TP1 : Fiabilité logicielle

04 janvier 2023

ARTAUD Cyril
DRAOU Meryem

Nous regardons la classe ComplexTest.

On lance d'abord le test testGetterImaginary puis on constate que le test passe. Ensuite on lance tous les tests est on constate que trois des tests ne passent pas : testProductReal, testProductImaginary et testTimeoutInfinite.

Pour que ces trois testes ne nous dérangent pas pour la suite nous mettons @Disabled devant les mots clés @Test de ces tests.

Pour vérifier si les méthodes avec les mots clés @BeforeAll, @AfterAll, @BeforeEach et @AfterEach fonctionne nous ajoutons des instructions System.out.println() dans chacune des méthodes ce qui donnera en sortie :

beforeAll
beforeEach
afterEach
beforeEach
afterEach
beforeEach
afterEach
beforeEach
afterEach
beforeEach
afterEach
beforeEach
afterEach
beforeEach
afterEach
beforeEach
afterEach
beforeEach
afterEach
afterAll

Ensuite on crée des attributs z, z1 et z2 dans la classe ComplexTest et dans la méthode setUp(), correspondant au @BeforeEach, on rajoute ces lignes pour mutualiser la création d'objet :

z = new Complex();
z1 = new Complex(1.0F, 2.0F);
z2 = new Complex(3.0F, 4.0F);

et nous supprimons ces lignes dans les autre méthodes de tests si elles sont identiques, sinon on change un des attributs à la valeur qui lui est nécessaire.

Après d'avoir lancé les tests avec gradlle on ouvre le fichier index.html qui se situe dans /build/reports/tests/test du projet et nous pouvons y voir qu'il y a 11 tests, qu'il y a eu 0 échecs et que 100 % sont réussis, qu'il y en a 3 qui ont été ignorés et ils ont été exécutés en 0.371s.

Nous avons pour nos tests :
z1 = new Complex(1.0F, 2.0F);

On ajoute trois méthodes pour tester la méthode inverse de la classe Complex :

- la première méthode testInverseReal() teste que la partie réelle du résultat soit correcte dans le cas d'un complexe non nul, la valeur attendue est bien 1 / 5. Le calcul effectué étant 1*1 + 2*2 = 5, et 1 étant la partie réelle de z1, on a donc bien 1 / 5 pour la partie réelle.

- la seconde méthode testInverseImaginary() teste que la partie imaginaire du résultat soit correcte dans le cas d'un complexe non nul, la valeur attendue est bien -2 / 5. Le calcul effectué étant 1*1 + 2*2 = 5, et -2 étant la partie réelle de z1, on a donc bien -2 / 5 pour la partie réelle.

- la troisième méthode testInverseZero() teste dans la cas contraire la levée d'une exception, à l'aide de assertThrowBy(), et un élément potentiellement hors limite soulève une IllegalArgumentException.

On constate alors que les tests passent.

Ensuite on complète la méthode product() de la classe Complex pour réaliser le produit d'un nombre complexe, et nous mettons en commentaire pour pouvoir tester, des @Disabled devant les mots clés @Test des 2 tests testProductReal() et testProductImaginary(). On constate que les tests passent.

Puis, nous complétons le code de la méthode static infinite afin que l'exécution de cette méthode ne termine pas puis activer le test sur infinite.

On modifie le test pour qu'il échoue si infinite ne termine pas en 100ms (utiliser l'assertion assertTimeoutPreemptively (voir la partie Timeout de la documentation). Vérifier que l'exécution des tests termine.

On modifie le test pour qu'il échoue si infinite ne termine pas en 100ms en utilisant l'assertion assertTimeoutPreemptively(). On constate que l'éxécution termine.