Uni ersité Paris Diderot **Programmation Orientée Objet**
L3 **Année 2012-2013**

TP n°4

Champs statiques et Introduction à l'héritage

On modélise une application devant servir à l'inventaire d'une bibliothèque. La classe `Bibliotheque` contiendra la seule méthode `main` pour tester les différentes classes de ce TP.

Exercice 1 Tous les documents possèdent un *titre*. Quand un document est créé, son titre est donné à la création et par la suite, il ne change plus.

1. Définir la classe `Document` avec son constructeur public, et la propriété `titre` privée et son accesseur.
2. On veut attribuer un *numéro d'enregistrement* unique dès que l'on crée un objet `Document`. On veut que le premier document créé ait le numéro 0, et que ce numéro s'incrémente de 1 à chaque création de document. Quelles propriétés faut-il ajouter à la classe `Document` ? Lesquelles doivent-être `static` ? Les ajouter sans leurs accesseurs, et modifier le constructeur. Puis ajouter une méthode `getNumero` renvoyant le numéro d'enregistrement du document.
3. Ajouter un second constructeur public permettant de définir aussi, en plus du titre, le numéro d'enregistrement (supposé positif ou nul) lors de la création d'un document. Quelles restrictions peut-on prendre afin d'éviter que deux documents créés aient le même numéro ? Dans les cas de restriction, on prendra le numéro suivant prévu, à la place du numéro indiqué par l'utilisateur.
4. Redéfinir la méthode

3. Définissez ensuite quelques classes supplémentaires, par exemple `BandeDessinee`, `Manga`, `DictionnaireBilingue`, etc. avec des propriétés en plus. Pour chacune de ces nouvelles classes, quelle classe étend-elle ?

Exercice 3 On veut implémenter une liste de documents dans laquelle on veut pouvoir ajouter des documents petit à petit, et y accéder par leur numéro d'enregistrement.

Pour implémenter cette structure, on va utiliser la classe prédéfinie de Java `LinkedList` qui implémente des listes simplement chaînées d'objets. Pour pouvoir utiliser cette classe , il faut l'importer en précisant `import java.util.LinkedList;` au début de votre fichier. Avant de vous mettre à programmer, voilà quelques instructions sur comment utiliser cette classe :

- Pour créer une liste `li` contenant des objets appartenant à la classe `Exemple` on procède la façon suivante :
  `LinkedList<ExempleClasse> li=new LinkedList<ExempleClasse>();`
- La méthode `int size()` permet de récupérer le nombre d'éléments dans la liste, en faisant par exemple :
  `int s=li.size();`
- La méthode `void add(E element)` ajoute l'objet `element` de la classe `E` en queue de liste (**Attention :** `E` doit aussi être le type avec lequel vous avez construit la liste, ici `ExempleClasse`).
- La méthode `E get(int i)` renvoie le i-ème élément de la liste.
- La méthode `E remove(int i)` enlève le i-ème élément de la liste et le renvoie.
- La méthode `E add(int i,E element)` insère `element` à la i-ème position dans la liste et décale tous les éléments successeurs en augmentant leur position dans la liste d'une unité.

Normalement ces méthodes devraient vous suffire pour ce TP, toutefois n'hésitez pas à être curieux et allez visiter la description de la classe `LinkedList` à l'URL suivante `http://download.oracle.com/javase/6/docs/api/java/util/LinkedList.html`.

On va ici créer une classe `ListeDeDocuments` avec un champ `LinkedList` seravant à stocker les documents de la liste.

1. Définir la classe `ListeDeDocuments` avec la liste privée `liste`. Écrire une méthode publique `ajouterDocument` qui prend en argument le titre d'un document, crée ce document et l'ajoute à la liste. Écrire une méthode `ajouterDocument` qui prend en argument un document et l'ajoute à la liste. Est-ce un problème si ces deux méthodes ont le même nom ?
2. Écrire une méthode `getDocument` qui prend en argument un numéro et renvoie le document de la liste ayant ce numéro (`null` s'il n'est pas dans la liste).
3. Écrire une méthode `enleverDocument` qui prend en argument un numéro de document et le retire de la liste.
4. Dans cette classe, redéfinir la méthode `toString` pour qu'elle renvoie les descriptions de tous les documents de la liste, séparées par un saut de ligne `"\n"`.
5. Définissez une méthode `tousLesAuteurs` qui renvoie la liste de tous les auteurs des livres de la liste de documents. Pour ce faire, on peut utiliser l'opérateur `instanceof` qui est inhérent à tout objet et qui prend en argument un nom de

classe et renvoie `true` si l'objet appartient à cette classe. Par exemple pour tester si un objet `obj` de la classe `Document` est aussi un objet de la classe `Livre`, on écrit `obj instanceof Livre`.

6. Définissez une méthode `tousLesDicos` qui renvoie une liste de documents (sous la forme d'une nouvelle instance de la classe `ListeDeDocuments`) contenant tous les dictionnaires de la liste.
7. Écrire une méthode `void tri()` dont le but est de trier la liste des documents par ordre croissant de leur numéro d'enregistrement. Le numéro le plus élevé se retrouvant ainsi en queue de liste.
8. Modifier la classe `Document` de telle façon qu'à chaque fois qu'un nouveau document est créé il soit enregistré dans une liste automatiquement. Pour cela s'inspirer de ce qui a été fait dans le premier exercice avec les numéros d'enregistrement. Ajouter également å la classe `Document` une méthode statique `accesDocument` qui affiche la liste des documents enregistrés.
9. Écrire un programme `main` dans la classe `Bibliotheque` qui teste ces différentes méthodes.