Grands Moulins

INITIATION À LA PROGRAMMATION (IP2)

TD 6-7 : Listes doublement chaînées

Exercice 1

On considère des listes doublement chaînées d'entiers.

```
public class Liste{
    private Element premier;
    private Element dernier;
}

public class Element{
    private int valeur;
    private Element suivant;
    private Element precedent;

    public Element(int valeur, Element suivant, Element precedent){
        this.valeur = valeur;
        this.suivant = suivant;
        this.precedent = precedent;
    }
}
```

Complétez les deux classes ci-dessus de sorte que la classe `Liste` possède les méthodes suivantes. Sauf indication particulière, vous procéderez selon votre préférence, de manière récursive ou itérative.

1. `int longueur()` qui renvoie la longueur de la liste.
2. `void afficher()` qui affiche la liste dans l'ordre.
3. `void afficherInverse()` qui affiche la liste dans l'ordre inverse.
   (Ici il est intéressant de réfléchir aux deux versions, itératives et récursives)
4. `void ajouterDebut(int valeur)` qui ajoute un élément au début de la liste.
5. `void ajouterFin(int valeur)` qui ajoute un élément à la fin de la liste.
6. `void ajouterAvant(Element e, int valeur)` qui ajoute un nouvel élément à dans la liste juste avant la première occurrence de `e`. Si `e` ne se trouve pas dans la liste, ne rien faire.
7. `void supprimer(Element e)` qui supprime l'élément spécifié.
8. `void supprimer(int valeur)` qui supprime tous les éléments possédant la valeur donnée.
9. `int somme()` qui calcule la somme des éléments de la liste de manière récursive.
10. `boolean estTriee()` qui teste si la liste est triée dans l'ordre croissant de manière récursive.

Exercice 2 (plus avancé)

1. Ajoutez une méthode `void inverser(Element e)` qui inverse la position dans la liste de `e` et de l'élément qui le suit directement. Si `e` est le dernier élément, ne rien faire.
2. Ajoutez une méthode `void passerEnOrdonnant()` qui effectue une lecture de la liste en inversant en cours de route les éléments adjacents dont les valeurs ne sont pas en ordre croissant.

3. En utilisant la méthode de la question précédente, et en la répétant tant que c'est nécessaire, vous pouvez écrire une méthode `void trier()` qui trie la liste par ordre croissant. Pour traduire la phrase "tant que c'est nécessaire" vous pouvez penser à modifier le type retour de la méthode `passerEnOrdonnant`
4. Ajoutez une méthode `boolean verifContrainte()` qui vérifie que pour tout triplet (a, b, c) d'entiers qui se suivent dans la liste on a : $c \neq a + b$.
5. Ecrivez une méthode `boolean estUnPalindrome()` qui teste si la liste est un palindrome. (On parle bien de la liste et pas du nombre obtenu en concaténant tous les nombres figurant dans la liste, ce serait un autre exercice)
6. Ecrivez une méthode `Liste moyenneLocale()` qui renvoie une nouvelle liste où l'élément i est la moyenne (arrondie) des éléments $i-1, i, i+1$.
7. Ecrivez une méthode `void annuleRedondances()` qui remplace chaque morceau de la liste où un même nombre est répété plusieurs fois par une seule occurrence de ce nombre. Par exemple $[2, 2, 2, 5, 5, 1, 2]$ devient $[2, 5, 1, 2]$.