MASTER MIAGE 2ème année
Université Paris Nanterre

Mémoire de fin d'études présenté pour
l'obtention du grade de master

Comment les flots de contrôle peuvent-ils nous permettre de faire du refactoring de code en Java.

Présenté par Thibault
Sartre

Tuteur :
...

Septembre 2018 — Juillet 2019

Sommaire

Chapitre 1

Introduction

1.1 Présentation

Le refactoring est une activité d'ingénierie logiciel consistant à modifier le code source d'une application de manière à améliorer sa qualité sans altérer son comportement vis-à-vis des utilisateurs. L'objectif du refactoring est de réduire les coûts de maintenance et de pérenniser les investissements tout au long du cycle de vie du logiciel en se concentrant sur la maintenabilité et l'évolutivité.[4]
"With refactoring you can take a bad design, chaos even, and rework it into well-designed code."[3]
Le refactoring permet donc de passer d'un code possédant de mauvaise base à un code propre.
Un bon refactoring doit pouvoir améliorer la qualité d'un code tout en gardant son fonctionnement du point de vue de l'utilisateur. Concernant la partie des tests, tout les tests qui fonctionnaient avant le refactoring se doivent d'être fonctionnels après.
Dans ce mémoire, nous allons analyser différentes techniques de refactoring. Puis nous allons étudier le principe des flots de contrôle.
Enfin je vous proposerais et évaluerais une solution pour faire du refactoring de code en utilisant les flots de contrôle pour le langage Java.
Ce sujet est intéressant et actuel car il faut commencer à ce soucier du code que l'on produit car plus l'on avance dans le temps plus les programmes sont lourds et contiennent de lignes de code. Avec la puissance des ordinateurs actuels, la plus part des développeurs ne prennent plus le temps d'écrire des codes de qualité car la machine sera de toute façon assez rapide pour compenser un code de mauvaise qualité.[2]
Avec le temps, la relecture et la modification de code sera difficile avec les programmes qui deviennent de plus en plus gros.[2]
Pour essayer de diminuer cette quantité de travail à l'avenir, il faut commencer à produire du code de qualité et bien structuré. Concernant les codes déjà existant il faudra donc faire du refactoring de code. Or le refactoring peut-être long selon la qualité des projets que l'on traite. Il serait donc intéressant et très utile d'avoir un programme permettant d'automatiser des parties de refactoring pour permettre de gagner du temps précieux. Ce mémoire a pour but de fournir une solution qui permettrais de faire du refactoring de code facilement pour gagner du temps tout en produisant du code de qualité.

Bibliographie

[1] Jean-Michel Doudoux. Le refactoring, Janvier 2007. https ://www.jmdoudoux.fr/java/dejae/chap009.htm.

[2] Romain Fallet. Le désenchantement du logiciel, Septembre 2018. https ://blog.romainfallet.fr/desenchantement-logiciel/.

[3] Martin Fowler. *Refactoring : Improving the Design of Existing Code*. Addison-Wesley Professional, 1999.

[4] Jean-Philippe Retaillé. *Refactoring des applications JAVA/J2EE*. Eyrolles, 2005.