Thomas LELIEVRE
Anthony LE MÉE

TP1
Unit testing with JUnit

Validation and Verification
University of Rennes 1

Question 1

Cette méthode de test ne prouve en aucun cas que la méthode « remove » fonctionne car on utilise la méthode « add » avant le « remove » sans être certain qu'elle-même fonctionne vraiment (ie. sans être elle-même prouvée).

Question 2

Idem que dans la question 1, la méthode « remove » n'étant pas prouvée on ne peut garantir que la méthode « add » soit correction.

Question 3

Il faut préalablement faire une méthode de test sur la méthode « add » afin de prouver qu'elle fonctionne. Ensuite, il faudra tester dans ma méthode « remove », après un add() d'un élément, le fait que l'élément supprimer l'est effectivement, que la taille diminue bien de 1, et pour finir que les autres éléments sont bien tous présents et qu'ils sont bien rangés.

Question 4

Non les tests sont exécutés dans un ordre aléatoire par JUnit. Il est donc inutile de se forcer à respecter un ordre précis dans l'écriture des méthodes de tests.

Question 5

Non, car les appels de la méthode « add() » ne couvrent pas forcément tout le code de la méthode addBefore. Nous avons peut être testé entièrement la méthode « add » mais ce n'est peut-être pas le cas de la fonction « addBefore » à tester. Des méthodes de tests indépendantes et entièrement destinées à « addBefore » seront obligatoires.

Question 6

Non, les tests ne seront donc pas suffisants car une méthode non couverte est une méthode non testée.

Oui il est possible de ne jamais couvrir un code en entier s'il y a du code mort dedans (ie. non accessible).

Cela dépend mais un code couvert à 100% n'est pas forcément sûr. L'essentiel est tout d'abord que les tests soient bien construits et qu'ils soient munis d'un bon jeu de données afin de prendre en compte tous les cas possibles.

Question 7

Oui on peut faire mieux en utilisant un bon jeu de données mais aussi en utilisant d'autres systèmes de couverture tels que le MC/DC.

Critères qualifiant un test comme bon :

- Possédant un bon jeu de données
- Couvre au moins l'ensemble du code

- Couvre l'ensemble des prédicats
- Couverture MC/DC

Bugs référencés

set(int, Object)	**l.288**	TestPhonyArrayList#**testSet**() Incrémentation sur la variable « index » inutile.
contains()	**l.288**	TestPhonyArrayList#**testContains** () Erreur dans le return
add(int,E)	**l.319**	TestPhonyArrayListAdd#**testAddEmptyList**() Oubli d'insérer l'élément à l'indice voulu.
addAll(int, Collection)	**l.451**	TestPhonyArrayListAdd#**testAddAllAtPositionEmptySource**() Branche morte.
remove(object)	**l.362**	TestPhonyArrayLIstRemove#**testRemoveNullObjectInTwoElementList**() Erreur dans la condition du if.
removeAll(Collection<?>)	**l.563**	TestPhonyArrayLIstRemove#**testRemoveAllCollectionOfObjectInOneElementList**() Erreur dans un for.