TP TQL - M1 ILSI

Développement Piloté par les Tests (TDD)

Description du TP

Une bibliothèque souhaite numériser son catalogue de livres. Il est demandé de de développer un système de gestion de bibliothèque en appliquant la méthodologie TDD et en utilisant JUnit 5 pour garantir la fiabilité du code à travers des tests unitaires.

Travail à réaliser

1. Créez un projet Maven et Ajoutez la dépendance JUnit 5 au fichier pom.xml.
2. **Développement de la classe Livre :**
   - Définir les attributs : titre, auteur, isbn et implémenter le constructeur et les getters.
   - Redéfinir les méthodes equals() et hashCode() pour comparer les livres par leur ISBN.
3. **Écriture des tests unitaires pour Livre :**
   - Vérifier la création correcte d'un livre et tester l'égalité entre deux livres ayant le même ISBN.
4. **Développement de la classe Bibliotheque :**
   - ajouterLivre(Livre livre) : void / supprimerLivre(String isbn) : void
   - chercherLivre(String isbn) : Livre / listerLivres() : List<Livre>
5. **Écriture des tests unitaires pour Bibliotheque :**
   - Vérifier l'ajout et la suppression d'un livre.
   - Tester la recherche d'un livre existant et inexistant.
6. **Utilisation des tests imbriqués (@Nested) :**
   - Structurer les tests en groupes logiques pour améliorer la lisibilité et l'organisation des cas de test.
   - Expliquer pourquoi cette approche est utile le test.
7. **Utilisation des tags et des tests répétés :**
   - Ajouter des tags JUnit 5 (@Tag) pour catégoriser les tests.
   - Utiliser les tests répétés (@RepeatedTest) pour exécuter plusieurs fois un test afin de vérifier la robustesse du code.
8. **Simulation avec Mockito :**
   - Implémenter une classe BibliothequeService qui utilise Bibliotheque.
   - Simuler le comportement de Bibliotheque à l'aide de Mockito pour tester BibliothequeService.
9. **Analyse de la couverture des tests avec JaCoCo (ou un outil équivalent) :**
   - Générer un rapport de couverture des tests et assurer une couverture d'au moins 80%.

Bonus

- Utiliser des tests paramétrés (@ParameterizedTest) pour vérifier différents cas de création et de comparaison.
- Utiliser des tests dynamiques (@TestFactory) pour générer automatiquement des cas de test en fonction d'une liste de livres.