

# Vérification et Validation

## TP1

### Question 1 :

On ne peut pas être sûr que la fonction `remove` fonctionne correctement puisque nous n'avons pas testé la méthode `add` avant (un élément non ajouté est forcément absent).

### Question 2 :

Non, ce test prouve qu'il y a une erreur, mais elle peut tout aussi bien être dans le `remove` que dans le `add`.

### Question 3 :

Pour être sûr que la fonction `remove` fonctionne correctement, il faut tout d'abord s'assurer que la fonction `add` est correcte.

### Question 4 :



L'ordre des tests n'est pas important. Par contre, il faut bien faire attention aux dépendances des différents tests. (Il est plus logique de faire les tests sans dépendances d'abord.)

### Question 5 :

La méthode `addBefore` n'est pas suffisamment testée. En effet, il faut s'assurer que l'élément ajouté est bien devant tous les autres.

### Question 6 :

Grâce à Jacoco, nous savons que nos tests couvrent la classe à 57 %. Nous ne pouvons pas sortir le rapport pour des raisons techniques, mais le plugin fonctionne dans l'EDI. Nous allons donc prouver son fonctionnement avec quelques captures :

 PhonyList	 57,2 %	1 243	931	2 174
---	--	-------	-----	-------

Voici un exemple de fonction passée dans Jacoco :

```

public boolean equals(Object o) {
    if (o == this)
        return true;
    if (!(o instanceof PhonyList))
        return false;

    PhonyList list2 = (PhonyList) o;

    if (this.size() != list2.size())
        return false;

    for (int i = 0; i < this.size(); i++) {
        Object o1 = this.get(i);
        Object o2 = list2.get(i);
        if (!(o1 == null ? o2 == null : o1.equals(o2)))
            return false;
    }

    return true;
}

```

De plus, nous pouvons dire que le code n'est pas possiblement couvert à 100 % a cause de la fonction `remove()`. En effet, si on passe dans le `if`, alors il est impossible de passer dans la condition imbriquée.

Enfin, couvrir tout le code ne garantie pas qu'il n'y a pas d'erreur dedans. Il est parfois nécessaire de ne couvrir que les cas d'utilisateurs utilisables, comme nous avons fait ici.

## Liste des erreurs trouvées

Test	Méthode testée	Description de l'erreur
set_list()	set()	Lorsque l'on ajoute un élément dans la liste et que l'on ré-effectue un get() avec le même index, alors l'élément retourné n'est pas celui escompté (ajouté). Ligne 198
contains_list()	contains()	Le premier élément de la liste n'est pas pris en compte avec la méthode contains() Ligne 331
removeAll_list()	removeAll()	Après la suppression de tout les éléments sauf un dans une liste, sa taille n'est pas de un. La méthode ne prend pas en compte le premier élément de la liste pour la suppression. Ligne 296
contains_listOne()	contains()	La méthode contains() ne fonctionne pas avec un seul élément de la liste. Cela est dut au fait que le premier élément de la liste n'est pas parcouru. Ligne 347