UQÀM 2023

Rapport TP2

Boutique en Java

Équipe: Cypress

Alireza Nezami

nezami.alireza@courrier.uqam.ca

MGL7460 Réalisation et maintenance du logiciel
Professeure : Monsieur Hafedh Mili

Contenu

Introduction :

Dans ce projet, j'ai pu mettre en pratique les concepts académiques présentés dans ce cours et j'ai appris beaucoup de choses en surmontant des défis techniques. Des concepts tels que : les contrôles de version (que nous avons faites dans *GitLab*), les tests unitaires, le pipeline dans *GitLab* et Sonar, qui sont tous des concepts et outils importants pour le développement de logiciels.

À la suit, j'expliquerai certaines des choses que j'ai faites et comprises dans ce projet:

1.Record (Java 16+):

La classe Adresse est définie comme un record. Les records, introduits dans Java 16, sont utilisés pour créer des classes de données simples et immuables. (J'ai rencontré d'une erreur à propos de Record lorsque j'ai téléchargé le projet pour la première fois sur le système local et j'ai réalisé que je devais mettre à jour la version Java.)

2. Patron *Singleton*:

Dans la classe Boutique, l'utilisation de la méthode *getBoutique* () qui retourne une instance de *BoutiqueImpl* (implémentation), indique l'utilisation du modèle Singleton. Ce modèle garantit qu'une seule instance de *BoutiqueImpl* existe dans toute l'application.

3. Patron *Factory*:

L'interface Boutique possède la méthode *getFabriqueBoutique*() qui retourne une instance de *FabriqueBoutique*. Cela peut indiquer l'utilisation du modèle *Factory*, utilisé pour créer des objets sans spécifier explicitement la classe de l'objet à créer.

4. Séparation Interface et Implémentation:

L'utilisation d'interfaces telles que Boutique, Client, Commande, etc., et la définition de méthodes à implémenter dans des implémentations spécifiques de ces interfaces, montre une séparation entre l'interface et l'implémentation. Cela aide à rendre le code plus flexible et évolutif.

5. Énumération:

L'utilisation d'*enum* comme Condition, Province, et Salutation montre l'utilisation de valeurs constantes et limitées, utilisées pour représenter un ensemble de valeurs constantes.

6. Gestion des Exceptions:

La classe *InventaireEpuise* est une exception personnalisée utilisée pour gérer des situations spécifiques dans le projet (comme l'épuisement des stocks). Cela indique l'utilisation de la gestion des exceptions en Java.

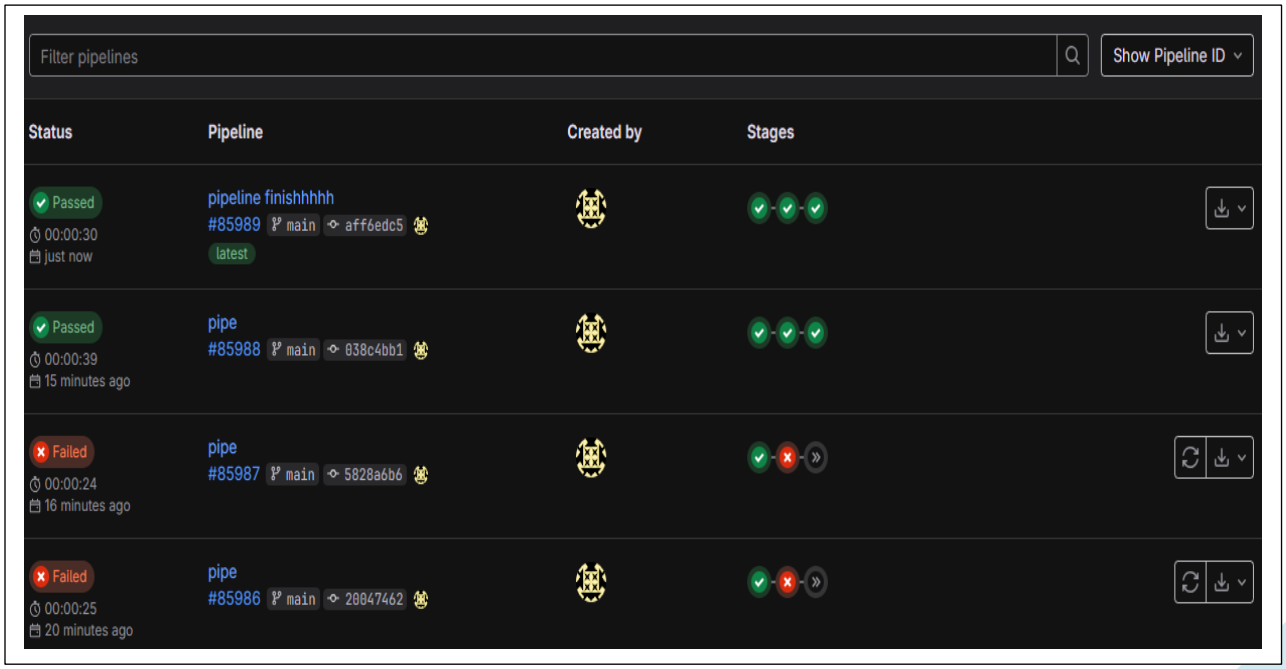
Ces éléments montrent une conception réfléchie et structurée de notre projet, utilisant des modèles de conception courants en Java pour améliorer la lisibilité, la maintenabilité et l'évolutivité du code.

Stratégie de tests unitaires :

1. J'ai implémenté les classes Client, Produit, *LigneCommande* et *ItemInventaire* avec toutes les méthodes de ces classes.
2. J'ai essayé d'exécuter différents tests et en plus de tester avec des valeurs logiques, j'ai également testé avec d'autres valeurs pour m'assurer que les méthodes fonctionnent correctement.
3. J'ai créé un dossier pour les tests afin que tous les tests se trouvent au même endroit. Pour améliorer la qualité de mon code de test, j'ai utilisé @BeforeEach la méthode *setUp*() pour :1) éviter de répéter le code 2) augmenter la lisibilité du code. De plus, pour éviter une mauvaise utilisation des données, j'ai utilisé @AfterEach la méthode *tearDown*() afin que leur valeur soit *Null* après l'exécution de chaque test de méthode.
4. J'ai nommé chaque test avec le nom de la classe à tester. Pour tester chaque méthode, j'ai d'abord écrit le mot-clé test, puis j'ai écrit le nom de la méthode. Donc, à la fois la méthode de test peut être identifiée et il est clair quelle méthode ce test est censé cibler.

Pipeline:

Après plusieurs tentatives de mise en œuvre du pipeline, celui-ci s'est finalement concrétisé et le TP a passé les tests avec succès.



Qualité du code :
Dans ce cours, j'ai fait connaissance avec Sonar, qui dispose d'un outil appelé *sonarLint*, qui permet d'éviter le *smell* code et les problèmes causés par la qualité du code et la maintenance du code lors de l'écriture du code. Cet outil est très utile, car au début nous remarquons les problèmes et nous les résolvons au début.