bibliothèque.

Nom méthode : `rechercherParDisjonctionDeCategories`
Type retour : `Livre[]`

Paramètre	Type	Description
`numCategories`	`int[]`	Un tableau contenant les numéros de catégorie à considérer pour la recherche.

Retourne un tableau contenant les livres retournés par la recherche par **disjonction** (OU) des numéros de catégorie valides présents dans `numCategories`. Autrement dit, les livres retournés doivent être ceux qui sont associés à au moins un des numéros de catégorie valides présents dans `numCategories`.

Précisions:

- Le tableau retourné doit être de longueur minimale. Par exemple, si le résultat de la recherche contient 3 livres, le tableau retourné doit être de longueur 3, si la recherche ne retourne aucun livre, le tableau retourné doit être de longueur 0, etc.
- Seuls les numéros de catégorie valides présents dans le tableau `numCategories` doivent être considérés pour la recherche.
- Si `numCategories` est `null` ou ne contient aucun numéro de catégorie valide, la recherche ne retourne aucun livre.
- Les livres dans le tableau de livres retourné par la méthode doivent être dans le même ordre que dans le tableau `livres` de cette bibliothèque.

1.1.5 Spécifications supplémentaires

- Vous devez respecter le principe d'encapsulation.
- Les seules variables globales permises sont les 2 attributs d'instance donnés à la section 1.1.2.
- Les méthodes énumérées dans cette section (1.1) sont les seules méthodes publiques permises dans la classe `Bibliotheque`. Vous pouvez et devriez cependant ajouter vos propres méthodes **privées** pour bien découper vos méthodes (séparation fonctionnelle).

2.Précisions

- Votre classe à remettre doit se trouver dans le paquetage par défaut.
- Assurez-vous que vos fichiers à remettre sont **encodés en UTF8**.
- Prenez soin de bien modulariser votre code à l'aide de méthodes cohésives, spécialisées, et bien paramétrées. Voici quelques lignes directrices pour vous guider dans la conception de vos méthodes :
  - Chaque fois que vous vous apercevez que différentes parties de votre code se ressemblent énormément, essayez d'en faire une méthode bien paramétrée (si possible) => évitez la répétition du code.
  - Faites une méthode pour chaque petite tâche spécifique à accomplir dans la résolution du problème.
- Vous DEVEZ respecter le style Java.
- Pensez à documenter correctement (Javadoc) chacune de vos méthodes (`public` ou `private`).
- **N'oubliez pas d'écrire (entre autres) votre nom complet, et votre code permanent dans l'entête de votre classe à remettre.**
- **Le non-respect de toute spécification ou consigne se trouvant dans l'énoncé du TP est susceptible d'engendrer une perte de points.**

Note : *Si quelque chose est ambigu, obscure, s'il manque de l'information, si vous ne comprenez pas les spécifications, si vous avez des doutes… vous avez la responsabilité de vous informer auprès de votre enseignante.*

3. Détails sur la correction

3.1 La qualité du code (40 points)

Concernant les critères de correction du code, lisez attentivement le document "**critères généraux de correction du code Java**" dans la section CONTENU DU COURS / Travaux, sur Moodle. De plus, votre code sera noté sur le respect du style Java, dont un résumé se trouve aussi dans la section CONTENU DU COURS / Travaux, sur Moodle.

Note : *Votre code sera corrigé sur la totalité ou une partie des critères de correction indiqués ci-dessus, ainsi que sur le respect des spécifications/consignes mentionnées dans ce document.*

3.2 L'exécution (60 points)

Un travail qui ne compile pas se verra attribuer la note 0 pour l'exécution.

Note : *Votre code sera testé en tout ou en partie. Les points seront calculés sur les tests effectués. Ceci signifie que si votre code fonctionne dans un cas x et que ce cas x n'est pas testé, vous n'obtiendrez pas de points pour ce cas. Il est donc dans votre intérêt de vous assurer du bon fonctionnement de votre programme dans tous les cas possible.*

4. Date et modalité de remise

4.1 Remise

Date de remise : 17 avril 2019 AVANT minuit

Le fichier à remettre : `Bibliotheque.java` – **NE PAS** REMETTRE votre fichierr dans une archive (ZIP, RAR…).

Remise via Moodle uniquement.

Vous devez remettre (téléverser) votre fichier sur le site du cours (Moodle). Vous trouverez la boîte de remise dans la section **BOÎTES DE REMISE - TRAVAUX PRATIQUES / REMISE DU TP3**.

4.2 Politique concernant les retards

AUCUN RETARD NE SERA ACCEPTÉ

4.3 Remarques générales

- Aucun programme reçu par courriel ne sera accepté.
- Vous avez la responsabilité de conserver des copies de sauvegarde de votre travail (sur disque externe, Dropbox, Google Drive, etc.). La perte d'un travail due à un vol, un accident, un bris… n'est pas une raison valable pour obtenir une extension pour la remise de votre travail.

Le règlement sur le plagiat sera appliqué sans exception. Vous devez ainsi vous assurer de ne pas échanger du code avec des collègues ni de laisser sans surveillance votre travail au laboratoire. Vous devez également récupérer rapidement toutes vos impressions de programme au laboratoire.

BON TRAVAIL !