Programmation – Listes

La classe $SmallSet$ définie plus bas 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.

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 set2
58   *            deuxième ensemble
59   *
60   * @return true si les ensembles this et set2 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 $MySet$.

Écrire toutes les méthodes manquantes de la classe MySet.

Classe SmallSet

La classe SmallSet permet de déclarer et de manipuler des ensembles définis sur le domaine $0 \ldots 255$. On choisit de représenter chaque ensemble E par un tableau **boolean** [256] tab avec la convention $i \in E \Leftrightarrow E.tab[i]$ vaut vrai.

```
1  public class SmallSet {
2
3    private boolean[] tab = new boolean[256];
4
5    public SmallSet () {
6        for (int i = 0; i <= 255; ++i) {
7            tab[i] = false;
8        }
9    }
10
11   public SmallSet (boolean[] t) {
12       for (int i = 0; i <= 255; ++i) {
13           tab[i] = t[i];
14       }
15   }
16
17   /**
18    * @return      nombre de valeurs appartenant à l'ensemble
19    */
20   public int size () { ... }
21
22   /**
23    * @param x     valeur à tester
24    * @pre         0 <= x <= 255
25    * @return      true, si l'entier x appartient à l'ensemble,
26    *              false sinon
27    */
28   public boolean contains (int x) { ... }
29
30   /**
31    * @return      true, si l'ensemble est vide, false sinon
```

```
32      */
33    public boolean isEmpty () { ... }
34
35    /**
36     * Ajoute x à l'ensemble (sans effet si x déjà présent)
37     *
38     * @param  x    valeur à ajouter
39     * @pre        0 <= x <= 255
40     */
41    public void add (int x) { ... }
42
43    /**
44     * Retire x de l'ensemble (sans effet si x n'est pas présent)
45     *
46     * @param  x    valeur à supprimer
47     * @pre        0 <= x <= 255
48     */
49    public void remove (int x) { ... }
50
51    /**
52     * Ajoute à l'ensemble les valeurs deb, deb+1, deb+2, ..., fin.
53     *
54     * @param begin   début de l'intervalle
55     * @param end     fin de l'intervalle
56     * @pre           0 <= begin <= end <= 255
57     */
58    public void addInterval (int deb, int fin) { ... }
59
60    /**
61     * Retire de l'ensemble les valeurs deb, deb+1, deb+2, ..., fin.
62     *
63     * @param begin   début de l'intervalle
64     * @param end     fin de l'intervalle
65     * @pre           0 <= begin <= end <= 255
66     */
67    public void removeInterval (int deb, int fin) { ... }
68
69    /**
70     * Réalise l'opération this ← this ∪ f.
71     *
72     * @param f       second ensemble
73     */
74    public void union (SmallSet f) { ... }
75
76    /**
```

```
77      * Réalise l'opération this ← this ∩ f.
78      *
79      * @param f       second ensemble
80      */
81     public void intersection (SmallSet f) { ... }
82
83     /**
84      * Réalise l'opération this ← this \ f.
85      *
86      * @param f       second ensemble
87      */
88     public void difference (SmallSet f) { ... }
89
90     /**
91      * Réalise l'opération this ← this △ f.
92      *
93      * @param f       second ensemble
94      */
95     public void symmetricDifference (SmallSet f) { ... }
96
97     /**
98      * Réalise l'opération this ← this.
99      */
100    public void complement () { ... }
101
102    /**
103     * Réalise l'opération this ← ∅.
104     */
105    public void clear () { ... }
106
107    /**
108     * @param f     second ensemble
109     * @return      true, si this ⊆ f, false sinon
110     */
111    public boolean isIncludedIn (SmallSet f) { ... }
112
113    /**
114     * @return      copie de this
115     */
116    public SmallSet copy () { ... }
117        return new SmallSet(tab);
118    }
119
120    /**
121     * @return     true, si this et f sont égaux, false sinon
```

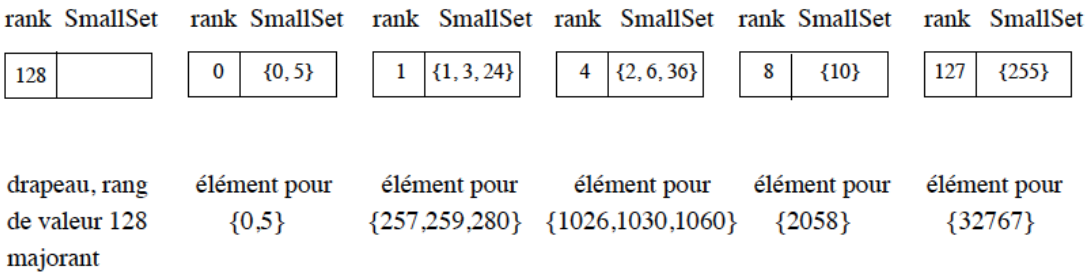
```
122     */
123    @Override
124    public boolean equals (Object o) { ... }
125
126    @Override
127    public String toString() {
128        String s = "éléments présents : ";
129        for (int i = 0; i <= 255; ++i) {
130            if (tab[i]) {
131                s = s + i + " ";
132            }
133        }
134        return s;
135    }
136 }
```

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 $0 \leq$ rang ≤ 127, on choisit de placer le majorant 128 dans le rang du drapeau.

Remarque :

Ce problème ressemble beaucoup au problème étudié au Cours/T 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.