JAVA

EXERCICES

Travail sur les Collections

Créez un package collections afin d'y mettre toutes les classes de l'exercice.

A. Etude de performance

L'objectif de ce TP est de manipuler et de comparer l'usage des différentes classes de Collections (List et Set)

1. int et Integer

En Java, il existe des types de base **(int, long, float, double, boolean,**…) et leurs équivalents sous forme d'objets (**Integer, Long, Double, Boolean**…).
Examiner dans la documentation, la classe Integer.
Quels sont les avantages et les inconvénients de manipuler des entiers sous la forme d'objets plutôt que de types simples ?

2. Ecrire un programme Java permettant de créer la liste des multiples de 3 inférieurs à 9999, dans un objet de la classe **ArrayList***. Afficher cette liste en utilisant la méthode toString de cette liste.*

3. Mesures du temps écoulé et de la mémoire utilisée.

Utiliser la méthode non statique **freeMemory()** de la classe **Runtime** pour mesurer la mémoire utilisée et la méthode statique **System.currentTimeMillis()** pour calculer le temps de calcul du programme précédent. N'hésitez pas à consulter la documentation Java pour connaître l'usage exact de ces méthodes. NB : selon la puissance de votre ordinateur, n'hésitez pas à modifier la valeur 9999 pour obtenir des résultats significatifs.

4. Ajouter au programme les instructions nécessaires à la suppression dans la liste des multiples de 3, un par un, dans l'ordre décroissant, de la valeur 9999 jusqu'à 3. Mesurer le temps nécessaire à cette suppression et la mémoire utilisée à la fin de cette suppression. Pourquoi ne revient-on pas à la valeur mémoire initiale ?

Utiliser la méthode **gc()** de la classe **Runtime** (voir la documentation de cette méthode) entre la suppression des éléments de la liste et le calcul de la mémoire utilisée. Que constatez-vous ?

5. Transformez le programme en définissant une méthode statique ***testCollection(Collection c).*** *Dans le programme principal, lancer cette méthode en lui passant successivement comme paramètre une* ***ArrayList****, une* ***LinkedList****, un* ***TreeSet*** *puis finalement un* ***HashSet****.*

Comparer les performances respectives (mémoire et temps de calcul) de ces 4 structures de données. Comment expliquez-vous de telles différences de performances ?

6. Les structures ArrayList et LinkedList semblent moins performantes que les structures Set. Par contre, quelles fonctionnalités possèdent les listes que n'ont pas les Set ?

Illustrer ces fonctionnalités en cherchant le 100ème élément de la liste.
On prendra soin de tester chaque exercice pour en valider le résultat.

B. Une classe LigneBrisee

On veut recréer une classe LigneBrisee (vue précédemment dans les exercices sur les objets) qui contiendra une liste de points, mais stockée maintenant dans une collection. Vous essaierez l'exercice avec les différents types de collection.

1. *Écrire la classe* **LigneBrisee,** *contenant la liste de points, avec ses accesseurs, mutateurs et* **toString***, listant les points de la ligne. Une instance contiendra au moins 2 points au départ.*
2. *Écrire une méthode* **contientPoint** *qui renvoie vrai si le point passé en paramètre est dans la ligne brisée et faux sinon.*
3. *Écrire une méthode* **addPoint** *permettant d'ajouter un point, à condition qu'il ne soit pas déjà dedans.*
4. *Écrire une méthode* **nbPoints** *permettant d'afficher le nombre de points de la ligne brisée.*
5. *Écrire une méthode* **deletePoint** *permettant de supprimer un point.*
6. *Ecrire une classe Main qui permettra de tester la création de la ligne brisée, l'ajout et la suppression d'éléments, avec à chaque fois un affichage de la ligne brisée.*
7. *Quelles sont les avantages inconvénients des collections testées ? Et par rapport à la classe LigneBrisée gérée par un tableau ?*

C. Boisson et cocktails

On veut créer, par la suite, un bar qui peut créer, réaliser et gérer des cocktails.
On a pour cela une classe **Boisson** qui aura un nom, une contenance et un prix.
D'elle hérite une classe **BoissonAlcoolisee** qui aura en plus un degré d'alcool.
Héritée aussi, la classe **BoissonNonAlcoolisee** aura elle un degré de sucre et non d'alcool.
On crée alors une classe **Cocktail** qui contient une liste (une collection) de boissons. On calculera son prix, son degré d'alcool et de sucre automatiquement (pas de setter, que des getter). Le prix sera majoré de 10% du total des boissons comprises dans le cocktail.
Toutes ces classes auront leur accesseurs/mutateurs définis, tout comme leurs constructeurs et les méthodes **toString**, **equals** et **compareTo** définis.
On prendra soin de créer une classe **Main** de test de ces classes.

1. *Écrire les classes* demandées*.*