TD1 – Test d’une application de gestion de commandes

Contexte

Bienvenue dans ce premier TD de plan de tests d’une application, où vous allez découvrir les tests unitaires en Java à travers un projet de gestion de commandes qui simule une expérience d’achat en ligne et se compose de fonctionnalités essentielles telles que la gestion de produits, le panier d’achat, le traitement des commandes et la génération de factures.

Le code source du projet vous est fourni. Vous serez amenés à :

- **Analyser le projet et rédiger un plan de test** pour définir les critères d'acceptation et les cas de test.
- **Implémenter les tests unitaires** en Java avec JUnit pour vérifier le bon fonctionnement du projet.
- **Identifier et corriger les incohérences** présentes dans le code source.
- **Rédiger un bilan de tests** pour évaluer les résultats et la couverture du code.

Objectifs

À la fin de ce TD, vous aurez :

- Rédigé un plan de test pour identifier les éléments clés à tester.
- Implémenté des tests unitaires en Java avec JUnit.
- Rédigé un bilan de tests incluant les résultats d'exécution et de couverture de code avec JaCoCo.

1. Plan de test

Avant de passer à l'implémentation, commencez par une **analyse approfondie du code fourni** pour bien comprendre les fonctionnalités. Ensuite, rédigez un **plan de test** qui détaillera les critères d'acceptation et les cas de test pour chaque fonctionnalité.

Instructions

Pour chaque fonctionnalité du projet, identifiez les **conditions de test** et **critères d'acceptation**. Cela permettra de définir les **cas de test** nécessaires pour valider le bon fonctionnement du projet. Ce plan de test servira de référence lors de l'implémentation des tests unitaires.

Modèle pour le plan de test :

Fonctionnalité	Critères d'acceptation	Conditions de test	Cas de test
Ajouter un produit au panier	Le produit est ajouté au panier et le total est mis à jour correctement.	- Produit en stock - Produit hors stock	1. Ajouter un produit disponible au panier et vérifier la mise à jour du total. 2. Ajouter un produit hors stock et vérifier que l'erreur s'affiche.
Appliquer un code promo	Le code promo valide réduit le total du montant spécifié, sinon, aucun changement.	- Code promo valide - Code promo invalide	1. Appliquer un code promo valide (ex : PROMO10) et vérifier que la réduction est correcte. 2. Appliquer un code promo invalide et vérifier l'absence de réduction.
…	…	…	…

2. Implémentation des tests unitaires avec JUnit

Après avoir complété le plan de test, implémentez les cas de test en utilisant **JUnit 5**. Assurez-vous que les tests sont organisés par fonctionnalité et qu'ils couvrent les critères d'acceptation et les conditions de test définis dans le plan de test.

Instructions

1. **Créer les tests** pour chaque méthode principale, en vous basant sur vos cas de test.
2. **Exécuter les tests** et vérifier leur validité. Notez les échecs pour analyse.
3. **Corriger les erreurs** dans le code source si nécessaire afin de faire passer les tests.
4. Validez chaque correction par une nouvelle exécution des tests.

Exemple de méthode de test :

```java
@Test
void testAddition() {
    Calculator calculator = new Calculator();
    int result = calculator.add(5, 3);
    assertEquals(8, result, "L'addition de 5 et 3 devrait donner 8");
}
```

3. Bilan de tests

Après avoir exécuté les tests et corrigé les erreurs, rédigez un **bilan de tests** qui synthétise les résultats obtenus. Ce document devra inclure :

- **Statistiques de couverture** : Indiquez la couverture de code atteinte (JaCoCo), le nombre total de tests, le nombre de tests réussis et échoués.
- **Analyse des anomalies** : Décrivez les anomalies trouvées dans le code, les corrections appliquées et les validations effectuées.

Modèle pour un rapport d'anomalie (à inclure dans le bilan de tests) :

ID	Description	Étapes pour reproduire	Résultat attendu	Résultat obtenu	Gravité	Solution apportée
ANOM-001	…	…	…	…	…	…

4. Consignes de remise

Afin d'évaluer votre travail, merci de déposer les livrables listés ci-dessous sur Campis Connect. Assurez-vous de bien les organiser et de nommer les fichiers de manière explicite.

- **Plan de test** dans un fichier PDF.
- **Bilan de tests** dans un fichier PDF.
- **Code source** : l'ensemble du code source du projet dans une archive ZIP, incluant les classes de l'application ainsi que les tests unitaires que vous avez réalisés.