R4.02 - Qualité de développement
TP3

Semestre 4
2022/2023

Le code pour ce TP est disponible sur Gitlab :

`https://gitlabinfo.iutmontp.univ-montp2.fr/r402/tp3/`

1 Découvrir JUnit 5

Les classes `JUnitDemoBasics` et `JUnitDemoAdvancedFeatures` donnent des exemples des fonctionnalités de JUnit 5. Lisez ces classes, et déterminez quels tests doivent réussir et quels tests doivent échouer. Exécutez les tests pour vous assurer que vos prédictions sont correctes.

2 Tests d'une application de commerce en ligne

Le but de cet exercice est de développer, à partir de *user stories*, une suite de tests pour l'application de commerce en ligne fournie par le package `ecommerce`. Considérez les deux *stories* ci-dessous et pour chacune d'elles, établissez un ensemble de cas de tests :

1. déterminer les différentes méthodes qui correspondent aux *stories*, ainsi que leurs différentes utilisations
2. pour chaque cas, listez les actions qui seront effectuées, et identifiez celles qui relèvent des valeurs préfixes, des valeurs de tests, des valeurs postfixes de vérification et des valeurs postfixes de sortie ;
3. déterminez la ou les propriétés à vérifier ;
4. implémentez les tests et exécutez-les.

Lors de l'implémentation des tests, il se peut que vous découvriez des problèmes de conception dans l'architecture de l'application. Si c'est le cas :

1. expliquez quel est le problème et comment il affecte la testabilité (observabilité ou contrôlabilité) ;

2. corrigez le problème, et justifiez le fait que votre modification ne contrevient pas à l'encapsulation des données.

User story 1 : En tant que client, je veux ajouter des articles à mon panier afin de pouvoir les commander.

User story 2 : En tant que client, je veux retirer des articles de mon panier afin de corriger des ajouts accidentels.

3 Tests pilotés par les données

Dans cet exercice, nous allons écrire des tests pilotés par les données pour la méthode `Math.abs(int a)` qui retourne la valeur absolue d'un nombre entier.

1. Écrivez une méthode de test paramétrée (`@ParameterizedTest`) avec deux arguments x et y, où x est l'argument passé comme entrée à la méthode testée, et y la sortie attendue.
2. Choisissez un ensemble de valeurs de tests, et écrivez une méthode statique qui génère des couples (x, y) à fournir comme entrées à la méthode ci-dessus. Utilisez l'étiquette `@MethodSource` pour préciser que le test utilise cette source de données, et exécutez le test sur les différentes entrées.

Dans un deuxième temps, nous allons écrire un test paramétré uniquement par un argument x qui correspond à l'entrée de la méthode.

1. Déterminez une propriété qui doit être vérifiée pour toute valeur x. Cette propriété doit être aussi précise que possible pour refléter la spécification de la méthode.
2. Implémentez votre test avec les étiquettes `@ParameterizedTest` et `@MethodSource` pour spécifier les données d'entrée. Exécutez le test.
3. Incluez les valeurs limites du domaine d'entrée (`int`) parmi les valeurs à tester. La méthode satisfait-elle votre assertion dans tous ces cas ? D'où vient le problème et comment le résoudre ?