

LOG3430 – Méthodes de tests et validation de logiciels

TP2- Test boîte noire d’une application

Auteurs

Thierno Barry 1550237

Tomas Costanzo 1578677

Automne 2016

9 Octobre 2016

# Introduction

Dans ce deuxième laboratoire du cours de LOG#430(Tests et validation de logiciels), nous avons à analyser et implémenter des tests fonctionnels (boite noire) du projet ListeChainnee implémenté au TP1 . Nous avons utilisé la technique et la méthodologie de description utilise dans le cours pour effectuer et présenter l’analyse.

# Analyse selon la methode category-partition :

Nous allons faire reference à l’énonce du TP1 pour avoir les specifications du code a tester.

## ListeChainee (Methode build)

Operateur :

Op1 : Union [Prop-Union]

Op2 : Intersection [Prop- Intersection]

Op3 : Difference [Prop- Difference]

Op4 : Symmetric Difference [Prop- SymmetricDif]

Op5 : is subset [Prop- is subset]

Op6 : is superset [Prop- is superset]

Op7 : Autres [Error]

Val1 :

V1\_1 : Vide [Prop-Vide]

V1\_2 : Liste non-vide [Prop- Normal]

V1\_3 : Liste avec doublons [Prop-doublon] [if Non-Vide]

V1\_4 : Invalide [Error]

Val2 :

V2\_1 : Vide [Prop-Vide]

V2\_2 : Liste non-vide [Prop-Normal]

V2\_3 : Liste avec doublons [Prop- Doublon] [if Non-Vide]

V2\_4 : Invalide [Error]

Le cas de test avec une valeur invalide ne peut pas être teste car cette fonction prend comme paramètres Val1 et Val2 des objets.

## Add

E1 : Nomal [Prop-Normal]

E2 : Ensemble Invalide [Error]

Le cas de test avec une valeur invalide ne peut pas être teste car cette fonction donc on fait un seul test avec un ensemble normal.

## RemoveAtPosition

Pos1 : Position valide [Prop-Normal]

Pos2 : Position Invalide [Error]

## RemoveItem

E1 : Ensemble valide [Prop-Normal]

E2 : Ensemble inexistant [Error]

## SetAt

P1 : Position valide [Pos-Normal]

P2 : Position invalide [Error]

## GetAt

P1 : Position valide [Pos-Normal]

P2 : Position invalide [Error]

## GetSize

S1 : Liste vide [Vide]

S2 : Liste normale [Size-Normal]

## Reset

Un seul test qui vérifie si la liste est vide après l’exécution de la méthode.

# Implémentation des tests

Les tests sont implémentés dans 3 fichiers différents. Le fichier ListeChaineeImplTest\_AC implémente les test AC de la méthode build de la classe ListeChainee. Le fichier ListeChaineeImplTest\_EC implémente les test EC de la méthode build de la classe ListeChainee. Le fichier MyListImplTest.java implémente les tests pour les méthodes de la classe ListeImpl.

## Test EC (Each choice):

### ListeChainee

Test 1 : Op1 , V1\_2 , V2\_2

Test 2 : Op2 , V1\_2 , V2\_2

Test 3 : Op3 , V1\_3 , V2\_2

Test 4 : Op4 , V1\_1 , V2\_2

Test 5 : Op5 , V1\_2 , V2\_2

Test 6 : Op6 , V1\_2 , V2\_2

Test 7 : Op7 , V1\_2 , V2\_2

### Add

Test 1 : E1

### RemoveAtPosition

Test 1 : Pos1

Test 2 : Pos2

### RemoveItem

Test 1 : E1

Test 2 : E2

### SetAt

Test 1 : P1

Test 2 : P2

### GetAt

Test 1 : P1

Test 2 : P2

### GetSize

Test 1 : S1

Test 2 : S2

### Reset

T1 : Vérifier si la liste est vide

## Test AC (All choice)

Pour les méthodes : add, removeAt, removeItem, setAt, getAt, getSize et reset les test AC et EC sont les pareils.

Pour la méthode ListeChainee (build) il y a 6 \* 3 \* 3 = 54 Test

Des commentaires **@javadoc** ont ete ajoutes au début de chaque fonction de test pour faciliter la compréhension et améliorer la lisibilité.

# Conclusion

Ce travail pratique nous a permis de nous familiariser et de mettre pratique les notions acquises sur les tests fonctionnels (Black box ) dans le cadre du cours.