TP N° 1
Les outils de test unitaires

Exercice 1. Tests unitaires et évaluation de la qualité des tests

Créez un nouveau projet et intégrez le code source de l'exercice 1 et les librairies correspondantes.

- Ecrire le test unitaire validant l'opération matricielle *determinant* de la classe *MatrixMathematics,* qui calcule le déterminant d'une matrice.
- Calculer la couverture du test.
- Compléter les tests unitaires pour atteindre une couverture de 100% pour la classe *MatrixMathematics.*
- Évaluer la qualité des tests avec Pitest sur la classe *MatrixMathematics.*

Exercice 2. Tests unitaires d'une base de données en mémoire

Créer un nouveau module et intégrez le code source de l'exercice 2 et les librairies correspondantes.

Le code source est composé de deux packages:

- *com.example.model*: contient l'entité *City* qui représente une ville.
- *com.example.service*: contient deux classes techniques: (1) la classe *DatabaseConnection* contient les méthodes de connexion, déconnexion et création de la base de données et (2) la classe *DatabaseService* contient les méthodes *addCity* pour insérer une ville à la base de données et *getCity* pour récupérer une ville par son identifiant.

Le travail demandé consiste à écrire les tests unitaires validant les opérations *addCity* et *getCity* en utilisant une base de données en mémoire.

Les paramètres à utiliser dans la classe *DatabaseConnection* pour accéder à une base de données en mémoire sont les suivants:

- *USER= "sa",*
- *PASS = "",*
- *JDBC_DRIVER = "org.h2.Driver",*
- *DB_URL = "jdbc:h2:mem:test";*

Exercice 3. Les tests doubles avec Mockito

Créez un nouveau module et intégrez le code source de l'exercice 3 et les librairies correspondantes.

Il s'agit d'utiliser les tests doubles pour tester les méthodes métiers d'une simple application de réservation d'hôtel.

Le code source est composé de trois packages:

- *com.example.model*: contient les entités du domaine.
- *com.example.dao*: contient les services d'accès à la base de données.
- *com.example.service:* contient les services métiers.

Utilisez Mockito pour tester la réservation d'une chambre et l'annulation d'une réservation (les méthodes de la classe *BookService)*.

Livrables.

Un fichier .zip contenant les fichiers source *.Java* des tests et les rapports HTML de Pitest du premier exercice.

Exercice 4. Application de l'approche TDD

Utilisez l'approche TDD pour écrire un programme qui affiche des nombres de 1 à 100. Pour les nombres multiples de 3, il affiche le mot "Fizz" au lieu d'afficher le nombre et pour les multiples de 5, il affiche le mot "Buzz". Pour les nombres qui sont multiples de 3 et 5, le programme affiche le mot "FizzBuzz".