TD 6
Plus de listes chaînées

1 Rappel de Cours

Dans ce TD, nous allons continuer dans l'implémentation de **collection** de type `List`, en s'intéressant à la notion de **liste triée**, et à celle de **liste circulaire**. Pour rappel, en cours nous avions vu la notion de **liste simplement chaînée**, et au TD précédent la notion de **liste doublement chaînée**.

2 Exercices

Exercice 1. Listes triées - CC 2016/2017

Le but de cet exercice est de créer une classe `ListeTriee` contenant une liste de nombres entiers triés par ordre croissant. La spécification de cette classe est la suivante :

```
1  public class ListeTriee {
2      // Attributs
3      ...
4      // Construit une liste vide
5      public ListeTriee() { ... }
6
7      // Insère un élément x à la bonne place dans la liste. On autorise les
       ↪  duplications.
8      public void insere(int x) { ... }
9
10     // Supprime UNE occurrence de l'entier x dans la liste. Renvoie true si
       ↪  une suppression a été effectuée, false sinon.
11     public boolean supprime(int x) { ... }
12
13     //Renvoie une chaîne représentant la liste.
14     public String toString(){ ... }
15 }
```

A titre d'exemple, le petit programme ci-dessous produira l'affichage : `5, 5, 7, 15, 22`

```
1 ListeTriee lt = new ListeTriee() ;
2 lt.insere(15) ; lt.insere(5) ; lt.insere(7) ; lt.insere(5) ; lt.insere(22) ;
3 System.out.println(lt);
```

Ajouter les lignes :

```
1 lt.supprime(5) ; lt.supprime(15) ;
2 System.out.println(lt);
```

Produira l'affichage : `5, 7, 22`

Question 1. Pour implémenter `ListeTriee`, nous allons nous baser sur une liste chaînée de type `File` (vue en cours).
- Proposez une classe privée `Element` pour contenir les éléments de la file.
- Proposez des attributs pour `ListeTriee`.

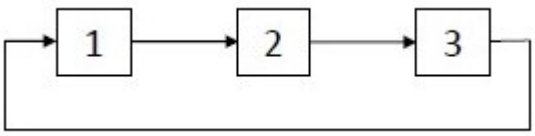
Question 2. Implémentez la méthode `insere` de `ListeTriee`.
Astuce : vous pouvez dessiner au fur et à mesure l'évolution de la liste `lt` dans l'exemple ci-dessus.

Question 3. Implémentez la méthode `supprime` de `ListeTriee`.

Question 4. Implémentez la méthode `toString` de `ListeTriee`.

Exercice 2. Liste circulaire - Exam 2015/2016 session 1

Une liste chaînée circulaire est une liste chaînée qui n'a pas de « dernier élément » pointant sur `null` : à la place, son « dernier élément » pointe sur le premier, faisant de la liste un cycle (*cf.* figure ci-dessous).



On suppose des listes chaînées fabriquées à partir de la classe `Element` suivante (pour simplifier nous ne respectons pas l'encapsulation) :

```java
class Element {
    // attributs
    public int valeur;
    public Element suivant;

    // constructeur
    public Element(int v, Element s) {
        this.valeur = v;
        this.suivant = s;
    }
}
```

Question 1. Écrivez une classe `ListeCirculaire`, respectant le principe d'encapsulation, qui implémente une liste circulaire.
- Vous pouvez (et devez) utiliser la classe `Element` qui vous est fournie.
- On vous demande les 4 méthodes suivantes :
  1. Un constructeur qui initialise une liste vide (aucun élément) : `public ListeCirculaire()`
  2. Une méthode qui vérifie si la liste est circulaire ou pas : `public boolean estCirculaire()`
  3. Une méthode permettant d'insérer un élément dans la liste : `public void ajouteElement(int x)`
  4. Une méthode comptant le nombre d'éléments de la liste avec un parcours de liste (interdiction d'avoir un attribut de classe comptant les éléments) : `public int nbElements()`