BPOO - Sujet TD 6

Dut/Info-S2/M2103

Table des matières

PreReq	Tableaux dynamiques, structures chaînées.
ObjTD	**Mettre en oeuvre en java une classe Ensemble**.
Durée	**1 séances** de 1,5h

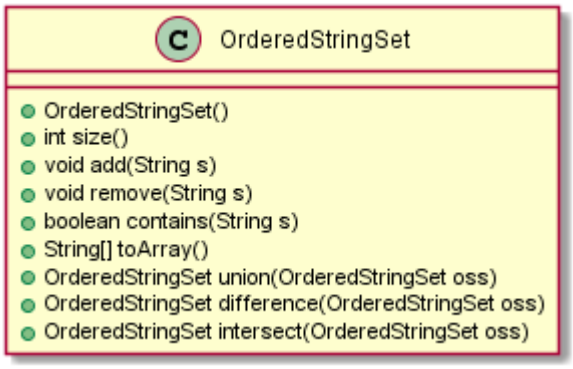
1. Domaine d'étude : implémenter un ensemble ordonné de String

On s'intéresse dans ce TD à la mise en oeuvre d'une classe **Ensemble ordonné** de chaînes de caractères.

Un ensemble ordonné est une **collection ordonnée qui n'accepte pas les doublons**.

- Par exemple, un ensemble de String n'accepte qu'une seule fois la valeur "bonjour". Par contre "bonjour" et "BONjour" sont acceptées.
- L'ensemble constitué des trois valeurs "a", "d", "b", "c" sera ordonné en { "a", "b", "c", "d" }.
- On supposera dans tout l'exercice que la valeur null est refusée (non ajoutée, non enlevé, non recherchée).

Le diagramme UML suivant indique l'interface générale de la classe attendue.



2. Réflexion sur la mise en oeuvre

Dans un premier temps, on imagine deux implémentations :

- par liste chaînée simplement chaînée comme déjà vu,
- par tableau dynamique comme déjà vu : le tableau a, à tout moment, une taille de celle du nombre d'éléments contenus dans l'ensemble.

Une implémentation par ArrayList<E> serait plus simple à mettre en oeuvre mais relève de la même approche : un tableau "dynamique".

Question : Indiquer rapidement les méthodes pour mettre en oeuvre les opérations (+avantages/inconvénients) dans chaque approche :

- pour l'insertion (add) :
  - Tableau dynamique :

    .

    .

    .
  - Liste chaînée :

    .

    .

    .
- pour la suppression (remove) :
  - Tableau dynamique :

    .

.

.

- Liste chaînée :

.

.

.

- pour la recherche (contains) :
  - Tableau dynamique :

.

.

.

  - Liste chaînée :

.

.

.

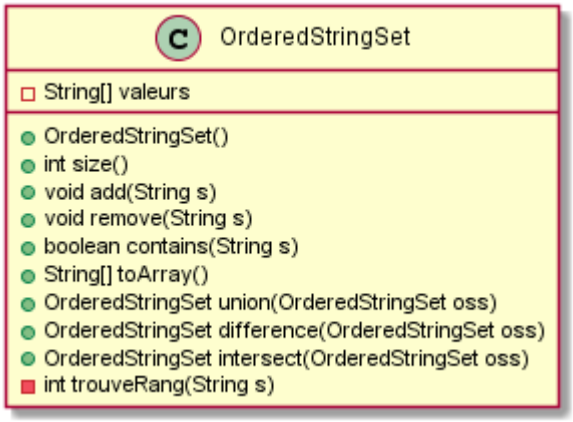
---

3. Mise en oeuvre par tableau dynamique

3.1. Principes

On retient la mise ne oeuvre par tableau dynamique.

Le diagramme UML suivant indique la conception réalisée pour cette implémentation.



Quelques exemples :

1. `OrderedStringSet oss = new OrderedStringSet()` ⇒ valeurs :

```
||  // Tableau de longueur 0
++
||
```

2. `oss.add("S3");` ⇒ valeurs :

```
|  0 |
+----+
|"S3"|  // Attention => en vrai des références
```

3. `oss.add("S1");` ⇒ valeurs :

```
| 0  | 1  |
+----+----+
|"S1"|"S3"|  // Attention => en vrai des références
```

4. `oss.add("S2");` ⇒ valeurs :

```
| 0  | 1  | 2  |
+----+----+----+
|"S1"|"S2"|"S3"|  // Attention => en vrai des références
```

5. `oss.add("S1");` ⇒ valeurs :

```
| 0  | 1  | 2  |
+----+----+----+
|"S1"|"S2"|"S3"|  // Attention => en vrai des références
```

6. `oss.add(null);` ⇒ valeurs :

```
| 0  | 1  | 2  |
+----+----+----+
|"S1"|"S2"|"S3"|  // Attention => en vrai des références
```

7. `oss.contains("S2")` ⇒ true
8. `oss.contains("S1")` ⇒ true
9. `oss.contains("toto")` ⇒ false
10. `oss.contains("s1")` ⇒ false
11. `oss.contains(null)` ⇒ false
12. `oss.remove(null);` ⇒ valeurs :

```
| 0  | 1  | 2  |
+----+----+----+
|"S1"|"S2"|"S3"|  // Attention => en vrai des références
```

13. `oss.remove("s2");` ⇒ valeurs :

```
| 0  | 1  | 2  |
+----+----+----+
|"S1"|"S2"|"S3"|  // Attention => en vrai des références
```

14. `oss.remove("toto");` ⇒ valeurs :

```
| 0  | 1  | 2  |
+----+----+----+
|"S1"|"S2"|"S3"|  // Attention => en vrai des références
```

15. `oss.remove("S2");` ⇒ valeurs :

```
| 0  | 1  |
+----+----+
|"S1"|"S3"|  // Attention => en vrai des références
```

16. `oss.remove("S1");` ⇒ valeurs :
17. `oss.remove("S3");` ⇒ valeurs :

```
||
+----+----+
||
```

Certaines méthodes de la classe sont déjà données :

3.2. Réalisation

- On vous donne la méthode `trouveRang(String s)` en version non optimisée.
- On vous donne aussi : le constructeur, `size`, `toArray`.

Ecrire le corps des méthodes suivantes :

- `contains`, `add`, `remove`.
- Ecrire les méthodes qui renvoient **un nouvel objet OrderedStringSet** : `union` (receveur union paramètre), `difference` (receveur "moins" paramètre), `intersect` (receveur intersection paramètre)
  - Ces méthodes sont elles des transformateurs ? des accesseurs (observateurs) ?
- Récrire `trouveRang(String s)` avec une recherche par dichotomie.

Dernière mise à jour 2015-03-20 10:40:58 CET