

## Contrôle continu – Première session

– Documents autorisés –

– Dispositifs électroniques (téléphones, calculatrices, etc.) interdits –

Répondez aux questions sur cette feuille. Si vous n'avez pas assez de place pour finir votre réponse, utilisez une feuille additionnelle.

Ce contrôle a 4 questions, pour un total de 20 points.

Question:	1	2	3	4	Total
Points:	7	6	7	0	20
Score:					

```
1 import java.util.List;
2 import java.util.ArrayList;
3
4 public class SortedIntegerSet {
5     protected List<Integer> list;
6
7     public SortedIntegerSet() { list = new ArrayList<Integer>();}
8     public int get(int index) {return list.get(index);}
9     public int size() {return list.size();}
10    public void add(int value) {list.add(value);}
11
12    public boolean containsAll(SortedIntegerSet other) {
13        if (other.size() > this.size())
14            return false;
15        int i = 0;
16        int j = 0;
17        while (i < other.size()) {
18            if (this.get(j) < other.get(i)) {
19                j++;
20                if (j ≥ this.size())
21                    return true;
22            } else if (other.get(i) == this.get(j))
23                i++;
24            else
25                return false;
26        }
27        return true;
28    }
29 }
```

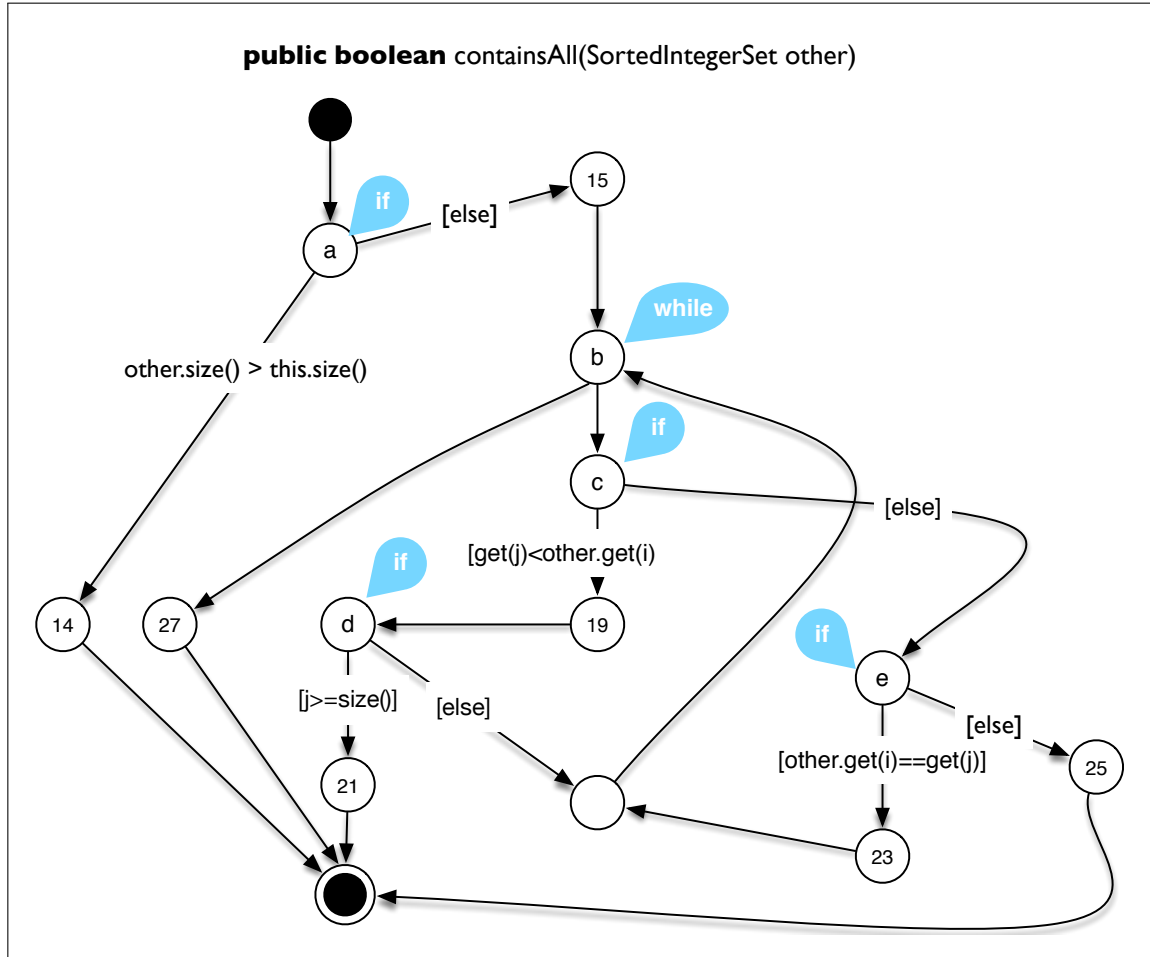
FIGURE 1 – Code de la classe SortedIntegerSet

### 1. Test structurel

La classe SortedIntegerSet (voir Figure 1) encapsule une liste d'entiers triés et un ensemble d'opérations pour manipuler cette liste. La méthode containsAll rend **true** si la liste courante contient tous les éléments de la liste **other** passée en paramètre.

- (2) (a) Donnez le graphe de flot de contrôle de containsAll. Vous pouvez utiliser les numéros de ligne pour nommer les sommets.

**Solution:**



- (2) (b) Combien de 1-chemins y a-t-il dans ce graphe ?

**Solution:**

1. a, 14
2. a, 15, b, 27
3. a, 15, b, c, 19, d, 21
4. a, 15, b, c, 19, d, b, 27
5. a, 15, b, c, e, 23, b, 27
6. a, 15, b, c, e, 25

- (2) (c) Donnez des données de test pour couvrir tous les 1-chemins de containsAll() et précisez le résultat attendu pour chacune de ces données. Vous n'avez pas à écrire le code JUnit correspondant, et vous pouvez simplement présenter vos tests sous la forme d'un tableau de la manière suivante :

$d_x$	$d_y$	...	Résultat attendu
$a_1$	$b_1$	...	$r_1$
$a_2$	$b_2$	...	$r_2$
...	...	...	...

avec  $d_x$ ,  $d_y$  les paramètres de la méthode,  $a_i$ ,  $b_j$  les données d'entrées de chaque test et  $r_1$ ,  $r_2$  les résultats attendus. S'il n'y a pas de résultat attendu valide, mettez un point d'interrogation dans la dernière colonne.

**Solution:**

other	this.list	...	Résultat attendu
{1}	$\emptyset$	...	<i>false</i>
$\emptyset$	{1}	...	<i>true</i>
{2}	{1}	...	<i>false</i> (erreur)
{2}	{1,2}	...	<i>true</i>
{1}	{1}	...	<i>true</i>
{1}	{2}	...	<i>false</i>

- (1) (d) Y a-t-il un bug dans cette méthode ? Si oui, lequel et à quelle ligne ?

**Solution:** Oui, ligne 21: `return true`; à la place de `return false` ;.

Remplacer	par
>	>=
true	false
this	other

TABLE 1 – Opérateurs de mutation

## 2. Analyse de la mutation

Considérez les opérateurs de mutation suivants:

- (2) (a) La Table 1 contient trois opérateurs de mutation. Appliqués sur containsAll, combien de mutants seront produits?

**Solution:** 7 mutants, 1 pour >, 2 pour true et 4 pour this.

- (2) (b) En considérant les opérateurs de mutation de la Table 1, calculez le score de mutation des données de test que vous avez produites lors de la question 1; détaillez la méthode que vous aurez suivie et vos résultats.

**Solution:**

Données	other	this.list	Rés. att.	M1	M2	M3	M4	M5	M6	M7	MS
DT 1	{1}	∅	false	–	–	–	K	–	–	–	1/7
DT 2	∅	{1}	true	–	–	K	–	–	–	–	1/7
DT 3	{2}	{1}	false	–	–	K	K	–	K	K	4/7
DT 4	{2}	{1,2}	true	–	–	K	–	K	–	K	3/7
DT 5	{1}	{1}	true	K	–	K	–	–	–	–	2/7
DT 6	{1}	{2}	false	–	–	–	–	–	–	K	1/7

- (2) (c) Qu'observez vous parmi les mutants ?

**Solution:**

### 3. Test fonctionnel

Considérez la méthode suivante:

```

1  /**
2  * This operation defines the decision procedure of the home automation system,
3  * ie when should shutters be opened or closed depending on the sensors.
4  *
5  * This operation is stateless/functional: it only takes inputs and returns
6  * a value, it does NOT change or use the state of the object.
7  *
8  * The requirements concerning the shutters are the following:
9  * - If luminosity is less than 400, we close the shutters for the night.
10 * - If luminosity is between 400 and 100 000, we open the shutters.
11 * - If luminosity is more than 100 000, we close to avoid the heat.
12 * - If the wind direction is WEST, and the wind speed is more than 40, we open
13 * the shutters so that they don't break (because the windows mostly face west).
14 * - In all other cases, we do nothing.
15 *
16 * @param windSpeed the wind speed to consider
17 * @param windDirection the wind direction to consider
18 * @param luminosity the luminosity to consider
19 * @return Open is shutters should be opened, Close if they should be closed,
20 *         Nothing if nothing must be done.
21 */
22 public PlannedAction analyze(int windSpeed, Direction windDirection, int luminosity);

```

- (1) (a) Identifiez les variables sur lesquelles jouer en entrée (les données de test), ainsi que les valeurs observables en résultat (Oracle).

**Solution:**

**Données de test:** windSpeed, windDirection, luminosity.

**Résultat observable:** instances de PlannedAction.

- (1) (b) En considérant pour cette question que Direction n'a que deux valeurs possibles (East et West), réalisez pour chacune de ces valeurs une analyse partitionnelle, afin d'en déduire des classes d'équivalences.

**Solution:**

**windSpeed** : [0-40], [41-∞].

**windDirection** : East, West.

**luminosity** : [0-399], [400-100000], [100001-∞].

**result** : Open, Close, Nothing.

- (2) (c) Etablissez une table de décision décrivant le comportement de la méthode.

**Solution:**

<b>windSpeed</b>	[0-40]		X		X		X	
	[41-∞]	X		X		X		X
<b>windDirection</b>	East		*		*		*	X
	West	X	*	X	*	X	*	
<b>luminosity</b>	[0-399]	*	X	X				
	[400-100000]	*			X	X		
	[100000-∞]	*					X	X
<b>Result</b>	Open	X			X	X		
	Close		X	X			X	X
	Nothing							

- (2) (d) A partir de cette table, déduisez un ensemble fini minimum de cas de test pertinents (données de test et oracle) pour la méthode.

**Solution:**

```
analyze(50, West, 1000) = Open
analyze(30, East, 200) = Close
analyze(50, West, 200) = Close
analyze(20, West, 1000) = Open
analyze(60, West, 500) = Open
analyze(20, West, 100001) = Close
analyze(60, East, 100001) = Close
```

- (1) (e) Voyez vous un problème dans la spécification de l'opération `analyze` ? Si oui, lequel ?

**Solution:** Oui, la méthode ne retourne jamais la valeur "Nothing".

- (2<sup>1</sup>/<sub>2</sub> (bonus)) 4. Pour chacune de ces déclarations, indiquez soit **vrai**, soit **faux**.
- (a) Le test statique peut produire des verdicts erronés.  
**A. Vrai** B. Faux
  - (b) Le test dynamique ne manque aucune erreur.  
A. Vrai **B. Faux**
  - (c) Une suite de test qui couvre tous les chemins trouve toutes les erreurs.  
A. Vrai **B. Faux**
  - (d) Le test par mutation permet de simuler la couverture des instructions.  
**A. Vrai** B. Faux
  - (e) Il existe des algorithmes qui garantissent l'ordre optimal pour le test d'intégration.  
A. Vrai **B. Faux**