PRG1 – TP5 – Listes

La classe *SmallSet* définie dans le TP4 permet de déclarer des ensembles définis sur le domaine 0... 255. Le TP consiste à pouvoir manipuler des ensembles d'entiers définis sur l'intervalle 0... 32767 (i.e. 2^{15} - 1). On décide de représenter chaque ensemble par une liste dont les éléments sont des objets *SubSet* avec :

```
1  public class MySet extends List<SubSet> {
2      private static final int MAX_RANG = 128;
3      private static final SubSet FLAG_VALUE =
4                          new SubSet(MAX_RANG, new SmallSet());
5      public MySet() {
6          super();
7          setFlag(FLAG_VALUE);
8      }
9      ...
10 }
```

```
1  public class SubSet {
2      public int rank;
3      public SmallSet set;
4    ...
5  }
```

Un entier x ($0 \leq x \leq 32767$) appartient à un ensemble si et seulement si la liste qui représente cet ensemble contient un élément tel que :

- le champ *rank* vaut x/256,
- x%256 appartient (au sens de la classe *SmallSet*) au champ *set*.

Dans la liste, on ne fait figurer que des éléments dont le champ *set* est **non vide**. En constatant que $0 \leq rank \leq 127$, on choisit de trier les éléments par rangs croissants.

1. Manipulation d'ensembles à l'aide des accès listes

Le programme principal initialise à vide tous les ensembles E_i puis itère sur un menu proposant des commandes activables par des boutons. Selon la commande, on lit un ou deux numéros d'ensembles notés n1 et n2 compris entre 0 et `MAX_SET-1` ; les commandes proposées sont :


```
1  /**
2   * @return true si le nombre saisi par l'utilisateur appartient à this,
3   *         false sinon
4   */
5  public boolean contains()
6
7  /**
8   * Ajouter à this toutes les valeurs saisies par l'utilisateur et
9   * afficher le nouveau contenu (arrêt par lecture de -1).
10  */
11 public void add()
12
13 /**
14  * Supprimer de this toutes les valeurs saisies par l'utilisateur et
15  * afficher le nouveau contenu (arrêt par lecture de -1).
16  */
17 public void remove()
18
19 /**
20  * @return taille de l'ensemble this
21  */
22 public int size()
23
24 /**
25  * this devient la différence de this et set2.
26  *
27  * @param set2
28  *            deuxième ensemble
29  */
30 public void difference(MySet set2)
31
32 /**
33  * this devient la différence symétrique de this et set2.
34  *
35  * @param set2
36  *            deuxième ensemble
37  */
38 public void symmetricDifference(MySet set2)
39
40 /**
41  * this devient l'intersection de this et set2.
42  *
43  * @param set2
44  *            deuxième ensemble
```

```
45   */
46  public void intersection(MySet set2)
47
48  /**
49   * this devient l'union de this et set2.
50   *
51   * @param set2
52   *             deuxième ensemble
53   */
54  public void union(MySet set2)
55
56  /**
57   * @param o
58   *             deuxième ensemble
59   *
60   * @return true si les ensembles this et o sont égaux, false sinon
61   */
62  public boolean equals(Object o)
63
64  /**
65   * @param set2
66   *             deuxième ensemble
67   * @return true si this est inclus dans set2, false sinon
68   */
69  public boolean isIncludedIn(MySet set2)
70
71  /**
72   * Créer this à partir d'un fichier choisi par l'utilisateur
73   * contenant une séquence d'entiers positifs terminée par -1
74   * (cf f0.ens, f1.ens, f2.ens, f3.ens et f4.ens).
75   */
76  public void restore()
77
78  /**
79   * Sauvegarder this dans un fichier d'entiers positifs terminé par -1.
80  */
81  public void save()
82
83  /**
84   * Afficher à l'écran les entiers appartenant à this,
85   * dix entiers par ligne d'écran.
86   */
87  public void print()
```


La classe *TpList* gère le menu et traite les commandes l'aide des **méthodes d'instance** de la classe *MySet*.

Certaines méthodes de la classe *MySet* sont fournies. Compléter la classe *MySet* pour définir les méthodes d'instance permettant de réaliser certaines autres commandes (voir liste ci-dessous).

Les classes offertes et les jeux d'essais sont disponibles sous `G:\l3miage\prg1\tp5\algo` (Windows) ou `/share/l3miage/prg1/tp5/algo` (Linux) : *list-util.jar, list.tar*.

Ces fichiers seront disponibles à partir du mardi 20 octobre 2015.

2. Écriture d'une mise en œuvre

Une fois la classe MySet.java terminée, compléter la classe List.java du répertoire `/share/l3miage/prg1/tp5/meo` (`G:\l3miage\prg1\tp5\meo`) pour une mise en œuvre des listes d'objets `T` en double chaînage par références.

Rendre lors de la dernière séance de la semaine du 12/10/2015

Les **méthodes d'instance** de la classe MySet, correspondant aux commandes : containment, add et intersection.

Rendre pour le vendredi 13/11/2015 au plus tard

Exclusivement :

- la classe *MySet.java* traitant les méthodes
  - add,
  - containment,
  - difference,
  - symmetricDifference,
  - equals,
  - isIncludedin,
  - intersection,
  - remove.

  Les autres méthodes ne sont pas demandées, mais elles peuvent apparaître au contrôle de TP.
- la classe *List.java*.

2015/2016

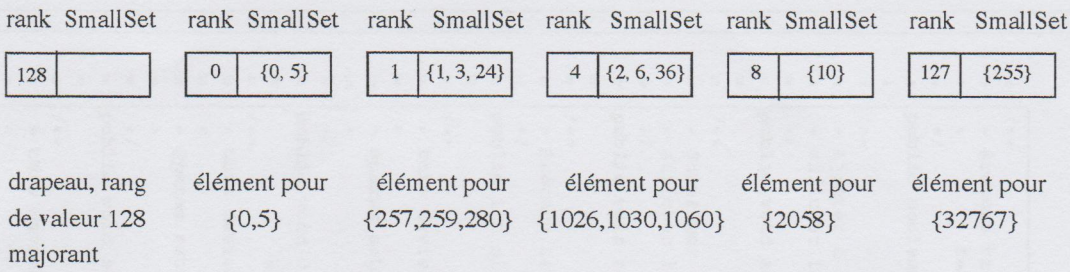
TP N° 5 Listes - Structures de données utilisées

Exemple montrant la représentation d'un ensemble par une liste

- **Ensemble au niveau utilisateur (domaine 0 .. 32767)**

  $\{0, 5, 257, 259, 280, 1026, 1030, 1060, 2058, 32767\}$

- **Liste d'éléments associée à cet ensemble**



où chaque élément de la liste est de type SubSet (voir énoncé du TP N° 5). Les éléments sont triés par rangs croissants, et comme $0 \leq \text{rang} \leq 127$, on choisit de placer le majorant 128 dans le rang du drapeau.

Remarque :

Ce problème ressemble beaucoup au problème étudié au Cours/TD N° 7 sur la représentation des matrices creuses par des listes de doublets. Ainsi, vous devrez adapter les résultats, en particulier pour l'écriture des commandes Intersection, Union, SymmetricDifference et IsIncludedIn.