JUnit 5

GHERGHOUCH ABDESSALAM

TABLE DE MATIÈRE

QU'EST-CE QUE JUNIT 5 ?

JUnit est un framework de signe protestant chez le parler de rationalisation Java. JUnit a dupe un conséquence inappréciable là-dedans lequel le soft du soft piloté par les tests et dramatique verset d'une décrassage de frameworks de tests unitaires connus précisément en deçà le nom de xUnit, à l'ascendance de JUnit.

Ce tutoriel explique l'manipulation de **JUnit** là-dedans lequel les tests unitaires de votre projet, univers en tortueux pile Java. Après actif cessé ce didacticiel, toi-même-même acquerrez des connaissances suffisantes là-dedans lequel l'manipulation du enveloppe de signe **JUnit** à cruellement desquelles toi-même-même pourrez déraper aux horizontaux suivants.

CARACTÉRISTIQUES DE JUNIT

JUnit offre :

- des primitives pour créer un test (assertions)
- statistiques sur l'exécution des tests
- des primitives pour gérer des suites de tests
- interface graphique pour la couverture des tests
- JUnit plug-in pour Eclipse

CONFIGURATION DE L'ENVIRONNEMENT

JUnit est un framework pour Java, donc la toute première exigence est d'avoir JDK installé sur votre machine.

Étape 1 : Vérifier l'installation de Java sur votre machine

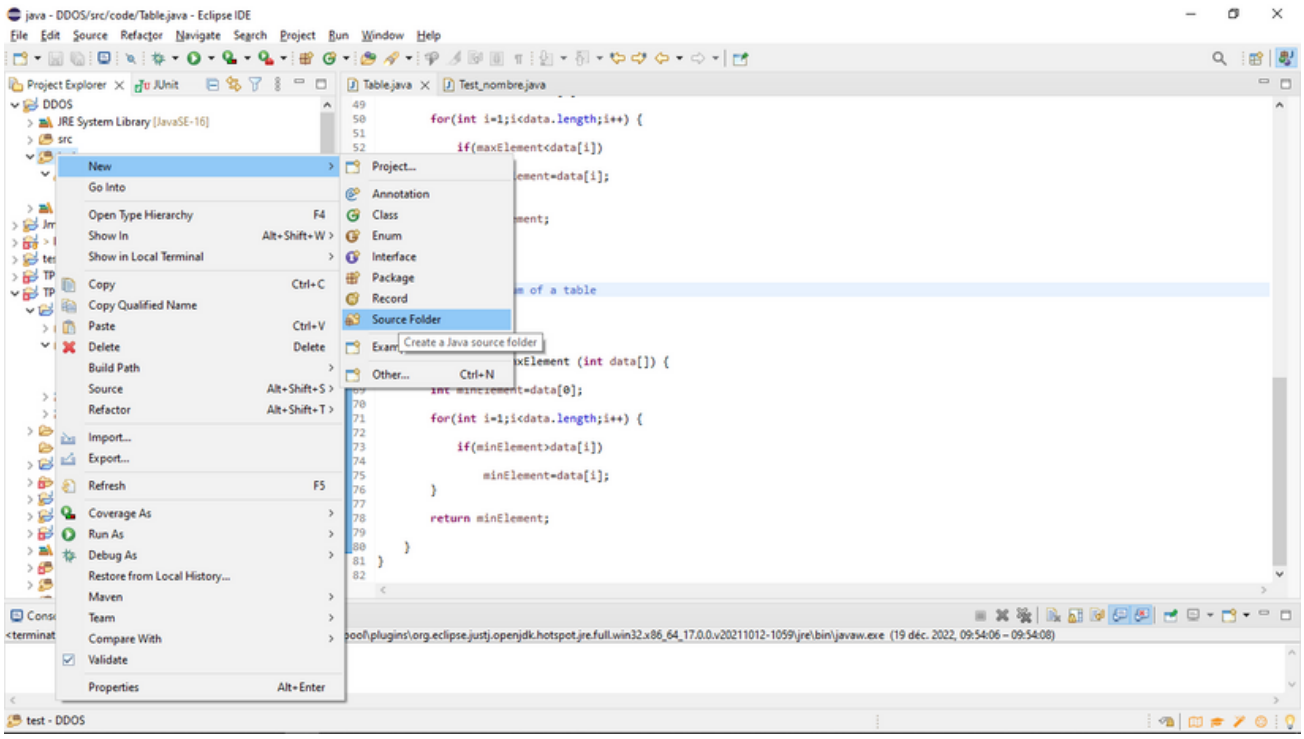
ouvrez la console et exécutez une commande java basée sur le système d'exploitation sur lequel vous travaillez

Activité	Commande
Windows	c:\> java -version
OS	$ java -version
Linux	machine:~ DDOS$ java -version

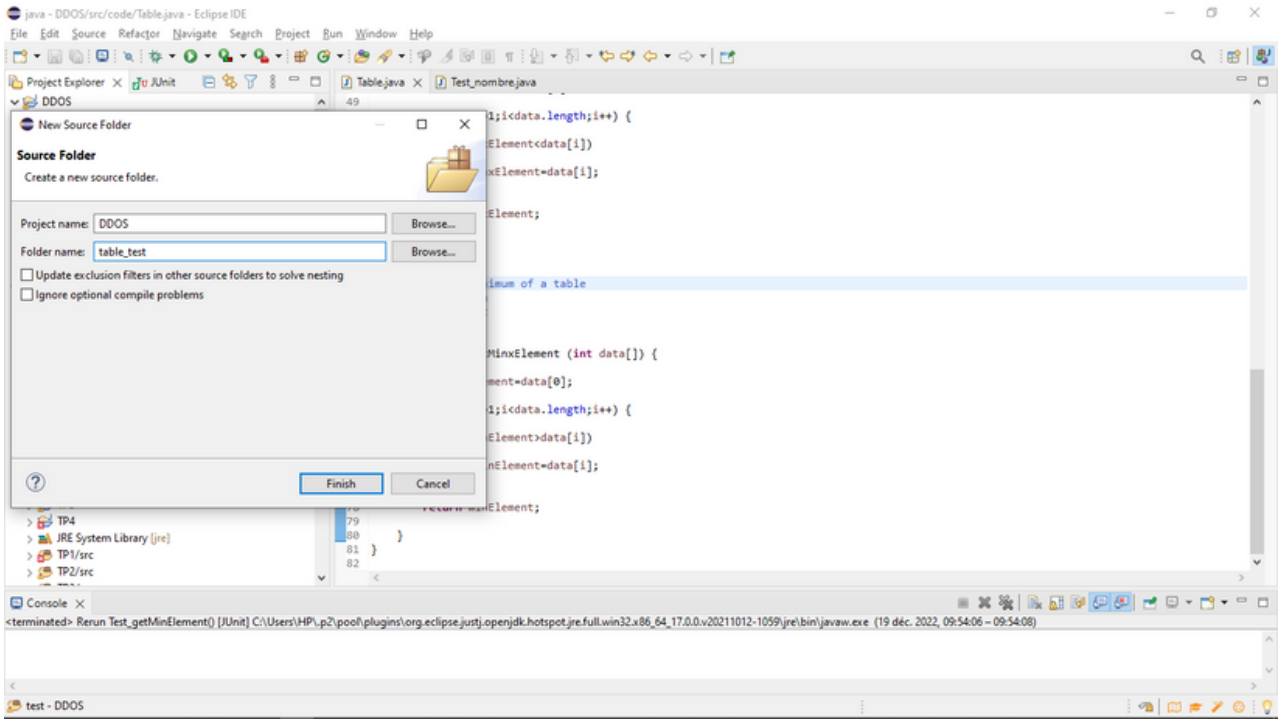
Étape 2 : Télécharger l'archive JUnit

Téléchargez la dernière version du fichier jar JUnit **http://www.junit.org.** sur moment de la rédaction de ce didacticiel, nous avons téléchargé Junit-4.12.jar et l'avons copié dans le dossier C:\>JUnit.

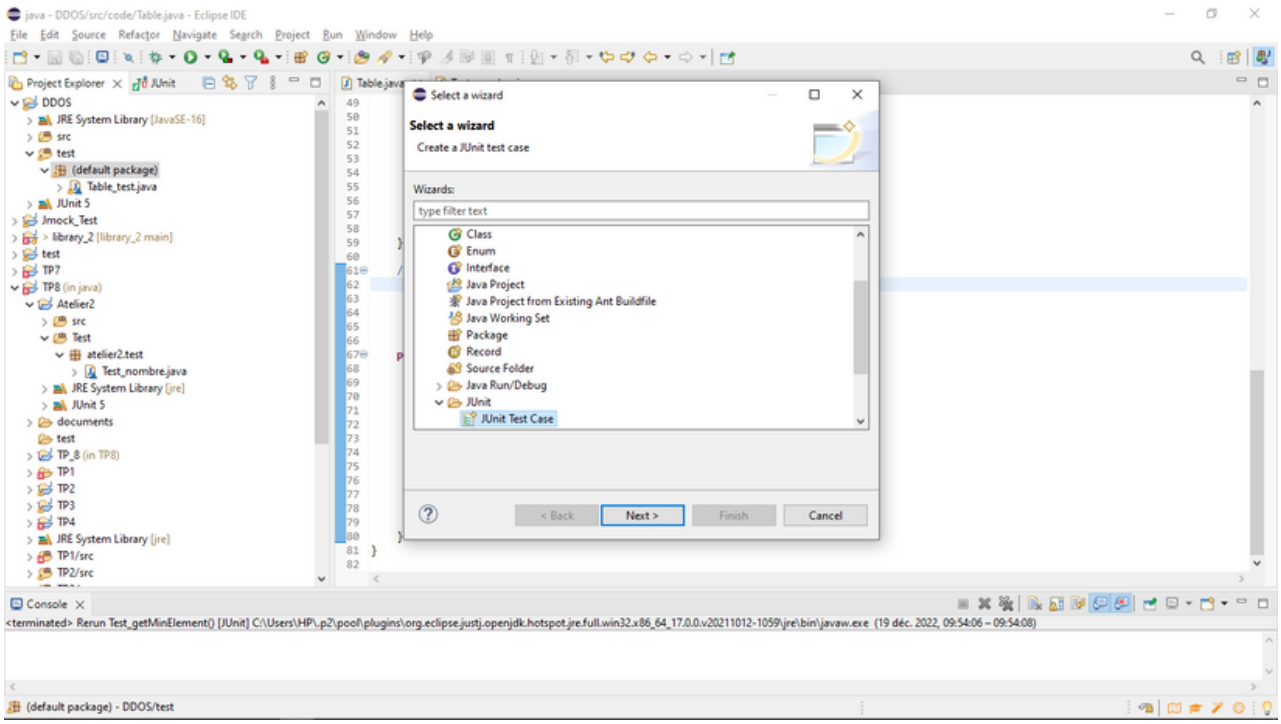
Étape 3:



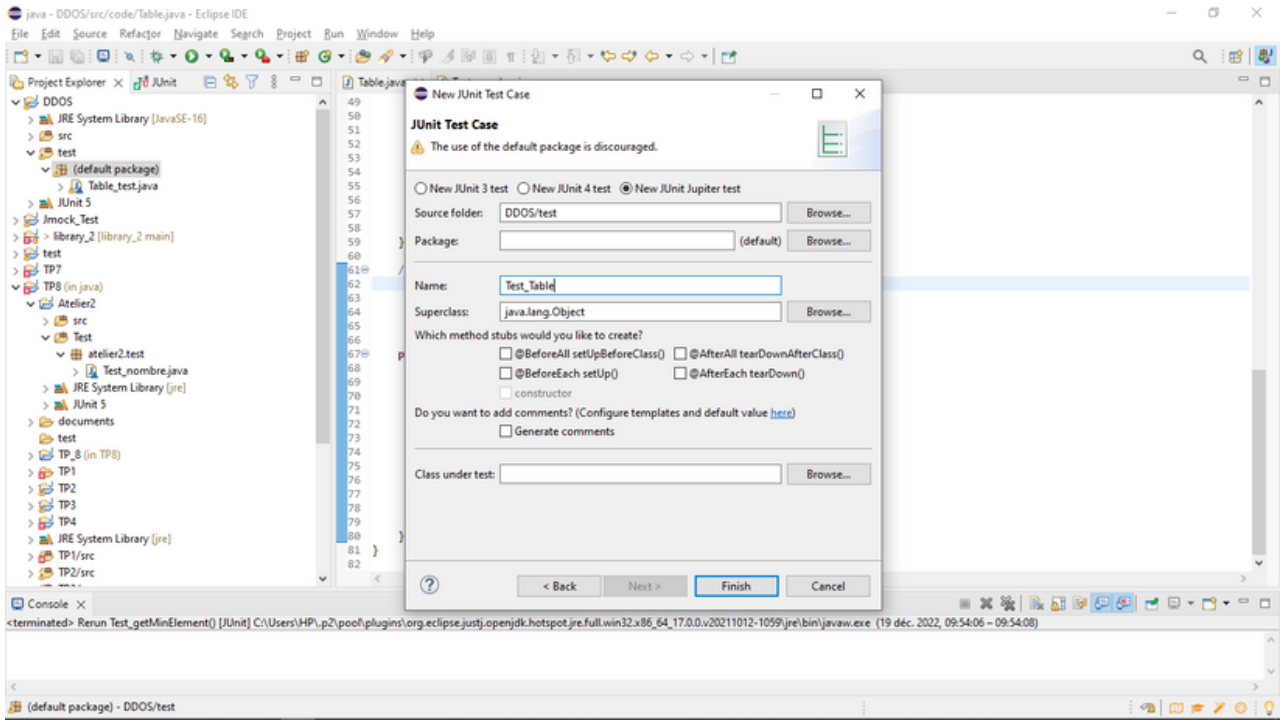
Étape 4:



Étape 5:



Étape 6:



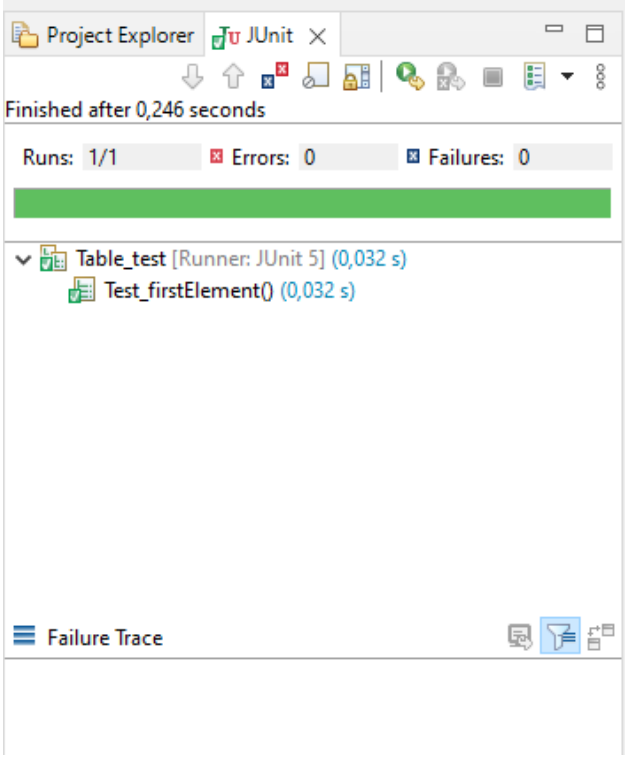
exemple1 des testes :

```java
/**
 * get the first element of a table
 * @param data
 * @return int
 */

public int firstElement(int data[]) {

    int fstElement=data[0];

    return fstElement;
}
```

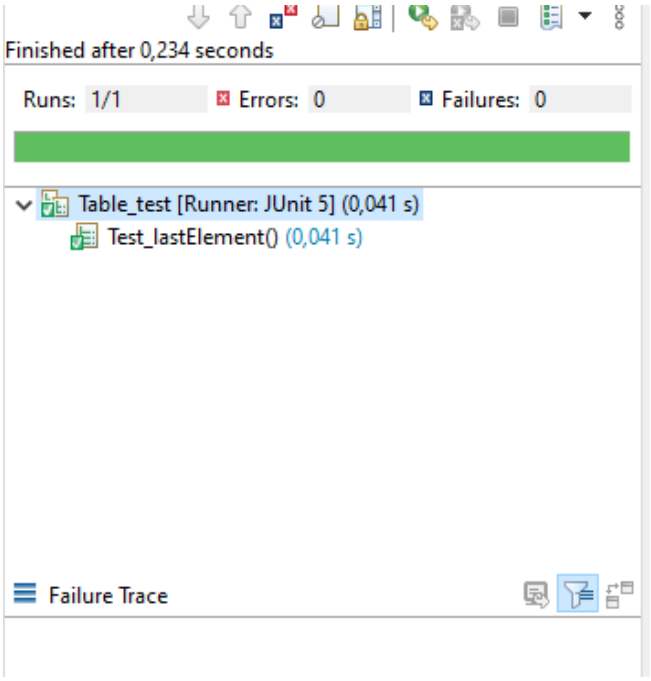


exemple2 des testes :

```java
@Test
void Test_lastElement() {
    // preparation des objets
    int[] tab1 = {2 , 3 , 9 , 10 , 1 , 5};
    Table t = new Table(tab1);

    int first = t.lastElement(tab1);

    assertEquals(5 , first);
}
```

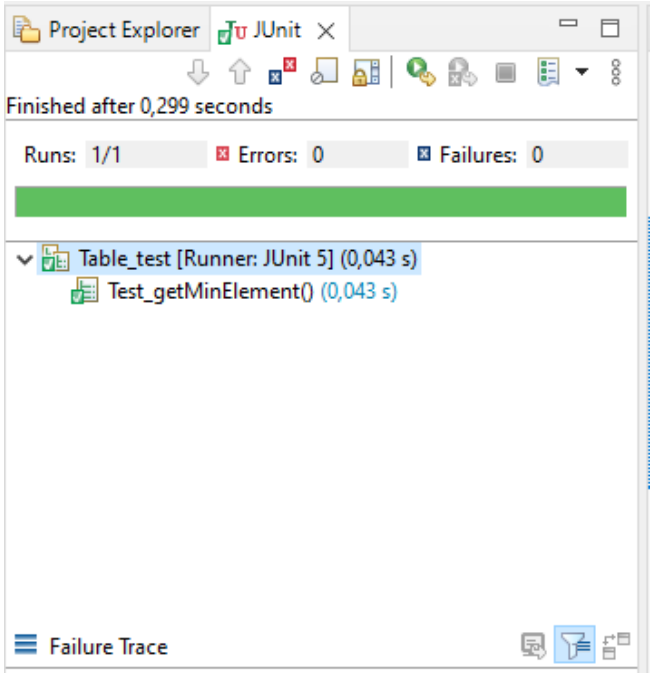


exemple3 des testes :

```java
@Test
void Test_getMinElement() {
    // preparation des objets
    int[] tab1 = {2 , 3 , 9 , 10 , 1 , 5};
    Table t = new Table(tab1);

    int first = t.getMinxElement(tab1);

    assertEquals(1 , first);
}
```

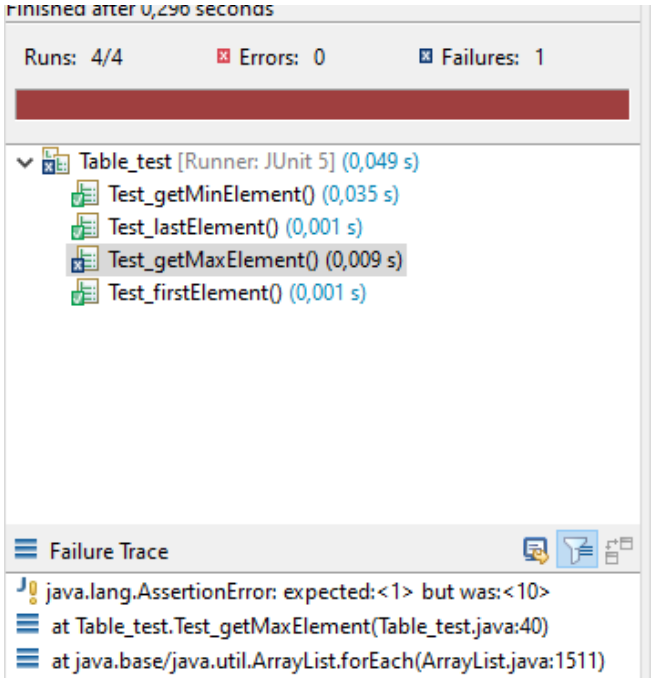


exemple4 des testes :

```java
@Test
void Test_getMaxElement() {
    // preparation des objets
    int[] tab1 = {2 , 3 , 9 , 10 , 1 , 5};
    Table t = new Table(tab1);

    int first = t.getMaxElement(tab1);

    assertEquals(1 , first);
}
```

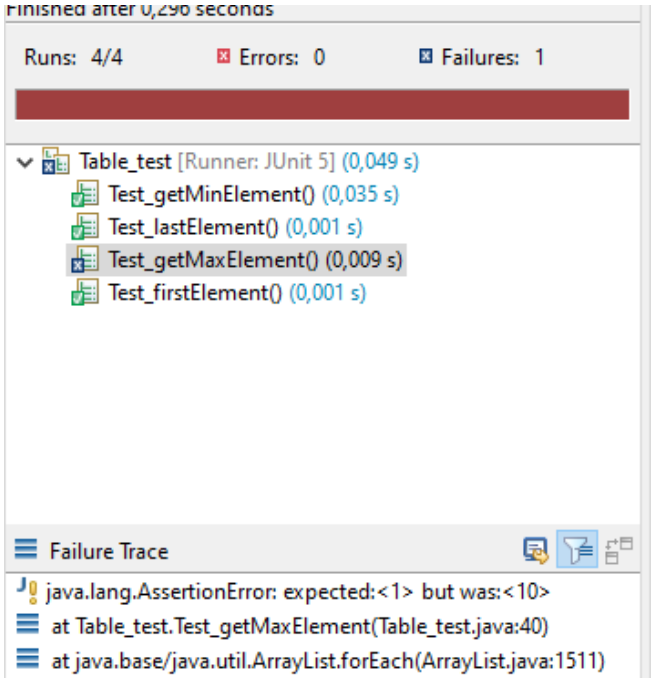


exemple5 des testes :

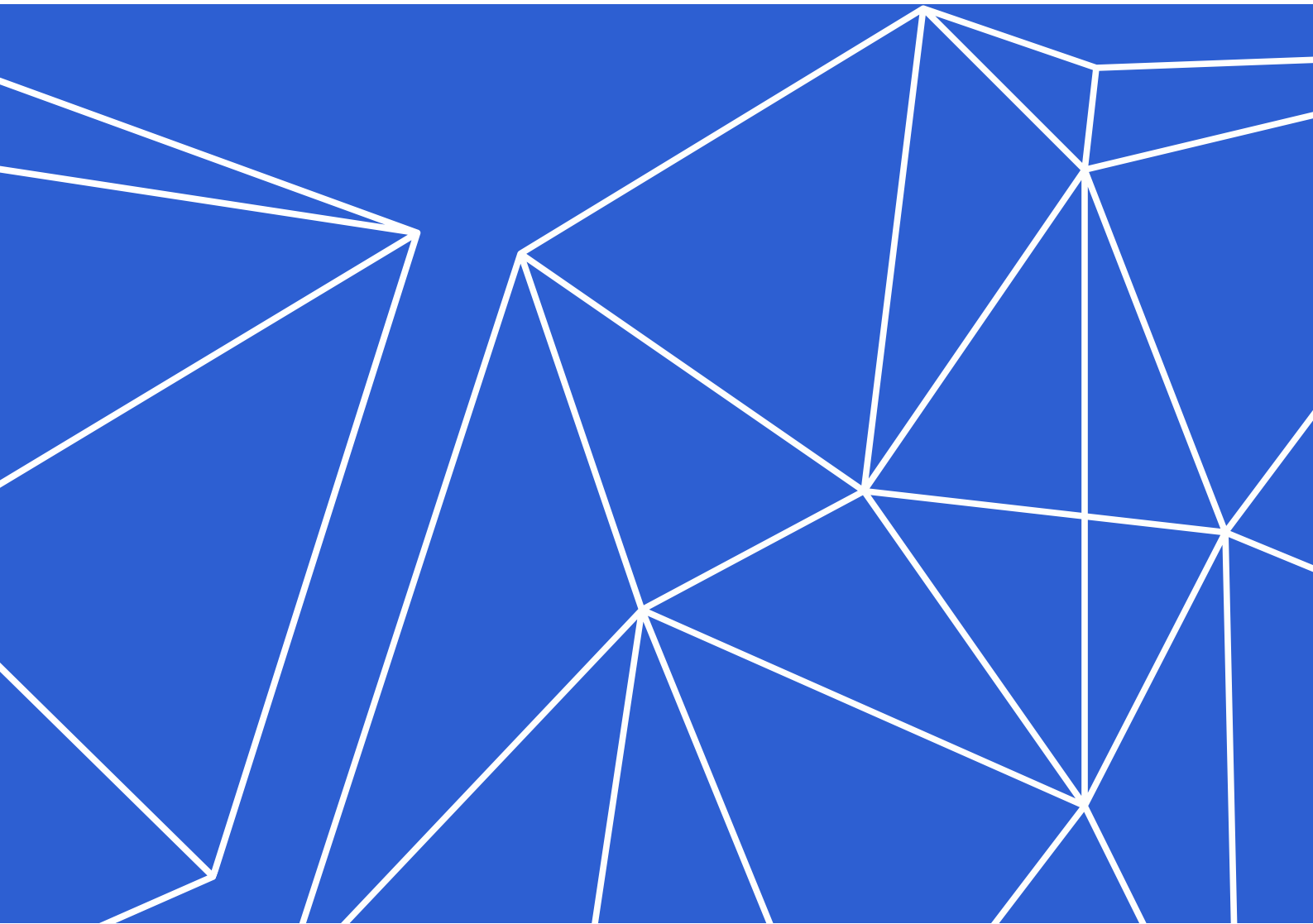
```
@Test
void Test_getMaxElement() {
    // preparation des objets
    int[] tab1 = {2 , 3 , 9 , 10 , 1 , 5};
    Table t = new Table(tab1);

    int first = t.getMaxElement(tab1);

    assertEquals(1 , first);
}
```



JMock



QU'EST-CE QUE JMOCK ?

C'est une Bibliothèque qui prend en charge le développement piloté par les tests de code Java avec des objets fictifs. Les objets fictifs aident à concevoir et à tester les interactions entre les objets de vos programmes. JMock se concentre sur la spécification explicite du comportement des simulacres à l'aide d'un DSL (Domain-Specific Language) spécialisé intégré dans le code Java.

EXEMPLE:

1- créer une interface

```java

public interface ExternalComponent {
  public String getData();
}
```

2- Ensuite, nous allons définir la classe que nous voulons tester, qui dépend d'une instance de l'interface ExternalComponent

```java

public class MyClass {
  private ExternalComponent component;

  public MyClass(ExternalComponent component) {
    this.component = component;
  }

  public String doSomething() {
    return component.getData();
  }
}
```

3-Maintenant, nous pouvons utiliser JMock pour créer une implémentation fictive de l'interface ExternalComponent et tester le comportement de MyClass

```java
import org.jmock.Expectations;
import org.jmock.Mockery;
import org.jmock.integration.junit4.JMock;
import org.jmock.integration.junit4.JUnit4Mockery;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(JMock.class)
public class MyClassTest {
  private Mockery context = new JUnit4Mockery();

  @Test
  public void testDoSomething() {
    // Create a mock ExternalComponent
    final ExternalComponent component = context.mock(ExternalComponent.class);

    // Set up the expectations for the mock component
    context.checking(new Expectations() {{
      oneOf (component).getData();
      will(returnValue("some data"));
    }});

    // Create an instance of MyClass with the mock component
    MyClass myClass = new MyClass(component);

    // Test the doSomething() method
    assertEquals("some data", myClass.doSomething());
  }
}
```