TP 1

Refactoring

DANS ce TP, on se propose de mettre en œuvre les techniques de *refactoring*, aliées à un outils de mesure de la qualité logicielle (analyse statique). Dans ce but, nous alons utiliser des *kata*, qui sont des exercices de programmation, souvent disponibles sur le web, dont l'objectif est de s'entrainer en travaillant certaines techniques de programmation ou de conception.

1.1 Kata

Un *kata*[1], par analogie avec les arts martiaux, est un exercice de programmation que l'on répète afin de s'entrainer. On en trouve de différents type, comme ré-implémenter le même problème de différentes manière, corriger des bugs, ou faire du refactoring.

Nous allons utiliser ici *GildedRose*[2], qui est un kata de refactoring connu, pour appréhender les concepts et outils liés au refactoring.

Récupérez l'archive sur l'ENT, qui continent, en plus du code Java, le wrapper Gradle `gradlew` qui facilitera la suite du travail.

1.2 Refactoring

Exercice 1 (Test unitaires) Ajouter des tests unitaires pour spécifier le comportement du programme dans son état courant. Ces tests serviront d'assurance de non régression lors des étapes de refactoring.

Vérifier que tous vos testes passent (par ex. `./gradlew test`). En effet, le code d'origine est considéré valide. Guidez vous des métriques de couverture de code (données par jacoco) pour vérifier que vous avec assez de tests.

1. http://codekata.pragprog.com/2007/01/code_kata_backg.html
2. https://github.com/emilybache/GildedRose-Refactoring-Kata.git

Exercice 2 (Refactoring) Initialisez un dépôt git pour gérer le code source. Il est en effet indispensable de s'appuyer sur un gestionnaire de version lors d'un travail de refactoring. La démarche est de procéder par petites modifications, en faisant un commit chaque fois que les tests passent.

Le fichier build.gradle est configuré pour inclure un certain nombre de plugins qui permettent de mesurer certaines métriques de qualité du code. Elles peuvent être une indication de problèmes au niveau du code et de l'architecture. Prenez donc le temps de les comprendre, et éventuellement les configurer.

En vous guidant notamment des métriques analysées par les plugins gradle, et sur les critères des *smells*[3], modifiez le code du kata afin d'en améliorer la qualité, tout en vous assurant que les tests précédents continuent de passer.

1.3 Travail à rendre

Pour ce TP, vous remettrez le code source de votre travail (avec les tests), ainsi qu'un bref résumé (2-3 pages max) expliquant l'architecture de votre système et la démarche suivie (format texte brut ou PDF), le tout dans une archive ZIP. Vous inclurez également le dépôt git. Attention de ne pas inclure les fichiers dépendant de votre IDE ou votre OS !

Vous préciserez bien les noms des membres du binôme !

3. http://users.csc.calpoly.edu/~jdalbey/305/Lectures/SmellsToRefactorings