TP COUVERTURE – Partie I

Numéro de Référence
#UNIVPM001

(Document de 11 pages)

Résumé

Ce rapport présentera le résultat de nos travaux sur la première partie du TP Couverture. Il synthétisera sous forme d'un compte-rendu notre éxperience de tests avec l'outil Eclemma.

Mots Clés

Couverture de code, Tests unitaires, Junit, Eclemma, TP2 Partie 1, FSIL.

Université Aix-Marseille
Centre Informatique
Luminy
13009 MARSEILLE CEDEX 13

SECTION DES REDACTEURS

Nom	Prénom	Contribution
MESTRALLET	Alexis	
RISCH	Philippe	

CONTACTS

Nom	Prénom	Email	Fonction
MESTRALLET	Alexis	Mestrallet.alexis@gmail.com	Etudiant FSI
RISCH	Philippe	Philippe.13009@yahoo.fr	Etudiant FSI

HISTORIQUE DES MODIFICATIONS

Modifications	Date	Version	Approbateur de la diffusion

TABLE DES MATIERES

1. ETUDE DE COUVERTURE DE CODE

Dans un premier temps nous avons consulté rapidement la documentation sur le site d'Eclemma, après quoi nous avons construit notre projet en récupérant les fichiers sources du TP précédent.

1.1. Un Code Simple

Nous avons donc lancé la série de tests de couverture de code sur le fichier PartialCoverTest avec Eclemma et obtenu les résultats suivants :

- 68.0% de couverture pour la classe PartialCover.java
- 94.5% de couverture pour la classe de test associée

Nous avons alors implémenté de nouveaux tests afin d'améliorer la couverture de code (i.e parcourir le maximum de branche sur le diagramme de flots résultant). Nous avons ainsi ajouté un test d'égalité ainsi qu'un test de supériorité. Ce qui nous a amené à une couverture de code plus complète : 97.0% pour les tests. En revanche, le taux de couverture de code de la classe n'a pas changé, ce qui nous permet d'affirmer qu'il était impossible de parcourir l'ensemble des branches du diagramme de flots résultant de ce programme.

1.2. Un Code Inconnu

Nous avons ensuite lancé une série de test de couverture de code sur un code inconnu (écrit par un tiers). Le but est de debugger le code source à l'aide de test unitaire et de test de couverture de code afin de nous assurer que nous testons bien le maximum de code.

1.2.1. Recommandation d'écriture de test unitaire

On remarque que certain des tests n'ont pas été nommé correctement. En effet les conventions de test veulent que le nommage des fonctions de tests soient explicites.

1.2.2. Utilisation d'eclemma

Après utilisation d'Eclemma sur le code, nous obtenons les résultats suivant :

- Test : 100 %
- Classe : 52,5 %

On remarque que Eclemma se base bien sur les test unitaires. Cette outil ne permet pas de déterminer la couverture de code maximale possible pour une classe.

1.2.3. Test du constructeur de la classe StringArray

Après ajout de notre test nous obtenons une couverture de code de 90,1 %.

1.2.4. Ajout de tests supplémentaires

Nous avons ensuite ajouté de nouveaux tests afion de parcourir des instructions non couvertes par les tests précédents.

Aini nous nous sommes rendu compte qu'il restait une branche non parcourue, celle du tableau vide et que le programme était buggé aux endroits précédemment non couvert par les tests.

1.2.5. Correction de la classe StringArray

Une fois le code couvert à 100%, nous avons pu passer au debuggage du constructeur. Nous avons ainsi trouvé trois problèmes majeurs :

- Une fois le tableau trié, la recherche d'éléments dupliqués ne couvrait pas l'ensemble du tableau, donc tous les éléments dupliqués n'étaient pas trouvés.
- Le code levait une exception : OutOfBoundException.
- Il manquait l'ajout du dernier élément du tableau list dans le tableau uniques.

BIBLIOGRAPHIE ET LIENS

http://www.eclemma.org

ANNEXES

GLOSSAIRE

INDEX

TABLES DES FIGURES

LISTE DES TABLEAUX