Faculté des Sciences
Aix*Marseille Université

Département Informatique et Interactions
Année 2020-21

Master 1 Informatique
Fiabilité Logicielle

TP Couverture de Tests

Le but de ce TP est d'utiliser de mesure la qualité de suite de test avec l'outil de couverture de test `EClemma` et d'en comprendre le fonctionnement et les limites.

1 Utilisation d'un outil de couverture

1.1 Prise en main

Aller sur la page `http://eclemma.org/` et récupérer la documentation sur l'outil `EClemma` . Vérifier que le plugin est installé sous votre version d'Eclipse, sinon l'installer pour l'ajouter aux bibliothèques de votre projet. Après installation la barre de menu et le menu `Run` doit proposer une option `Coverage`. Bien lire le manuel d'`EClemma` sur le calcul du taux de couverture, en particulier dans le cas des exceptions.

1.2 Exemple 1 : Essai

Récupérer la classe `Essai` sur le site AMETICE et utiliser la classe `EssaiTest` que vous avez écrite au TP précédent.

1. Comprendre le sens des couleurs (vert, rouge, jaune).
2. Comprendre le sens des taux de couverture pour une classe, un package, le projet.
3. Comprendre les couleurs données aux instructions de la classe de test.
4. Pourquoi certaines conditions booléennes sont en jaune ?
5. Est-ce qu'il est possible de couvrir toutes les instructions de la classe `Essai` ?
6. Enlever certains tests et voir comment la couverture évolue.

1.3 Exemple 2 : PartialCovering

Effectuer les mêmes opérations avec la classe `PartialCovering.java`. Que conclure ?

2 Etude de Couverture de code

2.1 Couverture des classes de l'application triangle

Reprendre le TP sur le triangle et de calculer son taux de couverture. Vérifier la couverture de vos tests sur les méthodes `readData` et `typeTriangle`.

1. Analyser les résultats pour la partie compte-rendu.
2. Ajouter des tests pour aller jusqu'à un taux de couverture de 100% si cela est possible. Pour chaque test ajouté, vous signalerez en commentaire quelle instruction supplémentaire il couvre.

2.2 Analyse d'un code inconnu

Cette partie va consister à utiliser les tests pour faire apparaitre certains problèmes dans un code donné mais dont on n'a pas la spécification et à en proposer une/des corrections. Dans la suite d'étapes suivantes, si un test fait apparaitre un comportement anormal, vous devrez corriger celui-ci dans le programme source java. Récupérer les fichiers `StringArray.java`, `TestStringArray.java` sur le site AMETICE. La classe `StringArray` a un attribut qui est un tableau de caractère et un constructeur `StringArray`, les méthodes `getMlist`, `getString`, `indexOf` et méthode `sizeOf` .

1. Sans chercher à comprendre le fonctionnement de la classe, lancer les tests `JUnit` et vérifier qu'ils passent. Ces tests ne respectent pas une des recommandation d'écriture de tests. Laquelle ? Les réécrire pour que cette recommandation soit suivie.
2. Utiliser `EClemma` pour vérifier la couverture des tests. Conclusion ?
3. La méthode `IndexOf` semble retourner l'indice de son paramètre dans le tableau et le constructeur semble trier et éliminer les doublons de la liste donnée en paramètre. Ajouter un test qui couvre la méthode `getString` et les tests permettant de couvrir totalement `IndexOf` si ce n'est pas le cas. Idem pour `sizeOf` et `getString`.
4. Ajouter un test qui prend le i^{eme} élément de la liste `slist1` et vérifier que c'est bien le ^{ieme} élément de l'objet `StringArray` construit à partir de `slist1`.
5. Le constructeur mérite d'être testé plus en détail. Ajouter un test avec le tableau "ab","ab" couvrant la duplication. Résultat ? Quel est le taux de couverture du constructeur avec cette suite de tests.
6. Ajouter un test supplémentaire pour la duplication avec "ab", "c"', "ab". Conclusion ? Est-ce que la couverture du constructeur est totale ?

7. Plusieurs problèmes sont présents. Un problème peut se régler en déplaçant une ligne de code. Est-ce suffisant ?
   (a) Etendre les tests pour augmenter la couverture.
   (b) Identifier les derniers problèmes, corriger et retester.
   (c) Quelle conclusion en tirez-vous ?

3 Travail à rendre

Voir le site pour la spécification des rendus (forme et date).
Le travail à rendre concerne d'une part ce TP et d'autre part un travail à faire pour préparer les TPs suivants.

Partie couverture

1. Une partie du compte-rendu présentera vos conclusions sur la partie `Essai, PartialCovering`. Une autre partie concernera la classe `Triangle` (les tests ont été rajoutés, pourquoi, et le résultat obtenu). La dernière partie décrira ce que vous avez constaté lors de l'analyse du code de la classe `StringArray` (analyse et correction du code et utilisation de la couverture pour détecter de potentielles erreurs).
2. Donner les nouvelles classes de tests pour la classe `Triangle`. Elles seront dans un package `testtriangle`
3. Donner les classes de tests successives et la corrections successives de StringArray. Les tests seront dans un package `teststringarray` et les sources dans `stringarray.`

Partie préparation des TPS .
Pour un TP suivant, il est nécessaire que vous ayez une base de données opérationnelle :

1. Solution 1 : installer sur votre compte une base de données `hsql` (archive sur le site du cours) qui ne demande pas d'être administrateur.
2. Solution 2 : si vous êtes administrateur de votre machine, installer une base de données MySQL ou Postgres.

Dans tous les cas, **vérifier que vous pouvez créer une base de données**. Il est demandé de lire de la documentation sur `jdbc` API permettant de connecter une base de données et Java, par exemple :
`http://jean-luc.massat.perso.luminy.univ-amu.fr/ens/jee/jdbc.html`

La partie compte-rendu donnera un résumé du travail réalisé et indiquer la solution retenue (qui doit fonctionner !).