TP4. Programmation Java

API Collections - Suite

Exercice 1

Soit `l_list` une liste chainée de `String`. Ecrire un programme Java contenant les fonctions permettant de :

- Créer une liste vide
- Ajouter des éléments à la liste : (exemple {Rouge, Vert, Bleu, jaune, Noir})
- Afficher tous les éléments de la liste
- Afficher les éléments de la liste à partir d'une position donnée
- Afficher les éléments de la liste dans le sens inverse
- Rajouter un élément donné à une position spécifique de la liste
- Insérer deux éléments en tête et en queue de liste
- Ajouter des éléments données à des emplacements spécifiques (exemple {Orange, Rose, Blanc} respectivement dans les positions: 1, 3 et 5)
- Afficher tous les éléments avec leurs indices dans la liste
- Vérifier si un élément appartient à la liste
- Supprimer un élément donné de la liste
- Supprimer le premier élément de la liste
- Permuter les éléments « Rouge » et « Noir » de la liste
- Mélanger les éléments de la liste
- Dupliquer la liste dans une nouvelle liste
- Supprimer le premier élément de la liste
- Récupérer (sans supprimer) le premier élément
- Récupérer (sans supprimer) le dernier élément
- Remplacer un élément d'indice i.
- Convertir la liste en ArrayList
- Vérifier si une liste est vide
- Comparer la liste avec une liste donnée

Afficher un menu qui permet à l'utilisateur de choisir le type d'opération à effectuer sur les listes chainées

Exemple

```
==================================
Opérations dur les listes chainées
==================================

   1- Créer une liste
   2- Ajouter des éléments à la liste
   3- Afficher les éléments d'une liste
   4- Etc.
```

Exercice 2

Vous allez ici écrire une classe `MyLinkedList` qui réalise une sur-couche de la classe `LinkedList` offerte par JAVA. Pour se faire vous aller ici utiliser la composition: la classe `MyLinkedList` possède un objet de la classe `LinkedList`. Vous ne devez vous servir que des méthodes de `LinkedList` suivantes:

- void addFirst(Object o)
- void addLast(Object o)
- Object get(int index)
- ainsi que les constructeurs de la classe `LinkedList`

Vous devez implémenter quatre méthodes dont certaines reprennent (sous un autre nom) les fonctionnalités offertes par `LinkedList`. Ces méthodes sont:

- *void ajouteEnTete(Object o)*
- *void ajouteEnQueue(Object o)*
- *Object get(int index)*
- *Object trie()*

Exercice 3

1- On veut représenter des poupées russes dans un programme objet. Ces poupées peuvent s'emboiter les unes dans les autres. On veut distinguer deux types de poupées : celles qui sont creuses et dans lesquelles on peut mettre une autre poupée si elle est plus petite, et celles qui sont pleines et ne peuvent rien contenir.

   - Proposez du code permettant de représenter ces poupées et, pour les poupées creuses, d'ajouter une poupée dedans ou d'en retirer une poupée (on ne peut ajouter une poupée que dans une poupée creuse qui n'en contient pas encore et on ne peut retirer une poupée que d'une poupée creuse qui en contient une autre)

2- On souhaite modéliser à nouveau le fonctionnement des poupées russes mais cette fois ci à l'aide d'une structure de données. Pour cela on définit la classe ListeDePoupees qui hérite de la version générique de la classe LinkedList. Le paramètre générique de la classe ListeDePoupees doit être défini de manière à n'autoriser que des poupées Russes à être ajoutées à la liste.

   L'objectif est de modifier la méthode add() de la classe LinkedList de manière à réaliser les tests nécessaires avant l'ajout.

   - Donner l'entête de la classe ListeDePoupees

   - Donner le code et l'entête de la méthode add() de la classe ListeDePoupees qui permet d'ajouter en fin de liste la poupée passée en paramètre si les conditions suivantes sont respectées : la dernière poupée de la liste est fermée et a une taille inférieure à la poupée passée en paramètre, la poupée passée en paramètre est ouverte et vide.