

TP 1 : Test unitaire avec *JUnit* et *EclEmma*

Le but de ce TP est la mise en application des techniques de test unitaire. Vous aurez pour cela à tester une classe afin d'y détecter des erreurs, en utilisant des tests au format *JUnit*. Au cours de ce TP vous utiliserez également l'outil *EclEmma* pour évaluer le degré de couverture des cas de test que vous écrivez.

Cette séance a pour but de vous familiariser avec le framework de test pour Java *JUnit* et l'outil de couverture de code *EclEmma*.

Vous trouverez sur la page web du cours (<http://www.combemale.fr/teaching/tql/>) le code source de la classe à tester. La classe sous test est une liste chaînée dont l'interface est donnée Figure 1. Avant de commencer, veuillez répondre aux questions préliminaires ci-dessous et bien lire les consignes quant au travail à rendre.

1.1 Questions préliminaires

La suite de test *JUnit* (V4) suivante a pour résultat FAUX lors de son exécution *JUnit*.

```
public class TestLinkedList{

    protected MyLinkedList list;

    @Before
    public void setUp() {
        list = new MyLinkedList();
    }

    @After
    public void tearDown() {
        list = null;
    }

    //test de la méthode remove(Object) sur une liste à 1 elt
    @Test
    public void testRemove() {
        Object o = new Object();
        list.add(o);
        list.remove(o);
        assertEquals("test de la méthode Remove",
                    list.size(), 0);
    }
}
```

Question 1 :

- A priori, ce résultat ne prouve en aucun cas que la méthode "remove" est erronée. Pourquoi ?
- Prouve-t-il qu'il existe une erreur au sein de la méthode "add" ?
- D'après le nom de ce cas de test, son intention est pourtant le test de la méthode "remove". Quelle(s) condition(s) doit-on vérifier pour que ce soit effectivement le cas ?

Question 2 :

- L'ordre de test des méthodes au sein de la classe *MyLinkedList* a-t-il une importance ? Pourquoi ?
- Supposons que l'on ait suffisamment testé la méthode "add(Object o)" et qu'elle nous paraisse correcte. On remarque, en analysant son code, qu'elle appelle la méthode "addBefore". Peut-on aussi considérer cette méthode comme suffisamment testée ? Pourquoi ?

MyLinkedList
- size: int
+ MyLinkedList(c: Collection)
+ MyLinkedList()
+ add(index: int, element: Object)
+ add(o: Object): boolean
+ addAll(index: int, c: Collection): boolean
+ addAll(c: Collection): boolean
+ addFirst(o: Object)
+ addLast(o: Object)
+ clear()
+ contains(o: Object): boolean
+ get(index: int): Object
+ getFirst(): Object
+ getLast(): Object
+ indexOf(o: Object): int
+ lastIndexOf(o: Object): int
+ listIterator(index: int): ListIterator
+ remove(index: int): Object
+ remove(o: Object): boolean
+ removeFirst(): Object
+ removeLast(): Object
+ set(index: int, element: Object): Object
+ size(): int

Figure 1 - Spécification de la liste

1.2 Test unitaire du système de liste chaînée

1.2.1 Consignes pour le travail à réaliser

Vous devrez obtenir :

- L'ensemble des cas de test définis numérotés suivant l'ordre de création opéré et accompagnés d'un commentaire précisant l'intention du cas de test (par exemple : « test de la méthode size() pour une liste vide »).
- La localisation et la correction des erreurs trouvées en précisant bien le numéro du cas de test ayant permis la détection. Vous présenterez ces résultats sous la forme d'un tableau (1^{ère} colonne : méthode/ligne, 2^{ème} colonne : numéro du cas de test, 3^{ème} colonne : ligne corrigée).
- Le pourcentage du code couvert par l'ensemble des cas de test.
- Les réponses aux questions de la partie 1.2.2 Bilan.



- Toute méthode publique doit faire l'objet d'au moins un cas de test
- N'oubliez pas, pour chaque cas de test, de faire figurer en commentaire son intention.

1.2.2 Bilan

Question 3 :

Si le pourcentage du code couvert par l'ensemble de vos cas de test n'est pas de 100%, pouvez-vous tout de même considérer votre jeu de test comme suffisant ? Ce peut-il qu'il soit impossible de couvrir la totalité du code ? Si oui, donnez un exemple. La couverture de code vous paraît-elle être un bon critère de test ?

Question 4 :

Si la couverture de code est un mauvais critère, comment peut-on le vérifier ? Connaissez-vous une technique d'analyse permettant de qualifier un critère de test ?