Université Paris 7 - Denis Diderot ED
Licence L3 Année 2010-2011, 2ème semestre

TP n°2

`JUnit` et débogage

Exercice 1 Créez une classe `Tableau` pour représenter des multi-ensembles d'entiers. Dans un multi-ensemble, contrairement à un ensemble un même élément peut apparaître plusieurs fois. Elle a deux attributs `tab` de type tableau d'entiers et `taille` de type entier (`taille` représente le nombre d'éléments présents dans le tableau).

À l'initialisation, `tab` est un tableau de taille 20 et `taille` vaut 0. Si on a besoin d'ajouter des éléments, on associe avec `tab` un tableau dont la taille est le plus petit multiple de 20 pour lequel on a assez de place pour insérer les éléments que l'on veut. Par exemple, si on veut représenter l'ensemble $\{4, 3, 5, 2, 7\}$, l'attribut `tab` va être un tableau dont `tab.length=20` et `taille` est égale à 5. Lorsque l'on ajoute un élément au tableau la taille augmente et lorsque l'on en enlève un la taille diminue.

Rajoutez les méthodes suivantes :

- `int getTaille()`,
- `Tableau copier()` qui renvoie une copie du tableau,
- `void addVal(int v)` et `void addVal(int[] v)` qui permet soit d'ajouter un élément soit un ensemble d'éléments,
- `void removeVal(int v)` pour retirer la valeur v du tableau (si v apparaît plusieurs fois on ne retire qu'une occurence), qui renvoie `true` si v est dans le tableau et `false` sinon,
- `int getVal(int index)` et `int setVal(int index, int v)` (la valeur retournée par `setVal` représente la valeur ancienne de `tab[index]` ou -1 si `index` n'est pas une position du tableau comprise entre 0 et `taille`) ; ces méthodes lanceront une exception de type `TableauIndiceEnDehorsDesBornesException` lorsque les paramètres ne correspondent pas à la taille du tableau,
- `void union(Tableau t)` et `void intersection(Tableau t)` qui réalisent l'union et l'intersection de deux tableaux. Pour l'intersection, vu que l'on manipule des multi-ensembles, un élément apparaît i fois dans l'intersection si il apparaît au moins i fois dans les deux multi-ensembles dont on fait l'intersection et exactement i fois dans l'un des deux.

Exercice 2 Créez une classe de test qui étende `TestCase` pour tester les méthodes que vous avez implémentées.

Exercice 3 Utilisez `javadoc` pour créer la documentation de votre classe.

Exercice 4

1. Rajoutez une méthode `int split(int val)` qui permute les valeurs dans `tab` et retourne un entier `ind` tel que : (1) sur la position `ind` de `tab` on retrouve la valeur `val`, (2) sur toutes les positions à gauche de `ind` on retrouve des valeurs inférieurs ou égales à `val` et (3) sur toutes les positions à droite de `ind` on retrouve des valeurs strictement supérieures à `val`. Cette méthode lancera une exception si `val` n'est pas une valeur présente dans `tab`.

2. Ecrivez une méthode `bool triRapide(int premier,int dernier)` qui trie le tableau entre les positions `premier` et `dernier`. On devrait choisir un pivot, appeler `split` sur pivot et appeler récursivement `triRapide` pour trier le tableau entre `premier` et la position `ind` retournée par `split` et entre `ind+1` et `dernier`.

Utiliser le debogueur pour suivre le comportement du code que vous écrivez (en particulier les différents appels récursifs).