Corrigez un test unitaire pour vérifier le code — Webinar 24/08/21

Description du webinaire : Vous utilisez vos compétences en test logiciel pour aider une banque. D'abord, vous corrigerez un test unitaire mal écrit, afin que ce test puisse vérifier le logiciel des distributeurs de billets. Ensuite, vous exécuterez le test corrigé sur un échantillon de code.

🎯**Compétence visée :** Mettre en œuvre des tests unitaires

Projets et paths concernés : P4 Java = P5 Salesforce : Testez l'implémentation d'une nouvelle fonctionnalité Java

Exercice 1

Scénario

Vous êtes développeur freelance depuis plusieurs mois, et vous êtes toujours prêt à aider des entreprises qui en ont besoin.

Cette semaine, le groupe OC Bank vous a contacté avec une offre de travail freelance. Il développe un nouveau logiciel pour ses systèmes de distributeurs de billets. Ce logiciel améliorera les systèmes existants, et donnera à OC Bank un avantage sur ses concurrents du secteur bancaire.

Malheureusement, un de ses développeurs est tombé malade après avoir développé la V1 du projet et deux tests unitaires. Cette V1 doit être testée, mais il n'a pas encore développé la suite de tests complets pour la vérifier. Sans les tests, OC Bank ne peut pas être sûr de son travail, ni faire évoluer l'application.

OC Bank vous a demandé de remplacer son développeur malade. Vous allez devoir repasser sur le code pour corriger les petites erreurs qui se sont glissées dedans (voir les TODO), et corriger les tests unitaires déjà écrits.

C'est parti !

README :

L'application est une spring boot-app fonctionnelle ; pour la faire tourner, il suffit de faire un mvn install et de la run via votre IDE.

Elle est exposée sur le port 8081.

Pour visionner son fonctionnement, un jeu de query est fourni via Postman.
La base de données est une base de données embarquée en H2 qui est recréée à chaque lancement de l'application, ce qui vous permettra de modifier les données lors de vos lancements de l'app.

Livrable

Les tests unitaires corrigés.

Corrigé

Un exemple corrigé sera disponible sur le lien GitHub au jour du webinaire.

Exercice 2

Scénario

La 1re étape étant validée par OC Bank, il vous est maintenant demandé de continuer à augmenter la couverture de code jusqu'à 70 %, uniquement sur les tests unitaires.

Pour cela, vous allez devoir ajouter des tests unitaires. Lorsque ce sera fait, il faudra générer le rapport JaCoCo pour prouver que toutes les méthodes ont bien été créées.

Livrable

Le rapport de test généré via JaCoCo avec une couverture sur les tests unitaires de 70 %. Il sera généré en utilisant Maven Site.

Corrigé

Un exemple corrigé sera disponible sur le lien GitHub au jour du webinaire.

Pour aller plus loin

- Une introduction rapide aux tests : https://medium.com/@kentbeck_7670/programmer-test-principles-d01c064d7934
- Utilisation de JUnit : https://www.vogella.com/tutorials/JUnit/article.html
- La bible du dev Java en français : https://www.jmdoudoux.fr/java/dej/chap-junit.htm
- Le meilleur livre sur les tests : http://barbra-coco.dyndns.org/yuri/Kent_Beck_TDD.pdf
- Configuration de JaCoCo : https://www.baeldung.com/jacoco
- Utilisation de Postman pour l'API Query : https://learning.postman.com/docs/getting-started/introduction/