DUT-Info/S3/M3105 (CPOA) : Projet refactoring

Table of Contents

Contexte

Vous êtes chargé de prendre la suite d'un développement pour laquelle la documentation est minimaliste et dont les sources sont disponibles ici :

https://github.com/codurance/task-list/tree/master/java

1. Récupérez les sources du projet :
   - soit en clonant le dépôt :

     ```
     git clone https://github.com/codurance/task-list.git
     ```

   - soit en téléchargeant directement le fichier .zip.

Tâches initiales :

- ☐ Remplacer ce `README.adoc` pour qu'il constitue votre rapport.

- [ ] Ajoutez le nom des membres du groupe, ce que vous avez modifié et comment on installe votre application (e.g., `mvn install` ou `gradle install`)

Refactoring

Vous avez jusqu'à la fin de la semaine 2 pour améliorer le plus possible cette application en (cf. détails juste après cette liste) :

- y ajoutant de nouvelles fonctionnalités au passage (cf. ci-dessous).
- s'attaquant aux types primitifs massivement utilisés
- mettant en oeuvre les bonnes pratiques objet

TIP	Commencez simplement, et documentez au fur et à mesure votre README.

TIP	Pour vous aider, anticipez l'évolution du code en regardant dans le dépôt d'origine les fonctionnalités supplémentaires envisagées. Vous pourrez en implémenter certaines pour démontrer la pertinence de vos modifications

Types primitifs

Si vous avez observé attentivement le code, vous vous êtes rendu compte qu'il utilise assez peu d'objets et par contre beaucoup de types primitifs (entiers, char, strings, collections, etc.). Les concepts métiers sont faiblement présent (`task`, `project`, etc.).

Essayez, en implémentant de nouvelles fonctionnalités par exemple, de vous débarrasser le plus possible des types primitifs.

TIP	Un bon exemple pour voir si vous vous focalisez sur les concepts métiers est de vérifier que les types primitifs sont uniquement employés dans les paramètres des constructeurs, les variables locales ou les attributs privés. Ils ne devraient jamais être retournés ou être passés en paramètres à vos méthodes (exception faite des lectures clavier, etc.).

À propos des tests

Vous remarquerez que le seul test fournit est un test fonctionnel. Il pourra vous être utile pour vérifier que votre application respecte toujours bien le cahier des charges initial. Vous pourrez même l'enrichir pour tester que vos nouvelles commandes (du type `view by …`) fonctionnent.

Par contre il ne vous dispense pas du tout des tests unitaires classiques pour les classes que vous allez produire.

Attendus du projet

GitHub repo

Utilisez ce repo pour votre code et votre rapport code. Utilisez des branches si vous pensez que c'est pertinent.

Les éléments obligatoires :

- ▢ vos noms
- ▢ une table des matières
- ▢ une courte explication par chaque fonctionnalité nouvelle ou refactoring précis avec des extraits de code illustratifs et une justification par exemple

Modèles UML

Aucun modèle n'est demandé, mais n'hésitez pas à en utiliser si vous trouvez que c'est pertinent.

Critères d'évaluation

NOTE	Vous pouvez travailler en groupe (2 max.).

- pertinence des choix
- pertinence des tests
- qualité globale du code
- qualité de la documentation (illustration, explications)
- nombre et difficulté des éléments de refactoring