TP - TDD

OUTILS REQUIS
- Java JDK 17 ou supérieur
- Un IDE IntelliJ
- Maven 3
- Git
- Un compte Github

Contexte

Vous travaillez pour l'Agence Nationale des Titres Sécurisés (ANTS) dans l'équipe en charge de développer la nouvelle API Java de gestion des permis de conduire. Votre Product Owner (PO) vous a transmit les différents besoin auxquels vous allez devoir répondre, ainsi qu'une proposition de conception réalisée par votre Teach Lead (TL).

Votre mission est de développer l'API demandée par votre PO en suivant la conception réalisée par votre TL. De plus, vous avez récemment entendu parler de Test Driven Development (TDD) et de pair programming, et en avez parler avec le TL de votre équipe. Ce dernier trouve que c'est une très bonne idée et vous missionne donc de former un groupe avec l'un de vos collègue pour réaliser la nouvelle application en TDD et en pair programming.

Le besoin

En tant que développeur de l'ANTS qui travail sur le CRM de gestion des permis de conduire, je souhaite avoir accès à une API développée en Java qui me permette de sauvegarder en base de données des permis de conduire, mais également de rechercher un permis de conduire par son identifiant. Enfin, je souhaite pouvoir retirer des points à un permis de conduire présent en base de données. Votre PO vous informe que la demande est très urgente et que vous disposez d'un délai d'une heure pour la réalisation de cette API. Votre TL ayant déjà réalisé la conception mets également à votre disposition un squelette d'application à retrouver ici : https://github.com/Bad-Pop/tp-tdd

Sauvegarder un permis de conduire

Vous devez développer un service prenant en entrée un numéro de sécurité sociale, qui valide ce numéro de sécurité sociale, créé un nouveau de permis de conduire avec ce même numéro validé et enfin sauvegarde le permis dans la base de données avant de retourner le nouveau permis.

RÈGLES MÉTIER
1. Vérification de la validité d'un numéro de sécurité sociale
    1. Ne doit pas être nul
    2. Ne doit contenir que des chiffres
    3. Doit avoir une longueur de 15
    4. Si le numéro de sécurité sociale n'est pas valide lever une exception de type **InvalidDriverSocialSecurityNumberException**
2. Création du permis de conduire
    1. L'identifiant du permis de conduire doit être généré grâce à la classe **DrivingLicenceIdGenerationService**
    2. Lorsqu'un permis de conduire est créé, il possède un total de 12 points
    3. Le numéro de sécurité sociale validé doit être renseigné à la création permis de conduire
    4. Sauvegarder le nouveau permis de conduire
    5. Retourner le nouveau permis de conduire

Rechercher un permis de conduire par son ID

Vous devez développer un service permettant de recherche un permis de conduire dans la base données par son id.

RÈGLES MÉTIER

1. Rechercher en base de données le permis de conduire par son id
2. Retourner un objet de type Optional pour gérer le cas où aucun permis n'existe avec cet id

Retirer des points sur un permis de conduire

Vous devez développer un service permettant de retirer des points à un permis de conduire.

RÈGLES MÉTIER

1. Rechercher en base de données le permis par son id
    1. Si aucun permis, lever un exception de type **ResourceNotFoundException**
    2. Si permis trouvé
        1. Retirer le nombre de points demandés au permis
            1. Le nombre de points d'un permis de conduire ne peut pas être inférieur à 0
        2. Mettre à jour le permis de conduire en base de donnée
        3. Retourner le permis de conduire avec le nouveau solde de points

Note aux étudiants : afin de gagner du temps, la base de données est simulée par la classe **InMemoryDatabase** présente dans le package **fr.esgi.cleancode.database**. Cette classe propose deux fonctions :

- save() : pour insérer ou mettre à jour un permis de conduire
- findById(UUID id) : pour chercher un permis de conduire par son id

Critères d'évaluation

À l'issu du temps impartit, vous présenterez au formateur le code réalis

Vous devez appliquer le TDD lors de ce TP, mais également toutes les notions abordées pendant le cours que vous jugerez utiles, mais vous devrez expliquer pourquoi. Voici le barème de l'évaluation :

- **Respect du besoin (5 points) :** le code réalisé répond au besoin en respectant les règles métier rédigées par votre PO.
    - *Sauvegarder un permis : 2 points*
    - *Rechercher un permis : 1 point*
    - *Retrait de points : 2 points*
- **Application du TDD (7 points) :** Pour chaque fonctionnalité, vous devrez dans un premier temps écrire les tests unitaires qui valident les règles métiers que vous a fourni votre PO. Une fois les tests rédigés, vous devrez créer un commit git afin de permettre à l'évaluateur de suivre votre cheminement, et pousser le commit sur votre repository git distant hébergé sur GitHub. Vous pourrez ensuite commencer à développer le service dont les tests sont rédigés. Une fois le service implémenté et que les tests sont au vert, faire un nouveau commit et le pousser sur le repository distant. Faire cela pour toutes les fonctionnalités.
- **La couverture de code (2 points) :** le taux de couverture de code de votre API est au moins égal à 85%.
- **Le nommage (2 points) :** vous avez utilisé des nommages qui respectent les règles abordées durant le chapitre dédié au nommage.
- **Respect du principe SRP (2 points) :** votre code respecte le principe de responsabilité unique.
- **Absence d'effets de bord (1 point) :** votre code ne contient pas d'effet de bord
- **La fonctionnalité retrait de point exploite Optional Java de Java (1 point):** le code de cette fonctionnalité exploite pleinement l'API Optional.

À l'issu du temps impartit, les binômes présenterons chacun leur tour le code qu'ils ont réalisés.