



Réalisation d'un tutoriel JUnit5

Présenté par : Jaza Bacem



Plan

1. Introduction : C'est quoi JUNIT?
 - a. Définition
 - b. Junit LifeCycle
 - c. Unicité de Junit5
2. Installation et configuration
3. Exemple : Traitement des quadrilatères
 - a. Présentation d'exemple
 - b. Les cas de tests
 - c. Exécution des tests

1. Introduction

a. Définition



Contrairement aux versions précédentes de JUnit, JUnit 5 est composé de plusieurs modules différents issus de trois sous-projets différents.

JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage.

La plate-forme JUnit sert de base pour lancer des frameworks de test sur la JVM.

Il procure un simple API pour tester l'application JAVA. La classe ou la méthode à tester est appelée : "Subject under test".

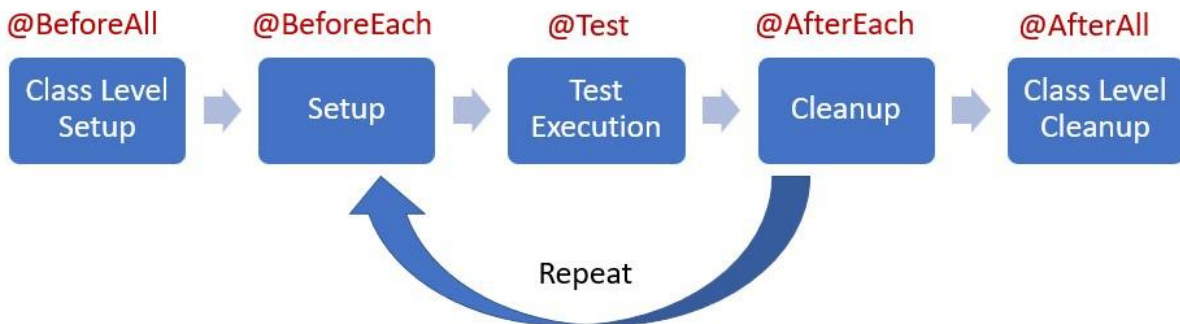
Donc pour faire un test, il faut créer une classe de test java et utiliser l'API de JUnit pour faire des scénarios de test et puis JUnit prend la responsabilité d'afficher les erreurs et les échecs de classe.

Un support de première classe pour la plate-forme JUnit existe également dans les IDE populaires (voir IntelliJ IDEA , Eclipse , NetBeans et Visual Studio Code) et les outils de construction (voir Gradle , Maven et fourmi).

Pour ce projet, on va s'intéresser en Visual Studio Code et MAven:



b. Junit Lifecycle



c. Unicité de Junit5

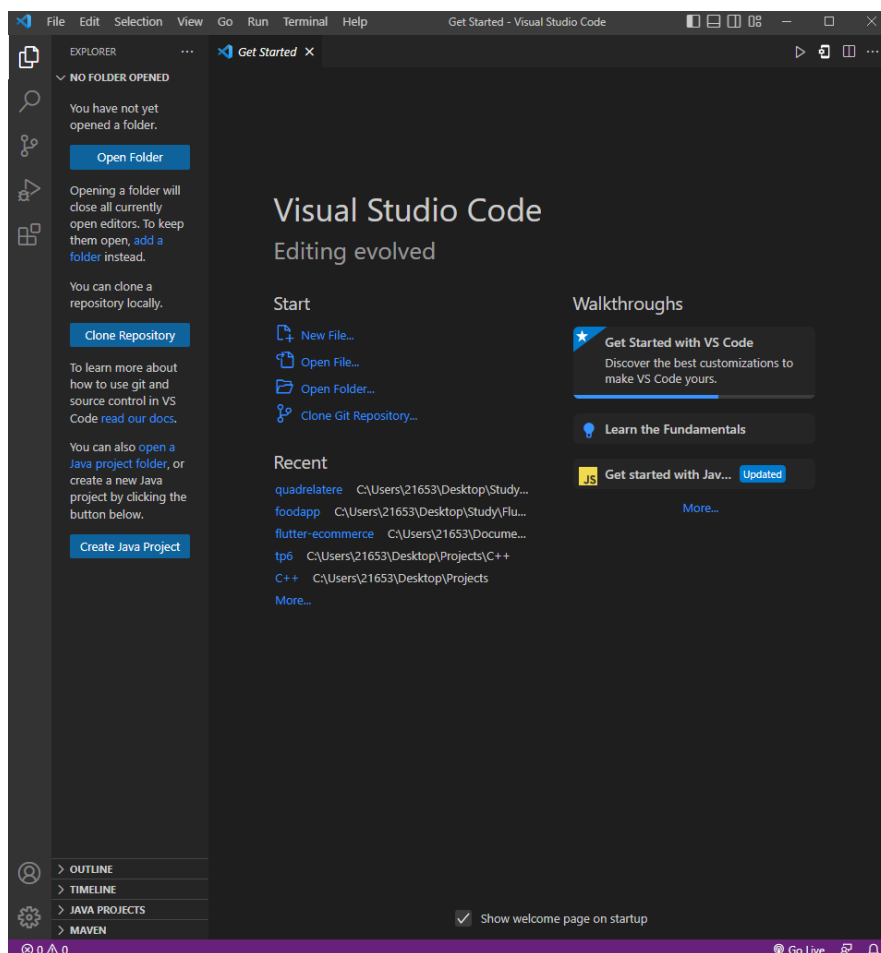
- le support et l'utilisation des nouvelles fonctionnalités de Java 8 : par exemple, les lambdas peuvent être utilisés dans les assertions
- une nouvelle architecture reposant sur plusieurs modules
- le support de différents types de tests
- un mécanisme d'extension qui permet l'ouverture vers des outils tiers ou des API

2. Installation et configuration

a. Vérification des installations

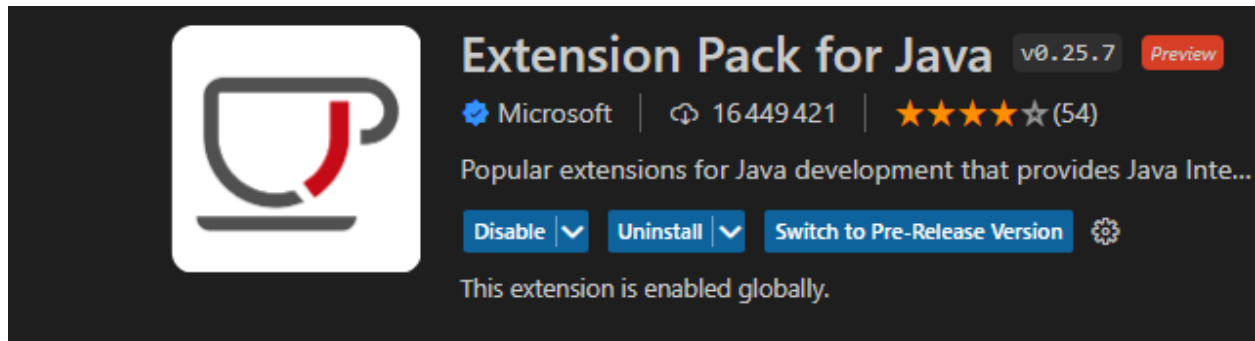
On doit vérifier ici les installation de JDK (Java Development Kit) et VSCode (Visual studio code) comme suit :

```
C:\Users\21653>java --version
openjdk 17.0.5 2022-10-18
OpenJDK Runtime Environment Temurin-17.0.5+8 (build 17.0.5+8)
OpenJDK 64-Bit Server VM Temurin-17.0.5+8 (build 17.0.5+8, mixed mode, sharing)
```



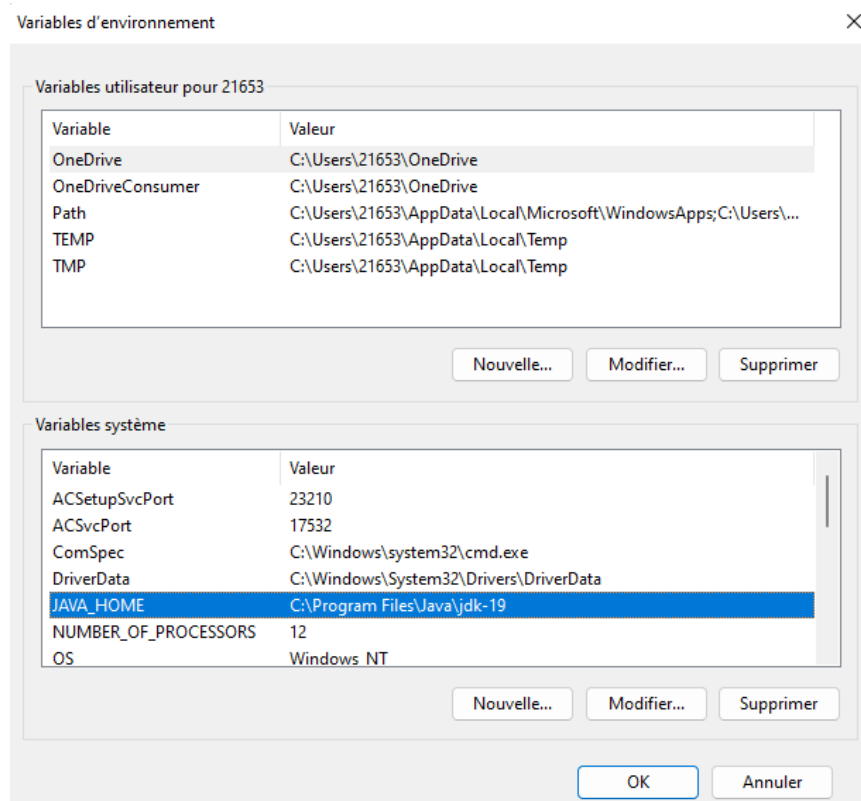
b. Java extension pack

On doit installer Java extension pack qui est une extension Java situé dans les packages de VSCode pour faciliter la gestion de java en VSCode.



c. JAVA_HOME

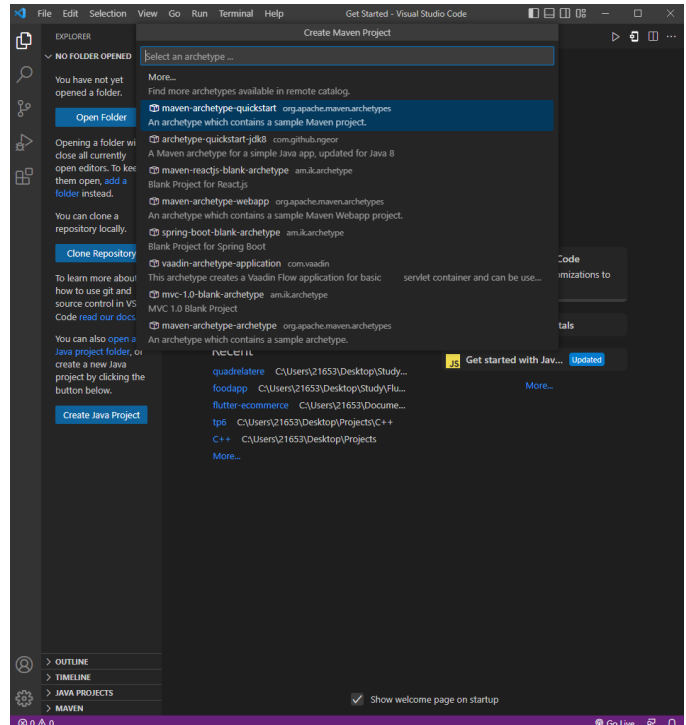
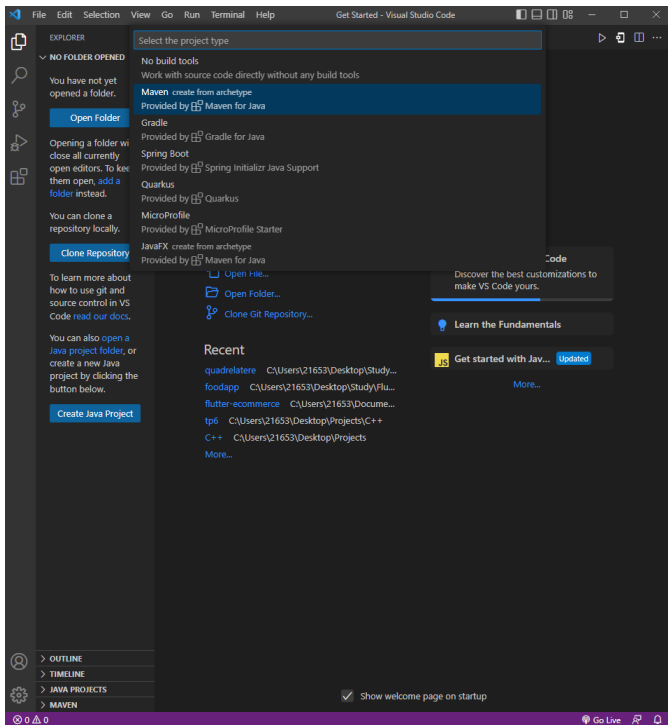
Il faut ajouter le variable d'environnement JAVA_HOME qui porte le path du JDK installé comme suit:



d. *Projet Java avec Maven*

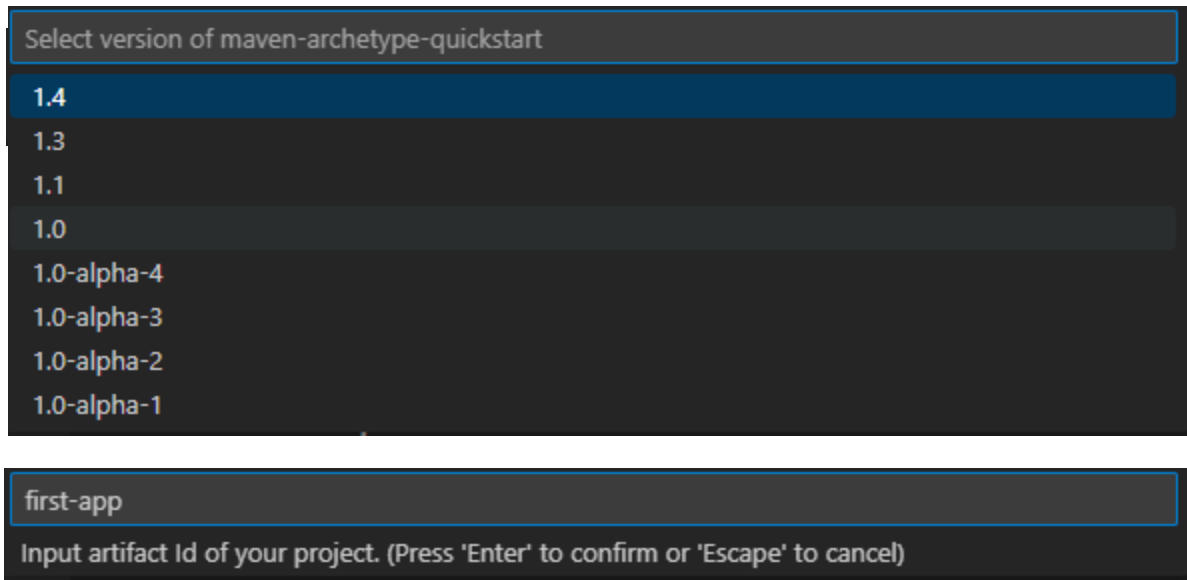
On va créer ici un projet Java en utilisant Maven Building Tool.

On appuie tout d'abord au bouton “Create Java Project” situé dans la page vierge de VSCode puis on appuie sur “maven-archetype-quickstart” et on choisit la dernière version.

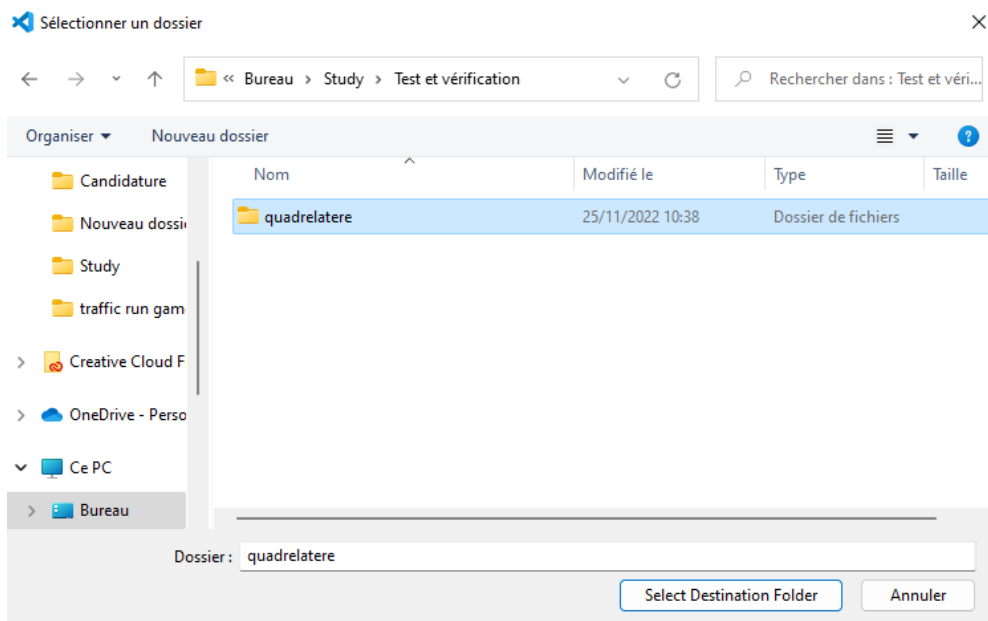


8

Le groupId peut être votre nom, la dernière version de maven la recommandé et le demo est le nom de votre application.



On doit maintenant créer le dossier qui va contenir le projet et le sélectionner.



3. Exemple : Traitement des quadrilatères

a. Présentation d'exemple

Les quadrilatères sont des polygones formés par une ligne brisée fermée ayant 4 côtés.

Parmi les quadrilatères qu'on peut trouver et qui sont étudiés dans notre cas :
parallélogramme, carré, rectangle, losange, trapèze.

Le programme sert à entrer les deux côtés consécutifs (AB et AC) du quadrilatère avec l'angle (ABC) et l'angle consécutif (CAD) pour former le quadrilatère (ABCD) et sortir une graphe qui le dessine en produisant son type, son périmètre et son aire.

b. Les cas de tests

- Côtés de valeurs négatives ou null => test invalide.
- Angles de valeurs négatifs.
- Les données forment un triangle => test invalide (Côtés égaux et angles égaux à 60).
- Angles nuls => test invalide.
- Angles plats => test invalide (ligne).
- Angles > 180 => test valide.
- Cas particuliers :
 - Carré.
 - Parallélogramme.
 - Rectangle.
 - Losange.
 - Trapèze.

- Trapèze isocèle.

c. Exécution des tests:

On a utilisé ici plusieurs propriétés de Junit5 pour tester le programme :

- Assertions (assertEquals, assertTrue, assertFalse).
- Assumptions (assumeTrue, assumeFalse).
- RepeatedTest. @RepeatedTest

D'autres propriétés de Junit5 :

- Parametrized tests (@methodSource): @ValueSource, @CsvSource.
- Nested tests : Organiser les tests.
- Disabled tests.

```
@BeforeAll
public static void setupAll(){
    f = new JFrame();
}
```

```
@BeforeEach
public void beginTest(){
    System.out.println("Begin Test"+n+"\n\n");
}
```

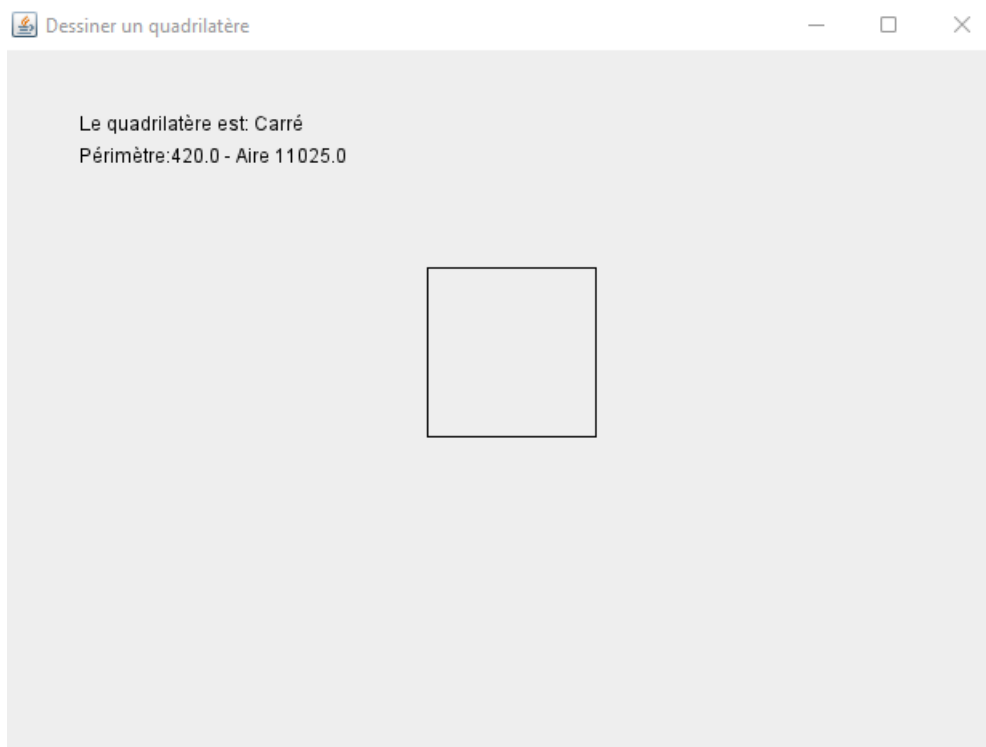
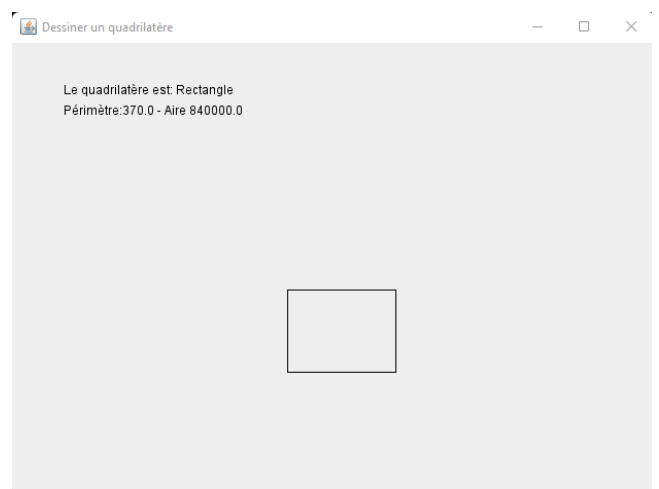
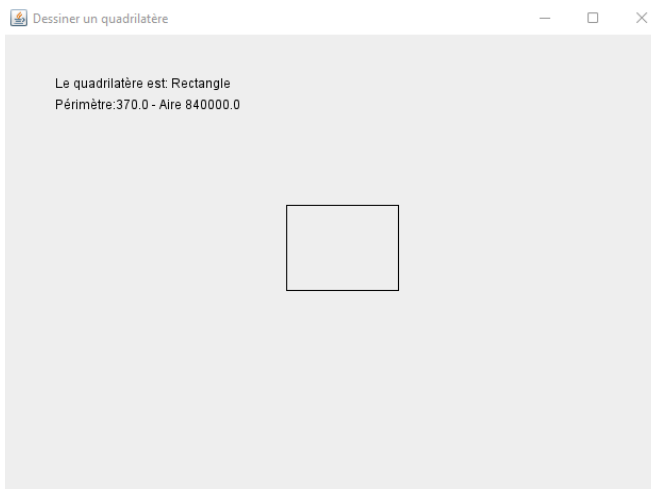
```
@AfterEach
public void effacer() {
    n++;
    q.setInterface(f);
    f.getContentPane().removeAll();
}
```

```
@AfterAll
public static void finishAll(){
    System.out.println(x: "Le programme est terminee.");
}
```

```
@Test
public void rectangle(){
    q = new Quadrilatere(a: 105,b: 80,angle1: 90,angle2: 90);
    q1 = new Quadrilatere(a: 105,b: 80,-90,-90);
    q1.setInterface(f);
    f.getContentPane().removeAll();
    Assertions.assertTrue(Double.compare(q.aire(), q1.aire())==0);
}
```

```
@Test
public void carre(){
    q = new Quadrilatere(a: 105,b: 105,angle1: 90,angle2: 90);
    String actual = q.getType();
    String expected = "Carré";
    Assumptions.assertTrue(expected== actual);
}
```

```
@Test
public void losange(){
    q = new Quadrilatere(a: 105,b: 105,angle1: 70,angle2: 110);
    Double expected = 105 * Math.sin(Math.toRadians(angdeg: 70))*105*100;
    Assertions.assertFalse(Double.compare(q.aire(),Math.round(expected)/100)==0);
}
```



```
[INFO] Running QuadrilatereTest
Begin Test1

Begin Test2

Begin Test3

Le programme est terminee.
[ERROR] Tests run: 3, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 12.22 s <<< FAILURE! - in QuadrilatereTest
[ERROR] losange Time elapsed: 3.009 s <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <false> but was: <true>
    at QuadrilatereTest.losange(QuadrilatereTest.java:61)
```