

TP Qualité Logiciel et Revue de Code

UE Génie Logiciel - Master SNS - Université Rennes 2

March 23, 2026

1 Introduction

Ce TP a pour objectif de vous familiariser avec les outils de qualité logicielle en environnement Java. À travers l'utilisation de JUnit 5, vous apprendrez à concevoir des suites de tests unitaires robustes, à pratiquer la revue de code pour identifier des régressions et à documenter votre travail via la Javadoc, garantissant ainsi la maintenabilité de vos développements.

2 Exercice 1 : Documentation et Qualité

2.1 Javadoc

Dans cette première section, vous devez rédiger la Javadoc complète pour deux méthodes fournies (**add** et **subtract**). Pour rappel, une Javadoc conforme doit contenir :

- Un résumé concis du comportement de la fonction.
- La description de chaque paramètre d'entrée (**@param**).
- La description de la valeur de retour (**@return**) si le type n'est pas **void**.
- Les éventuelles exceptions levées (**@throws**).

2.2 Revue de Code et Non-régression

Le but de cette section est de vous exercer à la revue de code. Quatre fonctions vous sont fournies avec un comportement erroné ou incomplet.

En vous appuyant sur la Javadoc de ces fonctions et sur les rapports d'erreurs générés par l'environnement de test, vous devez modifier le code source pour que 100% des tests passent. Une classe **Main** est à votre disposition pour exécuter manuellement certains scénarios et faciliter votre phase de débogage.

3 Exercice 2 : Couverture de code (Code Coverage)

L'objectif est d'atteindre **100% de couverture de code** (instructions et branches) sur les deux classes fournies en utilisant JUnit 5. Vous devrez vérifier le bon fonctionnement du système, mais aussi la gestion rigoureuse des cas d'erreurs.

Le projet contient deux classes interdépendantes :

- **Client** : Gère l'état d'un membre (nom, abonnement, dette, statut). Elle inclut la logique de facturation individuelle et le paiement à une salle.
- **Gym** : Gère l'établissement (solde, adresse) et une liste de membres. Elle permet de piloter la facturation globale de tous les clients actifs.

3.1 Mise en place

1. Créez un nouveau package dans le dossier `src/test/java` nommé `org.genielogiciel.exo.2`.
2. Créez deux classes de test : `ClientTest` et `GymTest` dans ce package.

3.2 Création des Tests

Pour chacune des classes, rédigez les méthodes de test permettant de couvrir l'intégralité du code source. Vous devrez utiliser les assertions JUnit 5 appropriées :

- `assertEquals(expected, actual)` pour vérifier les valeurs.
- `assertTrue(condition)` / `assertFalse(condition)` pour les états booléens.
- `assertThrows(Exception.class, executable)` pour valider la levée des exceptions.

3.3 Bonus

Pouvez-vous identifier un cas d'usage ou un scénario limite pour lequel le code actuel échouera systématiquement (ex: précision des nombres flottants, gestion de la liste de clients) ? Comment pourriez-vous modifier l'implémentation pour résoudre ce problème ?