Module M2103 – Java
TP 8 : *API Java Collections*

L'objectif de ce TP est la maîtrise de l'API Java Collections ainsi que l'écriture de code utilisant des itérateurs et le *for* optimisé.

Partie I : `ArrayList` et `LinkedList`

Modifier la classe `CatalogueBibliotheque` de manière à utiliser une `LinkedList` au lieu d'un `ArrayList` afin de stocker les documents de la librairie. Quels sont les changements à effectuer ? Dans le cadre du parcours de la structure, utiliser un itérateur ainsi qu'une boucle *for* optimisée.

Partie II : Evaluation de performance

En utilisant la classe `DocBibliotheque`, on évaluera les performances relatives de certaines Collections Java pour des tâches de recherche et d'insertion d'éléments.
On considèrera les collections `ArrayList`, `LinkedList` et `HashSet`. Chacune possède un constructeur sans paramètres, des méthodes **void add(Object)** et **boolean contains(Object)**. Consulter tout d'abord la documentation puis écrire une classe pour tester la performance de ces trois collections. Cette classe test doit créer quelques milliers d'objets `DocBibliotheque` et les insérer dans chacune de ces collections. Les opérations d'insertion et de recherche d'objets seront chronométrées (cf. méthodes correspondantes).
La classe test (dont un squelette est fourni dans le fichier **TestPerfCollections.java**) mettra en oeuvre les opérations suivantes :

- Création d'un tableau de 10,000 objets `DocBibliotheque`
- Création d'une instance de chacune des collections
- Insertion des objets `DocBibliotheque` dans chacune des collections
- Evaluation de la performance de recherche de ces objets `DocBibliotheque` pour chaque collection. La méthode **contains** sera ici particulièrement adaptée.

Partie III : Set et Map

Tâche 1

Faire en sorte que la classe `DocBibliotheque` implémente l'interface `Comparable` et proposer une implémentation de la méthode **compareTo** comparant des documents par l'intermédiaire de leur code d'archivage.

Ecrire une classe test qui créera un `Set` implémenté par un `TreeSet`. Ajouter plusieurs objets `DocBibliotheque` au `TreeSet` et utiliser un itérateur pour les afficher de manière à tester l'implémentation de la méthode **compareTo**.

Tâche 2

Ecrire un comparateur sur mesure pour les objets `DocBibliotheque` qui les ordonne selon leur titre. Tester votre implémentation.

Tâche 3

Ecrire une classe test qui créera une `Map` implémentée par un `TreeMap`. Ajouter plusieurs objets `DocBibliotheque` en utilisant leur code d'archivage comme clé. Utiliser la documentation API de `Map` pour récupérer un `Set` des valeurs du `TreeMap` et utiliser un itérateur afin de les afficher à l'écran. Dans quel ordre sont-ils affichés ?