Exercice de Clean Code et de réusinage

GLO-2003: Introduction aux processus du génie logiciel

Hiver 2022

1 Consignes

Le but de cet exercise est de mettre en pratique le Clean Code et le réusinage de code (`refactoring`). Pour se faire, nous vous fournissons un code de base à nettoyer et réusiner. Le code de base viole volontairement les principles de Clean Code de Robert C. Martin[1] et présente de mauvaises pratiques de programmation. Vous devez nettoyer le code et le réusiner afin de faciliter l'ajout d'une fonctionalité. Vous devez aussi noter les grandes lignes des modifications que vous avez réalisées pour un retour en classe.

Cet exercice peut être réalisé seul ou en équipe de 2 à 3 personnes. Nous vous conseillons de réaliser l'exercise en `Pair Programming`. Nous vous recommandons aussi de créer un `repository` GitHub pour partager le code entre les membres de l'équipe et de faire un commit par étape. Une solution sera proposée en Java sur le `repository` de l'exercise.

L'exercice est disponible en deux version, la version longue se trouve ici : glo2003/Exercice-CleanCode-Refactoring. La version courte de l'exercice se trouve ici : glo2003/Exercice-CleanCode-Refactoring-Court

2 Contexte

Vous venez d'être engagé en tant que consultant dans une start-up offrant des outils de gestion de personnels. Suite à une croissance rapide accompagnée de nombreux changements à la base de code, ils se retrouvent maintenant avec une productivité abyssale à cause de la pauvre qualité du code.

Votre mandat est de coacher les employés afin de leur donner les connaissances nécessaire pour améliorer la qualité de leur code base tout en continuant de livrer de nouvelles fonctionalitées.

En guise d'exemple, vous vous attaquez au module de paie (`payroll`) qui exemple plusieurs `code smells` et mauvais designs. Les employés de la start-up vous explique que le module de paie permet de gérer les employés, la création chèques de paie, la gestion des vacances (seulement pour la version longue de l'exercice) et quelques statistiques sur la paie.

3 Tâche

On désire ajouter un nouveau type d'employé au module de paie, un employé à contrat qui n'est payé qu'à la fin de certains `milestones` du projet. Les employés à contrat ne peuvent pas prendre de jours de vacances (seulement pour la version longue de l'exercice).

Toutefois, étant donné la pauvre qualité, cet ajout se révèle être tout un défi. Le code viole les principles de Clean Code et présente de mauvais designs. Vous relevé notamment que le code est exceptionnellement impropre, que la classe `CompanyPayroll` ressemble à une `god class` avec toute la logique centralisée dans cette classe.

Vous procédez en trois étapes pour ajouter la nouvelle fonctionnalité. Premièrement, vous nettoyez le code pour qu'il respecte davantage les principles du Clean Code. Ensuite, vous procédez à un réusinage afin de faciliter l'ajout de la fonctionnalité. Et finalement, vous ajoutez le nouveau type d'employé.

Vous pouvez modifier le code à votre guise, tant que vous réparez les tests de façon à ce que le code compile après chaque étape.

3.1 Clean Code

Profitez de cette phase pour vous familiariser avec la base de code en le nettoyant. Nous vous recommandons aussi de vous familiarisez avec les outils de réusinage présents dans tout bon environnement de développement.

3.2 Refactoring

Maintenant que vous avez pris vos marques dans la base de code, il est temps de le réusiner afin de faciliter l'ajout du nouveau type d'employé à contrat.

3.3 Ajout de fonctionalité

Maintenant que le code est propre, il devrait être facile d'ajouter le nouveau type d'employé. Le nouveau type d'employé est un employé à contrat qui n'est payé qu'à la complétion de certains `milestones` du projet. Pour simplifier la logique, on considère qu'un nouveau `milestone` est complété à chaque paie. Ainsi, on doit pouvoir créer un employé à contrat en spécifiant une liste de montants correspondant à la paie reçu lors de leur complétion. Ainsi, si on crée un employée avec les montants `[100, 200, 300]`, alors l'employé sera payé 100$ à la première paie, 200$ à la suivante, puis 300$ à la dernière paie. L'employé ne peut ensuite plus recevoir de paie.

Ces employés peuvent recevoir des augmentations de salaire qui sont ajouté au montant du `milestone` courant. Un employé à contrat ne peut pas recevoir d'augmentation après la fin de leur contrat, l'opération ne doit alors rien faire. Les employés à contrat ne peuvent pas prendre de journées de vacances (seulement pour la version longue de l'exercice).

4 Solution

La solution de cet exercice se trouve sur glo2003/Exercice-CleanCode-Refactoring. Il y a une branche par étape de la solution, soit *clean* pour le code après le nettoyage, *refactoring* après le réusinage et *new-feature* après l'ajout de la fonctionalité. Le fichier *CLEAN.md* résume les changements apportés lors de la phase de Clean Code et *REFACTORING.md* ceux de la phase de réusinage.

Références

[1] Robert C. Martin. *Clean Code : A Handbook of Agile Software Craftsmanship*. Prentice Hall PTR, USA, 1 edition, 2008. ISBN 0132350882.