Damien JOUANNO
Herminael ROUGIER

ESIR 3
20/11/2012

# Validation and Verification

## Unit testing with JUnit

1 Preliminary questions

*Question 1 This result cannot prove in any way that the method “remove” is correct. Why?*

Ce test n'est pas un test unitaire correct car il repose sur l'utilisation de plusieurs méthodes de l'objet MyLinkedList. La méthode n'est jamais testée et, si elle a comportement défectueux, il est impossible d'affirmer que le comportement de la méthode remove() soit correct.

*Question 2 Does it prove that there is an error in the method "add"?*

Non car l'erreur peut provenir soit de add() soit de remove() mais l'assertion ne permet pas de savoir laquelle des 2 méthodes n'a pas un comportement correct. Par exemple :

- si les 2 méthodes sont inactives, le test sera correct malgré un comportement erroné
- si add() n'est pas correcte mais que remove() l'est, ou inversement, le test sera en échec sans être capable de déterminer laquelle des 2 méthodes n'a pas le comportement attendu.

*Question 3 According to the name of the test case, the intent is to test the method "remove". What conditions must be met in order for this test to be effective?*

Toutes les méthodes utilisées par le test de la méthode remove() doivent disposer de leurs propres tests unitaires qui assurent leur bon fonctionnement. Concernant la méthode size() qui est un getter, dans notre cas, on considère que celle-ci est correct d'office car l'attribut qu'elle retourne n'est pas public et son comportement est très simple (seulement un return).

*Question 4 Does the order of test methods within the class MyLinkedList matter? Why?*

Oui car des tests sont dépendants d'autres tests. Par exemple, il est préférable de tester le bon fonctionnement de la méthode add() avant celui de la méthode remove() car cette dernière dépend de la première. En revanche, certains cas ne peuvent être résolus car les tests sont interdépendants. Par exemple, le test de la méthode get() a besoin d'utiliser la méthode add(), et le test de la méthode add() nécessite l'utilisation de la méthode get() pour vérifier le bon ajout de l'élément.

*Question 5 Suppose that we have adequately tested the method add(Object o) and that it appears to be correct. We note, by analyzing the code, that it calls the "addBefore" method. Can we also consider this method as sufficiently tested? Why?*

Oui dans la mesure ou les tests sur la méthode add on été concluant et qu'il n'ont pas mis en évidence un bug. On peut en conclure que la méthode addBefore replit son rôle dans le

cadre de la méthode add.

2 Unit testing

*Question 6 If the percentage of code covered by all of your test case is not 100%, can you nonetheless consider your test set as sufficient? Can it be impossible to cover the entire code? If yes, give an example. Do you think that code coverage is a good metric?*

On ne peut obtenir une couverture de 100 %, et cela pour plusieurs raisons : il peut y avoir du code mort, donc impossible de tester ce code puisque l'on ne peut l'appeler. De plus, dans certains cas, on ne peut passer dans certains conditions. Elles sont impossible à atteindre.

# Liste des bugs trouvés :

- La méthode contains est erroné, ce n'est pas un supérieur stricte mais un supérieur ou égale. Cette erreur a été détecté par le test `testContains`
- Une erreur dans la méthode set, on ne doit pas incrémenter la variable index (elementData[index] et pas ++index). Cette erreur a été détecté par le test `testSet`
- Dans la méthode add, on doit rajouter un set en plus pour réaliser les changements. Cette erreur a été détecté par le test `testAddAt`
- Dans la méthode batchRemove, on doit ajouter les differentes Exception requises : NullPointerException et ClassCastException. Cette erreur a été détecté par le test `testRemoveAllWithNullPointerException et testRemoveAllWithClassCastException.`
- Dans la méthode batchRemove, il y a une erreur sur la variable elementData (tableau d'objet), on va la remplacer par une variable elementDataTmp. Cette erreur a été detecter dans le test `testRemoveAll`