Algorithmique avancée - TD5

—o000o——o000o—

Implémentation des itérateurs pour listes chaînées

Préambule

Les itérateurs sont des objets qui permettent un parcours simplifié et sécurisé d'une structure de type liste. En Java classique, ils sont accessibles depuis l'interface `Iterator<T>` et notamment la classe `listIterator<T>` qui instancie cette interface. L'idée de cette séance est de dériver notre classe `SList<T>` et de l'enrichir avec cette interface `Iterator<T>`. Pour cela, nous allons créer une nouvelle interface `IList<T>` qui étend `LList` pour avoir l'ensemble des méthodes des listes et également `Iterable<T>` pour avoir accès aux itérateurs.

Principe général Un **itérateur** donne accès à 2 méthodes principales :

1. `boolean hasNext()` qui retourne `true` si et seulement si la structure de liste possède un élément suivant.
2. `T next()` qui retourne la valeur suivante dans la liste et avance d'une position.

Réalisation pour réaliser cet **itérateur**, la solution la plus simple consiste à ajouter le code suivant à votre classe :

```
public Iterator<T> iterator() {
        return new DummyIterator<>(this);
}
public class DummyIterator<T> implements Iterator<T> {
         // a completer
}
```

Problèmes et solutions cette implémentation pose plusieurs problèmes :

- il faut que le premier appel à `hasNext()` vérifie que la liste chaînée n'est pas vide
- il faut que le premier appel à `next()` retourne le 1er élément de la liste en cours de parcours

Pour résoudre ces problèmes :

- un itérateur sera représenté sous la forme d'un nouveau pointeur (en plus des pointeurs `head` et `last`) nommé `current` et ajouté à la classe principale,

- ce pointeur pourra être utilisé pour paramétrer le comportement de `hasNext()` et `next()`
- comme un nouvel itérateur sera créé à chaque parcours, il faut que le constructeur de `DummyIterator` ré-initialise le pointeur courant.

1 Implémentation des méthodes obligatoires

Question 1 Créez une interface `IList<T>` et la classe `ISList<T>` qui dérive de `SList<T>` et implémente l'interface `IList<T>`

Question 2 Ajoutez le champ `Node<T> current` à votre classe `ISList<T>`.

Question 3 Complétez les méthodes `boolean hasNext()` et `T next()` sur la base des explications précédentes dans la classe interne `DummyIterator`.

2 Ré-écriture des méthodes de la classe `SList`

Question 4 Réécrivez les méthodes suivantes en les précédant de la mention `@Override` :

- `boolean contains(T elem)`
- `int indexOf(T elem)`
- `T get(int index)`
- `void set(T elem, int index)`