Report TP1 of V&V

Apprentice : Nicolas Duquesne

1 Preliminary questions

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

On ne teste pas la méthode add avant.

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

Non

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?

Il faut que la méthode add fonctionne correctement pour vérifier la méthode remove.

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

Non, en Java, l'ordre des méthodes n'a pas d'importance et les annotations @Before et @After dans la classe de test précisent bien à quel moment elles doivent être éxecutées.

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?

En testant la méthode add(Object o) qui appelle la méthode addBefore, on test les deux méthodes en même temps. La méthode addBefore est utile au bon fonctionnement de la méthode add, si le résultat des test de add est concluant, addBefore doit bien faire son travail. Maintenant si addBefore est utilisé ailleurs, il sera peut-être nécessaire d'effectuer des tests

plus poussés sur celle-ci.

2 Unit testing

1.

Dans la Methode **public E set(int index, E element)**, ligne 235 :

⇒ Il faut enlever le ++ : elementData[++index]=element; ⇒ elementData[index]=element;

```
@Test
public void set_list() {
	// Creating a call context
	PhonyList<Integer> list = list(75, 93, 0, 46, -56);

	// Oracle
	assertEquals((Integer)46, (Integer)list.set(3, 28));
	assertEquals((Integer)28, (Integer)list.get(3));
}
```

2.3 Final question

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?

Mes tests couvrent 75% de couverture, on peut considérer que c'est suffisant dans la mesure où il impossible de couvrir et tester l'ensemble du code, par exemple pour tester la méthode hugeCapacity afin d'éviter de dépasser la capacité max définie dans la classe Integer (MAX_VALUE = 0x7fffffff), il faut remplir une liste avec énormément d'éléments pour vérifier que le dépassement n'a pas lieu. En bref on peut difficilement couvrir l'ensemble des possibilités et combinaison, c'est mathématiquement impossible.

La couverture du code est une bonne mesure mais cela ne doit pas être la seule, le test unitaire est une partie des tests à effectuer.