Génie Logiciel Avancé

Année 2014-2015

Prof. Burkhart Wolff
wolff@lri.fr

Thibaut Balabonski, Delphine Longuet
blsk@lri.fr, longuet@lri.fr

TP - Test unitaire avec JUnit

Semaine du 27 octobre 2014

L'objectif de ce TP est d'écrire et d'exécuter des tests avec JUnit pour une classe Java dont les instances sont des tableaux redimensionnables. À partir de la spécification informelle donnée, vous devez écrire un ensemble de classes JUnit de façon à pouvoir tester des implantations en boîte noire. Les squelettes des classes de test ainsi que les implantations à tester sont disponibles à l'adresse

`http://www.lri.fr/~longuet/Enseignements/14-15/TestRarray.zip`

Spécification On considère la classe `Rarray` implantant des tableaux redimensionnables. Une instance de la classe `Rarray` est un tableau dont les valeurs sont des objets supposés non `null`. Un même objet peut être présent plusieurs fois dans le tableau.

Le constructeur de la classe `Rarray` permet d'initialiser un Rarray vide de la capacité initiale passée en argument, qui doit être strictement positive. Il est toujours possible d'ajouter un élément à un Rarray, sa capacité est augmentée si besoin. On peut supprimer une occurrence d'un objet avec la méthode `remove` ou toutes les occurrences d'un objet avec la méthode `removeAll`. Ces deux méthodes renvoient vrai si un élément a effectivement été supprimé, faux si l'élément passé en argument n'est pas présent dans le tableau. Il est également possible de vider entièrement un Rarray avec `clear`. La méthode `contains` permet de savoir si un élément est présent dans un Rarray, la méthode `nbOcc` donne le nombre d'occurrences d'un objet et la méthode `size` donne le nombre total d'éléments présents dans le tableau.

Le squelette de la classe Java `Rarray` est le suivant.

```
public class Rarray {

    public Rarray(int capacite) throws RarrayError { }
    public void add (Object elt) { }
    public boolean remove (Object elt) { }
    public boolean removeAll (Object elt) { }
    public void clear() { }
    public boolean contains (Object elt) { }
    public int nbOcc(Object elt) { }
    public int size() { }

}
```

Introduction à JUnit 4 JUnit est un outil permettant d'écrire et d'exécuter des tests unitaires sur des programmes Java. Il est intégré à Eclipse mais est égalenemt disponible à l'adresse `http://www.junit.org/`.

Un test en JUnit 4 est une méthode annotée par `@Test`. Les méthodes de test sont généralement regroupées en une classe dédiée aux tests. Le corps d'une méthode de test doit comporter quatre parties :

- le *préambule*, qui permet de créer les objets et de les amener dans l'état nécessaire pour le test ;
- le *corps de test*, dans lequel la méthode à tester est appelée sur les objets créés ;
- l'*identification*, qui permet de délivrer le verdict du test (succès ou échec) en vérifiant un ensemble de propriétés (assertions) sur l'état des objets après le test. Le tableau 1 résume les différentes assertions possibles en JUnit.
- le *postambule*, qui réinitialise les objets.

Il est possible de grouper les tests ayant un préambule commun (c'est-à-dire devant être exécutés dans le même état) en une classe et de définir une méthode qui exécutera ce préambule avant chacun des tests de la classe. Cette méthode doit être annotée par `@Before`. De la même manière, si tous les tests d'une classe ont un postambule commun, on peut définir une méthode annotée par `@After` qui sera exécutée après chacun des tests de la classe.

Mise en route. Sous Eclipse, créez un nouveau projet java. Créez un package `test` dans `src` et importez dans ce package les fichiers du répertoire TestRarray fourni : clic droit sur le package > Import > File System puis dans le répertoire TestRarray, sélectionnez tous les fichiers. Ajoutez ensuite JUnit 4 au *classpath* : clic droit sur le projet > Build Path > Add Librairies > Junit4. Enfin ajoutez une des implantations de la classe `Rarray` fournies : clic droit sur le projet > Build Path > Add External Archives, puis ajouter le fichier `rarray13.jar` du répertoire Implantations, par exemple.

Ouvrez les trois classes Java fournies. Par exemple, la classe `TestRarrayVide` contient la méthode `ajouterVide`, qui vérifie qu'il est possible d'ajouter un objet dans un Rarray vide. Après avoir ajouté l'objet, on vérifie qu'il est présent en un exemplaire et que le nombre d'éléments est passé à 1. Le préambule du test se trouve dans la méthode `creerRarrayVide`. On remarque qu'on rattrape une éventuelle exception dans le corps de la méthode `creerRarrayVide`, au cas où l'initialisation échoue.

Pour exécuter ces tests sur une implantation en boîte noire (une archive .jar), il faut ajouter le fichier au *classpath* comme vu précédemment. Exécutez ensuite `TestRarrayVide` en tant que test JUnit sur l'implantation `rarray13.jar`. Le résultat des tests apparaît dans un nouvel onglet : un test ayant levé une exception non rattrapée est répertorié dans *Errors*, un test ayant échoué (*AssertionFailedError*) est répertorié dans *Failures*.

Exercice 1

1. Complétez les squelettes des trois classes de test fournies en suivant les exemples donnés. Pour chacun des tests, vous préciserez *obligatoirement* en commentaire l'objectif du test ainsi que le résultat attendu. Pensez à tester aussi bien les cas qui doivent réussir que les cas qui doivent lever une exception : l'objectif est de couvrir un maximum de cas. Pensez également aux cas aux limites.
2. Exécutez vos tests sur chacune des implantations fournies et rédigez un rapport de test sous la forme d'un tableau : pour chaque implantation, dites si les tests ont réussi ou échoué et donnez les raisons apparentes des fautes trouvées. Vous pouvez suivre le modèle suivant :

Implantation	Résultats des tests	Fautes trouvées
rarray13	Échec	Supprimer un objet présent plusieurs fois supprime toutes les occurrences de l'objet

Exercice 2 : complément à la classe `Rarray`

On ajoute à la classe `Rarray` des méthodes permettant de manipuler les objets du tableau en fonction de leur indice. La méthode `index` permet de connaître l'indice d'une occurrence d'un objet et la méthode `get` permet de connaître l'objet présent à un indice. On peut supprimer l'objet se trouvant à un indice donné avec `removeInd` et remplacer un objet par un autre à un indice avec `replace`. Si l'objet dont on cherche l'indice n'est pas présent dans le tableau, la méthode `index` lève l'exception `ObjectNotFound` qui hérite de l'exception `RarrayError`. Les trois autres méthodes lèvent l'exception `OutOfRarray` également héritée de `RarrayError` si l'indice passé en paramètre est hors du tableau ou si aucun élément n'est stocké à cet indice.

```
public class Rarray {

    ...
    public int index(Object elt) throws ObjectNotFound { }
    public Object get(int ind) throws OutOfRarray { }
    public Object removeInd(int ind) throws OutOfRarray { }
    public Object replace(int ind, Object elt) throws OutOfRarray { }

}
```

Complétez vos tests pour ces méthodes, puis exécutez de nouveau vos tests sur les implantations fournies. Complétez le rapport de test avec les nouvelles erreurs trouvées.

À la fin du TP Envoyez par mail à Thibaut Balabonski [1] une archive .zip contenant :
- les trois classes de test complétées ;
- votre rapport de test.

Méthode	Rôle
`assertEquals(Object a, Object b)`	Vérifie que les objets a et b sont égaux
`assertSame(Object a, Object b)`	Vérifie que a et b sont des références vers le même objet
`assertNotSame(Object a, Object b)`	Vérifie que a et b ne sont pas des références vers le même objet
`assertNull(Object o)`	Vérifie que l'objet o est `null`
`assertNotNull(Object o)`	Vérifie que l'objet o n'est pas `null`
`assertTrue(boolean e)`	Vérifie que l'expression e est vraie
`assertFalse(boolean e)`	Vérifie que l'expression e est fausse
`fail()`	Provoque l'échec du test

FIGURE 1 – Méthodes d'assertions en JUnit

1. `blsk@lri.fr`