Rapport TP2

Rosine Rolande Simo Tegninko, 20183729
Yu Deng, 20151659

Tâche 1

(Q1) Pour le Q1, nous avons choisi deux métriques, NCH et Nombre de tests.

1. NCH fait référence au nombre de commits dans l'historique de la classe. Ensuite, nous pouvons juger s'il est adapté à sa complexité en regardant le nombre de commits. Voici pourquoi : s'il y a trop de commits, alors on sait aussi que la personne qui a écrit ce code le met constamment à jour afin de pouvoir l'adapter à la complexité.

2.Plusieurs tests sont sûrs de trouver de nombreux problèmes. Et pour chaque test, nous devons "augmenter" la complexité de "l'entrée" pour détecter si le niveau de document de la classe est approprié à sa complexité.

(Q2) Pour le Q2, nous avons choisi deux métriques, NLF(Le nombre de lignes de code d'une fonction) et LCOM.

1. De manière générale, une fonction ne doit pas dépasser 40 lignes de code. S'il y a trop de couches de fonctions, cela affectera notre jugement logique lors de la vérification du code. 2. Plus le score LCOM est faible, meilleure est la cohésion de sa structure et meilleur est le degré de corrélation entre les parties du module.

(Q3) Pour Q3, nous avons sélectionné deux métriques, CSEC et Temps total de l'exécution.

1. CSEC fait référence à "couplage simple entre classes", nous savons que plus une classe est couplée à d'autres classes, plus une modification de cette classe peut influencer de classes. Ensuite, CSEC a joué un grand rôle en testant si le code est mature, If ce code peut avoir un CSEC inférieur, la charge de travail des tests sera également réduite.

2. Pour Temps total de l'exécution, lorsque nous testons, le résultat est très lent. Tout le monde veut obtenir une réponse le plus rapidement possible, ensuite si le temps d'exécution est lent, le code est moins mature et moins adapté aux besoins des gens.

(Q4) Pour le Q4, nous avons choisi deux métriques, TPC et NEC.

1. Pour TPC, nous testons en fonction de la classe, donc au lieu de tester directement l'intégralité du code, nous pouvons juger si chaque partie peut être exécutée correctement grâce à des tests automatiques. S'il y a un bogue dans l'un d'entre eux, alors lorsque le code global est automatiquement testé, certains bogues seront trouvés.

2. Pour NEC, il est évident que si un code a beaucoup d'erreurs après son exécution, alors le code doit avoir une erreur dans un certain lien. Lors du test automatique, il a dû rencontrer des problèmes, ensuite, ce code ne fonctionne pas bien dans les tests automatiques.

Tâche 2

On peut répondre à la question 2 en mesurant le CSEC, ce qui est aussi plus facile à faire car on a déjà complété ce code (lcsec.py) dans le premier devoir. Par exemple, si la classe MethodVisitor a un CSEC=3, la sortie du lcsec doit avoir, parmi autres, la ligne :

./gr/spinellis/ckjm/MethodVisitor.java, gr.spinellis.ckjm, MethodVisitor, 3

Tâche 3