Médiathèque : tests unitaires, d'intégration et de validation

Préparation de votre environnement de travail

Dans cette étape, nous vous demandons de partir de l'archive proposée. En effet, le projet dans cette archive intègre des modifications présentées pendant le cours sur les tests.

Lors de cette étape, nous vous conseillons de suivre les diapositives du cours pour vous aider à répondre aux questions. Pensez à consulter les diapositives référencées dans ce sujet de TP.

- Récupérez le fichier Tests.zip dans votre espace de travail.
- Importez comme nouveau projet l'archive que vous venez de sauvegarder. Cela crée le projet Eclipse Test-osssature.
- Ce projet est configuré pour utiliser le canevas logiciel JUnit.
- Vous devez modifier le chemin de configuration du projet pour qu'il utilise le fichier util.jar de votre répertoire.

Canevas logiciel JUnit et contenu de l'archive

Nous vous proposons dans ce TP d'utiliser le canevas logiciel (la librairie) JUnit pour écrire les tests. Ce canevas logiciel utilise les annotations Java. Vous avez eu une introduction au canevas logiciel JUnit en cours.

Exécution des tests existants

Un certain nombre de tests présentés dans le cours ont été implantés. À titre d'exemple, ouvrez la classe JUnit_DocumentTest et parcourez rapidement les méthodes de test existantes. Exécutez les tests unitaires de cette classe dans Eclipse: dans le menu *Run*, choisissez *Run as* puis *JUnit test* ; cette action ouvre une perspective avec une barre verte, rouge ou bleue et des indicateurs montrant si les tests se sont déroulés correctement ou non.

En guise d'essai d'exécution en erreur d'un test, modifiez la méthode testReachableStates en retirant l'instruction d1.metEmpruntable(). Ré-exécutez la classe JUnit_DocumentTest et observez que le test correspondant « ne passe plus ».

Tests unitaires

À partir du cours, mettez à jour le fichier JUnit_DocumentTest.java du paquetage tests en ajoutant les tests qui manquent. Vous devez ajouter les bonnes assertions pour chaque test et spécifier si besoin l'exception que doit générer le scénario du test (par exemple, comme dans l'annotation suivante « @Test(expected=OperationImpossible.class) »). Voici les tests qui manquent :

1. « nombre d'emprunts » : emprunter un document 5 fois de suite et vérifier que son nombre d'emprunts a évolué correctement ;
2. « test 4.1 » : vérifier qu'il n'est pas possible (une exception est levée) de restituer un document qui est empruntable mais non emprunté;
3. « double emprunt »: vérifier qu'il n'est pas possible d'emprunter un document qui est déjà emprunté ;

4. « exception quand constructeur incorrect » : vérifier qu'une exception est levée quand le genre passé en argument est null dans le constructeur de la classe Document.

Exécutez les tests unitaires dans Eclipse: dans le menu *Run*, choisissez *Run as* puis *JUnit test* ; cette action ouvre une perspective avec une barre verte, rouge ou bleue ainsi que des indicateurs montrant si les tests se sont déroulés correctement ou non.

Tests d'intégration avec Junit

Pendant le cours, une séquence a été proposée concernant la coordination par la classe FicheEmprunt de l'interaction « emprunter ». Ce test d'intégration est réalisé dans la classe IntegrationTest du paquetage tests de l'ossature fournie pour ce TP.

En vous inspirant des premiers tests d'intégration de la classe IntegrationTest, complétez la classe en écrivant les tests d'intégration pour l'interaction « restituer » qui testent l'enchaînement correct entre FicheEmprunt, Document et Client. Les deux cas que nous vous proposons de tester sont les suivants :

1. l'emprunt est rendu à temps :
    - création d'un document,
    - mise du document à empruntable,
    - création d'un client,
    - création d'une fiche d'emprunt,
    - restitution du document par l'appel à une méthode de la classe FicheEmprunt et vérification que la restitution est correcte du côté du document (le document peut être emprunté à nouveau) et du côté du client (le nombre d'emprunt en cours a diminué),
2. l'emprunt est rendu en retard :
    - la séquence est similaire à celle du test d'intégration précédent avec un appel à Datutil.addAuJour(...) pour avancer artificiellement l'horloge, un appel à la méthode verifier() de la classe FicheEmprunt pour tester si l'emprunt est en retard, et un appel à la méthode premierRappel() pour marquer le client en retard.

Tests de validation

Pendant le cours, le fichier de tests de validation AcceptanceTest du paquetage src a été construit avec les deux premiers tests de la table de décision du cas d'utilisation « emprunter un document » présentée dans le cours.

1. Écrivez les tests de validation pour le cas d'utilisation « emprunter un document » pour au moins un des éléments de la table de décision (3-7) présentés mais non traités en cours. Commencez par les tests 4, 5 et 6, et terminez par les tests 7 puis 3, ce dernier étant de loin le plus long à réaliser.
2. [Question optionnelle] Vérifier que la suppression d'un genre de la médiathèque ne peut être réalisée que lorsqu'aucun document ne référence ce genre,
    - Écrivez le test de validation correspondant.