TD 5
Listes doublement chaînées : implémentation

1 Rappel de Cours

Dans ce TD, nous allons réaliser l'implémentation d'**une collection** de type `List`, en utilisant **des listes doublement chaînées**, en cours nous avions vu **une liste simplement chaînée**.
Dans une liste doublement chaînée, à partir d'un **élément** on peut non seulement aller au suivant, mais aussi remonter au précédent. On a donc **deux chaînages** en sens contraire : le chaînage des « suivants », et le chaînage des « précédents ».

2 Exercices

Exercice 1. Modélisation - création d'une classe

On veut implémenter la classe `ListeDouble`, une classe représentant une liste doublement chaînée d'**éléments de type entier**. Les premières lignes du code (hors `import`) sont :

```
class ListeDouble {
    /*Classe interne*/
    private class Element {
        // ...
    }
    /*Attributs de la classe ListeDouble*/
    Element teteDeListe ;
    Element finDeListe ;

    public ListeDouble() { /* ...*/ }

    public boolean estVide() { /* ...*/ }
    public void insereEnTete(int x) { /* ...*/ }
    public void insereEnFin(int x) { /* ...*/ }
    public void insereApres(int v, int x) { /* ...*/ }
    public void insereAvant(int v, int x) { /* ...*/ }
    public void supprime(int v) { /* ...*/ }
    @override
    public String toString() { /* ...*/ }
}
```

Question 1. Quels sont les attributs de la classe interne `Element`, qui représente un « noeud » de la liste doublement chaînée ?
Proposez une implémentation simple de cette classe.

Question 2. Dessinez une liste doublement chaînée de 4 `Element` (vous mettez les valeurs que vous voulez dans les `Element`).

Question 3. Implémentez le **constructeur** de `ListeDouble` ainsi que les méthodes `insereEnTete`, `insereEnFin`, `insereApres`, `insereAvant`, `supprime`, `toString`.

- Dans le cas d'`insereApres` et `insereAvant`, la valeur x doit être insérée après (resp. avant) la première occurrence rencontrée de la valeur v en parcourant la liste de la tête vers la fin. **Si la valeur v n'est pas rencontrée, il n'y a pas d'insertion.**
- Dans le cas de `supprime`, le premier élément rencontré de valeur v doit être supprimé (sens de parcours de la tête vers la fin).

Question 4. Proposez une méthode :

```
public void insertionTriee(int x)
```

qui insère x à la bonne place dans la liste pour que celle-ci soit toujours triée par ordre croissant des valeurs de la tête vers la fin.

Exercice 2. Utilisation - création d'instances et utilisation du modèle

Maintenant que vous avez une classe `ListeDouble` qui modélise/conceptualise la notion de liste d'entiers que l'on peut parcourir vers l'avant ou vers l'arrière, vous pouvez exploiter ce modèle pour créer des listes doublement chaînées de valeurs entières.

Question 1. Proposez une méthode (de test) utilisant ce modèle. Dans cette méthode vous :

1. Définirez deux instances de liste doublement chaînée.
2. Dans la première vous :
   - insérerez les valeurs suivantes : 11, 12, 3,
   - puis vous afficherez cette liste,
   - puis vous insérerez la valeur 22 avant le 12, et la valeur 32 après le 3.
3. Dans la seconde vous insérerez les valeurs : 10, 1, 100, 20, 4 de manière à ce que lorsqu'on affiche cette liste on obtienne l'affichage suivant :

```
1, 4, 10, 20, 100
```

Question 2. Soit le code ci-dessous, indiquez quels affichages seront faits lors de l'exécution de ce code :

```java
ListeDouble maListe = new ListeDouble();
maListe.insereEnTete(25);
maListe.insereEnFin(30);
maListe.insereEnTete(40);
System.out.println("Etat de la liste : " + maListe.toString());

maListe.insereEnFin(60);
maListe.insereApres(25,75);
maListe.insereApres(12,36);
System.out.println("Etat de la liste : " + maListe.toString());

maListe.insereAvant(25,42);
maListe.insereAvant(12,44);
System.out.println("Etat de la liste : " + maListe.toString());

maListe.supprime(60);
maListe.supprime(42);
System.out.println("Etat de la liste : " + maListe.toString());

maListe.insertionTriee(72);
System.out.println("Etat de la liste : " + maListe.toString());


maListe = new ListeDouble();
maListe.insertionTriee(25);
maListe.insertionTriee(10);
maListe.insertionTriee(40);
maListe.insertionTriee(30);
maListe.insertionTriee(32);
System.out.println("Etat de la liste : " + maListe.toString());
```