1 Utilisation de l'interface List

Dans une classe `ListUtils`, en utilisant l'une des implémentations de l'interface Java List, on vous demande de :

1. Créer une méthode `List<Integer> genereRdmIntList` qui génère une liste d'entiers de taille aléatoire (30 maximum tout de même !), initialisée avec des entiers positifs tirés aléatoirement et inférieurs à 100.
2. Écrire une méthode `affiche(List<Integer> l)` pour pouvoir afficher la liste sous la forme :
   $a \rightarrow b \rightarrow ... \rightarrow x$, où a est la valeur du premier entier de la liste, b la seconde, et ainsi de suite. Pour ce faire, vous utiliserez un itérateur. N'oubliez pas le retour à la ligne.
3. Écrire une méthode `afficheInverse(List<Integer> l)` qui affiche la liste en ordre inverse. Vous utiliserez cette fois-ci un itérateur de type `ListIterator` qui permet d'itérer sur une liste dans les deux sens (cf. Javadoc).
4. Écrire une méthode `int somme(List<Integer> l)` qui renvoie la somme des éléments de la liste.
5. Écrire une méthode `int moyenne(List<Integer> l)` qui renvoie la moyenne entière des éléments de la liste.
6. Écrire une méthode `int max(List<Integer> l)` qui retourne la valeur maximale contenue dans la liste.
7. Écrire une méthode `int min(List<Integer> l)` qui renvoie la valeur minimale contenue dans la liste.
8. Écrire une méthode `List<Integer> positions(List<Integer> l, int n)` qui, étant donné un entier n, renvoie la liste des positions (renvoie la liste vide sinon).
   Exemple :$1 \rightarrow 22 \rightarrow 45 \rightarrow 56 \rightarrow 1 \rightarrow 34 \rightarrow 1$
   `positions(1)` doit renvoyer $0 \rightarrow 4 \rightarrow 6$
9. Écrire une méthode `List<Integer> paire(List<Integer> l)` qui renvoie la liste des éléments pairs.
10. Écrire une méthode `boolean estTrie(List<Integer> l)` permettant de vérifier si la liste est triée dans l'ordre croissant.
11. Écrire une méthode `List<Integer> trie(List<Integer> l)` qui renvoie une liste triée (on pourra s'aider de certaines des méthodes précédentes).

En utilisant le framework JUnit et la classe `ListUtilsTest` fournie dans les ressources du TP, assurez-vous que toutes vos méthodes passent les tests associés avant de passer à la suite. Vos classes de test seront placés dans un répertoire de sources `tests`.

2 Implémentation d'une liste simplement chaînée à double extrémité

Dans cette partie, nous allons réaliser l'implémentation d'une liste doublement chaînée. Comme indiqué sur la Figure 1, il s'agit d'une liste chaînée dans les deux sens, qui contient dans l'ordre les éléments 45 et 11.

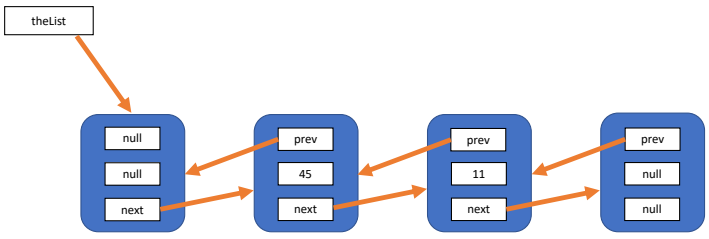

FIGURE 1 – Liste à double extrémité

Nous utiliserons la représentation suivante : une liste contient les références sur le premier noeud et sur le dernier noeud, chaque noeud contient une valeur et une référence vers le noeud suivant.

```
public class MyList<E> {
    E element;
    MyList<E> previous;
    MyList<E> next;
    // méthodes pour traiter une liste
}
```

Pour connaître le comportement des méthodes suivantes à implémenter, vous vous appuierez sur la Javadoc de l'interface `List`.

Question 1 : Créez cette classe `MyList<E>`, ainsi que les constructeurs ad-hoc.

Question 2 : Programmez les méthodes :

```
boolean isEmpty()
int size()
void clear()
String toString() /*Renvoie une chaîne de caractères contenant
  les éléments de la liste séparés par des virgules,
  et encadrés par des crochets.*/
E get(int index)
```

Question 3 : Programmez les méthodes suivantes de recherche dans la liste :

```
int indexOf(Object o)
boolean contains(Object o)
int lastIndexOf(Object o)
```

Question 4 : Programmez les méthodes suivantes d'ajout et de suppression dans la liste :

```
boolean add(E element)
void add(int index, E element)
E remove(int index)//Supprime l'élément situé à l'indice index et le retourne.
boolean remove(Object o)
```

Question 5 : Programmez toutes les méthodes manquantes de façon à ce que votre liste implémente l'interface `List` de Java.

En utilisant le framework JUnit 4 et la classe `MyListTest` fournie dans les ressources du TP, assurez-vous que toutes vos méthodes passent les tests associés.