PRG – TP3 à TP6 – Listes

Il est fortement conseillé de lire la totalité du sujet avant de se lancer dans la réalisation.

1 Introduction

On souhaite créer une classe en Java appelée `MySet` permettant la représentation et la gestion des ensembles dont les éléments se trouvent dans l'intervalle des entiers de 0 à 32 767. Pour une gestion et un stockage efficaces, on décide que la définition de la classe `MySet` soit faite à l'aide d'une liste chaînée dont les éléments sont des objets de type `Subset`. Chaque `Subset` correspond à un sous-ensemble d'entiers de taille 256. Le squelette de la classe `Subset` est le suivant :

```
1  public class SubSet {
2      public int rank;  // compris entre 0 et 127
3      public SmallSet set;  // ensemble d'entiers entre 0 et 255
4    ...
5  }
```

Un entier x ($0 \leq x \leq 32\,767$) est représenté par son `rank` $r = x/256$ et par son modulo $m = x\%256$. Ainsi x est représenté dans le `SubSet` de rang *r* avec *m* appartenant à l'ensemble `set` associé, où `set` est une instance de la classe `SmallSet` permettant de manipuler des ensembles d'entiers variant de 0 à 255. Ce `SubSet` est alors un élément de la liste `MySet` qui représente l'ensemble des entiers d'un utilisateur. Le squelette de la classe `MySet` est le suivant :

```
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 }
```

Par conséquent, un entier x ($0 \leq x \leq 32\,767$) appartient à l'ensemble defini par `MySet` si et seulement si la liste `MySet` contient un élément `SubSet` tel que :

— la valeur du $rank$ vaut $x/256$,

— l'élément $x\%256$ appartient au champ set de cet élément.

Propriétés à respecter :

— les éléments de la liste `MySet` sont triés par ordre croissant selon le rang,

— dans la liste `MySet`, on ne fait figurer que des éléments dont le champ *set* est **non vide**,

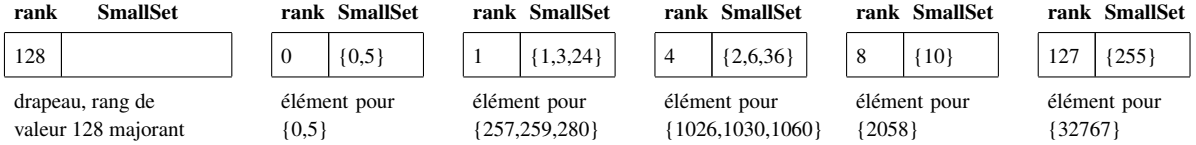
— le majorant 128 est associé au rang de l'élément drapeau de la liste.

1.1 Exemple d'un ensemble représenté avec une liste `MySet`

– On souhaite représenter l'ensemble suivant avec une liste `MySet` :

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

– Voici la liste des éléments associée à cet ensemble :



Rappelons que chaque élément de la liste est de type `SubSet`. Les éléments sont triés par rangs croissants, et 0 ≤ rang ≤ 127. On place le majorant 128 comme le rang du drapeau.

1.2 La classe SmallSet

La classe `SmallSet` permet de déclarer et de manipuler de petits ensembles d'entiers dont les éléments appartiennent à l'intervalle de 0 à 255.

Notez bien que l'implémentation de la classe `SmallSet` vous est fournie. Par conséquent, vous devez l'importer puis l'inclure dans votre projet.

Méthodes de la classe SmallSet

int size()	nombre de valeurs appartenant à l'ensemble
boolean contains (int x)	vérifie si l'entier x appartient à l'ensemble
void boolean isEmpty ()	vérifie si l'ensemble est vide
void add (int x)	Ajoute x à l'ensemble (sans effet si x déjà présent)
void remove (int x)	Retire x de l'ensemble (sans effet si x est absent)
void addInterval (int deb, int fin)	Ajoute à l'ensemble les valeurs deb, deb+1, . . ., fin.
void removeInterval (int deb, int fin)	Retire de l'ensemble les valeurs deb, deb+1, . . ., fin.
void union (SmallSet set2)	Réalise l'opération **this** $\leftarrow$ **this** $\cup$ set2.
void intersection (SmallSet set2)	Réalise l'opération **this** $\leftarrow$ **this** $\cap$ set2.
void difference (SmallSet set2)	Réalise l'opération **this** $\leftarrow$ **this** $\setminus$ set2.
void symmetricDifference (SmallSet set2)	Réalise l'opération **this** $\leftarrow$ **this** $\triangle$ set2.
void complement ()	Réalise l'opération **this** $\leftarrow$ $\overline{\textbf{this}}$.
void clear ()	Réalise l'opération **this** $\leftarrow$ $\emptyset$.
boolean isIncludedIn (SmallSet set2)	return true, si **this** $\subseteq$ set2, false sinon
SmallSet clone ()	return copie de **this**
boolean equals (Object set2)	return true, si **this** et set2 sont égaux, false sinon
String toString()	

2 Manipulation d'ensembles avec la classe `MySet`

Ci-dessous, on vous donne le squelette commenté de la classe `MySet`. Le code source associé à cette classe se trouve dans le fichier *list.tar* disponible sur Moodle. Les classes compilées de `List<T>` et `SmallSet` sont disponibles sous Moodle dans *list-util.jar* et vous permettent de tester votre `MySet` **avant** d'implémenter `List<T>`, puis `SmallSet`.
Inspectez attentivement les méthodes d'instance à fournir puis compléter le squelette de la classe `MySet` par les méthodes suivantes.

```
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
```

3 Test de `MySet`

Pour tester le bon fonctionnement des méthodes d'instance implémentées de la classe `MySet`, les deux méthodes suivantes doivent être suivies :

— *test en mode interactif* : Vous disposez d'un widget JAVA muni d'une interface graphique qui permet d'appeler les diverses méthodes d'instance de la classe `MySet`. Au démarrage de l'application, le programme principal crée `MAX_SET-1` ensembles vides. Puis, selon l'opération à réaliser, on lit un ou deux numéros d'ensembles notés n1 et n2 compris entre 0 et `MAX_SET-1`. La classe $TpList$ met en liaison les opérations enclenchées par l'usager via l'interface et les méthodes d'instance de la classe $MySet$.

— *test en mode bash* : Vous disposez d'un jeux d'essais disponible dans la source du code fournie exécutable avec JUnit.

4 Écriture d'une mise en œuvre d'une liste (en PRG)

La classe MySet.java étant terminée, on vous demande maintenant de compléter la classe `List.java` (définie en PRG) qui représente une mise en œuvre des listes d'objets `T` en double chaînage par références. Vous devez ensuite utiliser *votre* propre meo dans le code `MySet.java`. Vérifier le bon fonctionnement de votre `MySet.java` avec votre mise en œuvre.

5 Écriture de la classe `SmallSet` (bonus)

Rappel : La classe `SmallSet` permet de déclarer et de manipuler de petits ensembles d'entiers dont les éléments appartiennent à l'intervalle de 0 à 255.
Chaque petit ensemble E est représenté par un tableau de booléens de taille 256, attribut tab, tel que E.tab[i] == True **si et seulement si** $i \in E$. Dans cette partie, vous devez implémenter votre propre classe `SmallSet`. Pour cela, on vous fournit le squelette du code dans le fichier `SmallSet.java` se trouvant sur Moodle. Une fois la classe `SmallSet` complétée, n'oubliez pas de tester le bon fonctionnement de votre code. Pensez à utiliser au maximum les opérateurs logiques.

6 Travail à rendre

Les fichiers complétés `MySet.java`, `List.java` et (éventuellement) `SmallSet.java` sont **à rendre sur Moodle au plus tard le samedi 7 octobre 2023 à 20h**.

Vous veillerez à ne pas modifier les noms de classes et les packages des squelettes de code fournis.

Le code source livré devra être entièrement compilable et l'en-tête de vos fichiers sera un commentaire (idéalement de type JavaDoc) indiquant les noms et prénoms des auteurs (les deux membres du binôme).

Attention, si vous programmez sur votre ordinateur personnel, vos classes doivent pouvoir être utilisées dans un environnement du type des salles de TP Istic.

Travail préliminaire à rendre à votre enseignant.e lors de la 1ere séance de TP

Soit les deux listes suivantes :

n1 : {10, 21, 79, 121, 350, 826, 930, 931, 4259, 30201}

n2 : {10, 21, 121, 360, 420, 777, 806, 2001, 3058, 32767}

Donnez la représentation des listes n1 et n2 tel qu'elles sont définies par MySet.

Donnez la représentation des listes résultantes des opérations suivantes :

(a) n1 $\cap$ n2

(b) n1 $\triangle$ n2

(c) n1 $\cup$ n2