Faculté des Sciences
Aix*Marseille Université

Département Informatique et Interactions
Année 2017-2018

Master 2 FSIL
Fiabilité 1

TP Couverture de Tests

Le but de ce TP est d'utiliser l'outil de couverture de test *Emma* et d'en comprendre le fonctionnement et les limites.

1 Utilisation d'un outil de couverture

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

2 Couverture du triangle

Il s'agir de reprendre le TP sur le triangle et de calculer son taux de couverture. La classe `Triangle` contient contient les deux méthodes `readData` et `typeTriangle`, des gettters et setters. Deux suites de tests ont été écrites pour cles deux méthodes.

1. Utiliser l'outil de couverture pour calculer le taux de couverture de vos suites de tests.
   (a) Comprendre le retour de l'outil pour identifier ce qui est couvert dans les sources du code testé, du code testeur, du code non testé, et les codes couleurs.
   (b) Si vos tests ont couvert tout le code testé, supprimer des tests (utiliser @Ignore), par exemple pour le cas triangle isocèle pour voir les changements de couverture successifs (passer du rouge au vert via le jaune).
   (c) Quelle conclusion en tirez-vous sur la couverture des conditions booléennes par l'outil.
2. Rajouter éventuellement les tests pour obtenir une meilleure couverture.
3. Supprimer des tests tant que tout le code testé reste couvert.

3 Etude de Couverture de code

3.1 Un code simple

Récupérer les fichiers `PartialCover.java` et `PartialCoverTest.java` et trouver le taux de couverture de la méthode `returnZeroOrOne`. Si nécessaire ajouter des tests pour obtenir le taux de couverture maximal possible.

3.2 Un code inconnu

Récupérer les fichiers `StringArray.java`, `StringArrayTest.java` sur le site du cours. Dans la suite d'étapes suivantes, si un test fait apparaitre un comportemnt anormal, vous devrez corriger celui-ci dans le programme source java. La classe `StringArray` a un attribut qui est un tableau de caractère et un constructeur `StringArray`, les méthodes `getString`, `indexOf` et méthode `sizeOf` .

1. Sans chercher à comprendre le fonctionnement de la classe, lancer les test 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 `getSting`.
4. 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.
5. Ajouter un test supplémentaire pour la duplication avec "ab", "c"', "ab". Conclusion ? Est-ce que la couverture du constructeur est totale ?
6. Ajouter un test qui prend le i^{eme} élément de la liste *slist1* et vérifie que c'est bien le ieme élément de *array1*.
7. Il reste un problème avec la classe *StringArray*. (a) Etendre les tests pour le faire apparaitre, (b) Identifier le problème, (c) Quelle conclusion en tirez-vous ?

4 Un code plus complexe à tester

Récuperer les fichiers `Magasin.java` et `Article.java`. La classe `Article` décrit un article donné par un `nom`, `prix` et `numéro` de nomemclature (on supposera que le numéro de nomenclature est unique). La classe `Magasin` a un attribut stock qui est un tableau d'Articles qui est trié par prix croissant puis numéro de nomenclature croissant. La méthode `chercherDicho` effectue une recherche d'article en utilisant la recherche dichotomique. Plusieurs implementations sont fournies `chercherDicho1,2,...,5.java`.

1. Ecrire une classe de tests pour `Article`.
2. Ecrire les classes de tests permettant de tester chacune des implémentations `chercherDichoi`.
3. Rédiger un rapport de tests permettant d'établir la qualité des tests et donner éventuellement un avis sur les implémentations de la recherche dichotomique (analyse des erreurs s'il y en a).
4. Ecrire une méthode `chercherDichoEtu` qui réalise la recherche dichotomique et la tester.

5 Travail à rendre

Le rendu se fera en deux étapes : un rendu **à la fin du TP** dans l'activité devoir `TPCouverture Partie1` et un à effectuer **avant le jeudi 12 octobre 23h55** dans l'activité devoir `TPCouverture Partie2`. Comme pour le TP précédent, le rendu est une archive zip qui s'appelera `TPCOUVERTURE-PARTIE(1 ou 2)-TPi.zip` et qui crée un répertoire du même nom au désarchivage. **Les fichiers pdf devront respecter le modèle de documents donné sur la page du cours à la rubrique bibliographique**.

1. Le premier rendu contiendra les classes de tests pour `returnZeroOrOne`, les classes de tests pour `StringArray` et un `RapportTPcouverturePartie1.pdf` expliquant tout ce que vous avez retenu sur ces deux exemples (description de l'utilisation du taux de couverture pour améliorer els tests, détection et correction des erreurs).
2. Le deuxième compte-rendu donnera les tests effectués pour `Article` et `chercherDicoi` (les tests devant être identiques, ne donner qu'une seule suite de test pour un i choisi) et le rapport de test. Il contiendra aussi toutes les sources de vos classes de test au format pdf formatées à l'aide d'un formateur comme *a2ps*.

3. Question subsidiaire à répondre dans la partie 2. On peut mettre les tests d'une classe `Toto` d'un package `toto` soit dans un sous-package `testToto`, soit créer un package `test` au même niveau que `src` qui contiendra les différents packages de tests. Discuter les avantages et inconvénients de chaque approche.