

SOMMAIRE

1. Le test Logiciel
2. Le Test automatique
3. Principes fondamentaux du Développement dirigé par les tests : *TDD Test Driven Development*
4. Framework Junit
5. Framework de tests automatisés



FRAMEWORK JUNIT

Dr. Soukeina Ben Chikha

CONTEXTE

1. Identifier qu'est ce qu'on va tester (coder😊). Quelle fonctionnalité et non quelle application : On est dans le cadre de test unitaire
2. Coder son test !
 - Il faut disposer d'un Framework de test permettant de créer et d'exécuter des tests automatisés.
 - bibliothèque Java : JUnit
 - Un IDE : Eclipse



A decorative graphic consisting of two thin, light brown lines that intersect on the left side of the slide. One line is oriented diagonally from the top-left towards the bottom-right, while the other is oriented diagonally from the top-right towards the bottom-left.

A. TESTS UNITAIRES

MISE EN PLACE D'UN PREMIER CYCLE TDD

1. Créer nouveau Projet Java
2. Créer votre première classes de votre application
3. Créer votre test de cette classe
4. Lancer le test

On va réaliser le projet avec comme exemple une calculatrice qui réalise des opération arithmétiques

MISE EN PLACE D'UN PREMIER CYCLE TDD

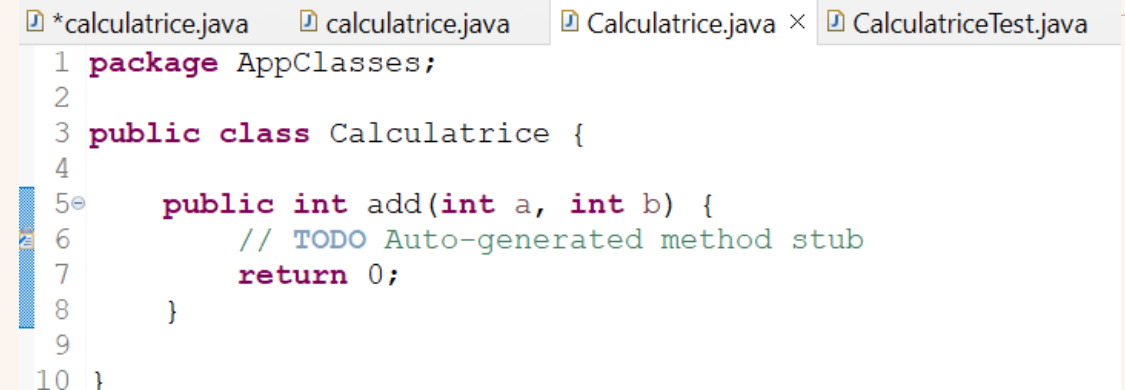
Installer Eclipse SDK (fait)

Installer JUNIT5

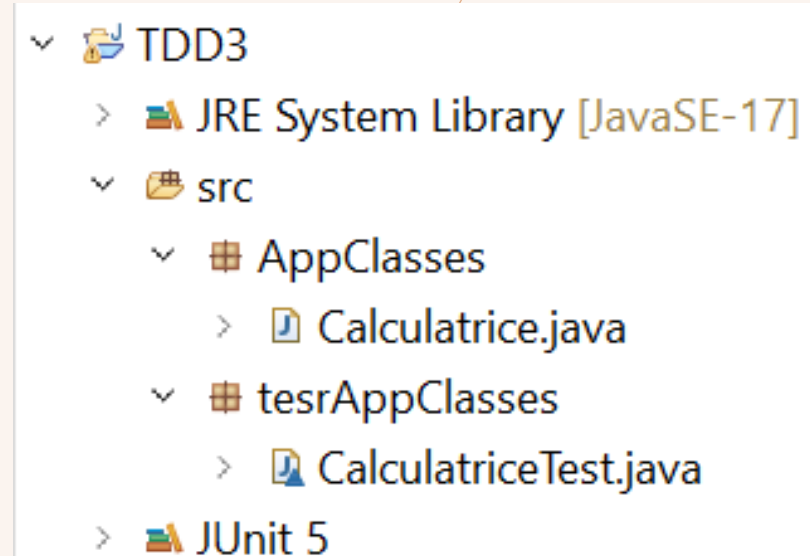
Lancer un nouveau projet Java

File > New > Projet Java

- Créer classe Calculatrice avec une méthode add
- Créer des packages différents pour les classes de l'application à développer et celles de test



```
1 package AppClasses;
2
3 public class Calculatrice {
4
5     public int add(int a, int b) {
6         // TODO Auto-generated method stub
7         return 0;
8     }
9
10 }
```



```

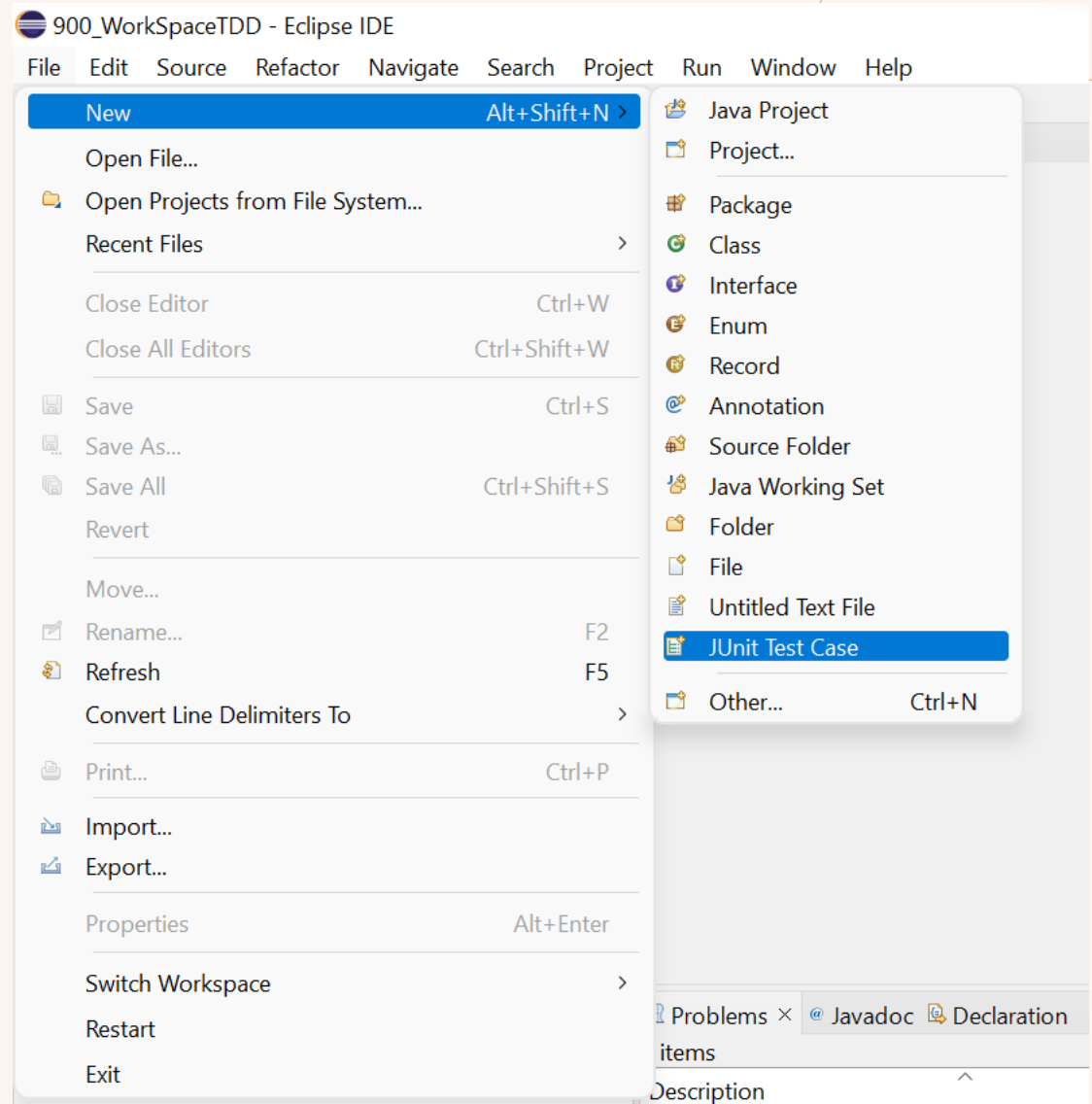
v TDD3
  > JRE System Library [JavaSE-17]
  v src
    v AppClasses
      > Calculatrice.java
    v tesrAppClasses
      > CalculatriceTest.java
  > JUnit 5
```

MISE EN PLACE D'UN PREMIER CYCLE TDD

Installer Eclipse SDK

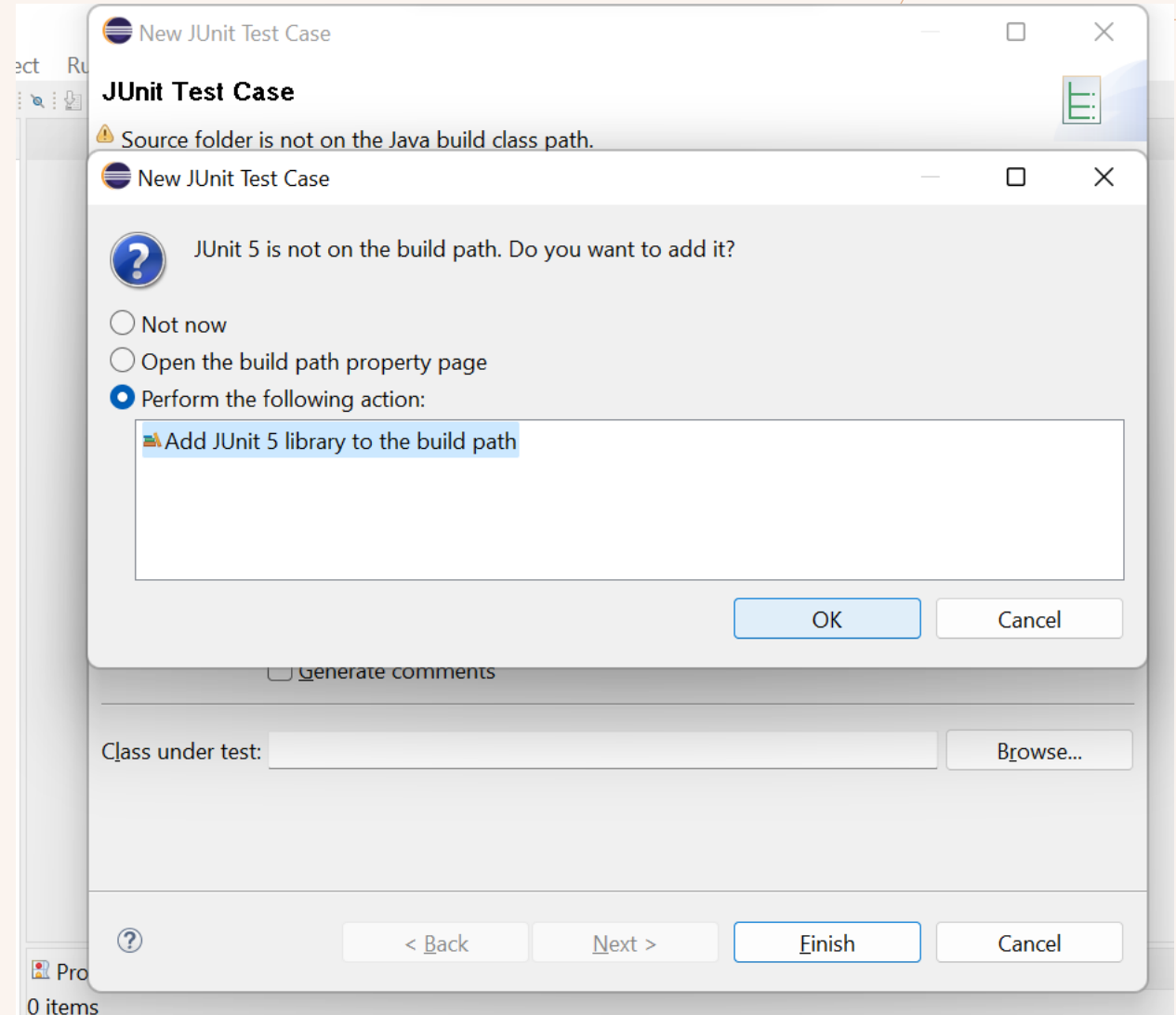
Dans le même projet java ajouter
une classe de test

File > New > JUnit Test Case



MISE EN PLACE D'UN PREMIER CYCLE TDD

- Si l'environnement vous demande d'ajouter JUnit 5 au classPath, >OK



MISE EN PLACE D'UN PREMIER CYCLE TDD

- La classe est créée avec une méthode par défaut **test**
- **fail**(String message)
- Échoue la test avec le message en paramètre.
- **@Test** est une annotation elle est obligatoire pour que le test soit lancé
- Elle sera abordée en détail plus tard

```
calcul_test.java x
1 package tdd;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6
7 class calcul_test {
8
9     @Test
10     void test() {
11         fail("Not yet implemented");
12     }
13
14 }
15
```

MISE EN PLACE D'UN PREMIER CYCLE TDD

- Modifier cette méthode avec une méthode calcul_Test avec les 3 A.
 - //Arrange
 - //Act
 - //Assert

```
*calcul_test.java ×
1 package tdd;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6
7 class calcul_test {
8
9     @Test
10    void testAddDeuxNbrPos() {
11        //1-Arrange
12
13        //2-Act
14
15        //3-Assert
16    }
17
18 }
19
```

MISE EN PLACE D'UN PREMIER CYCLE TDD

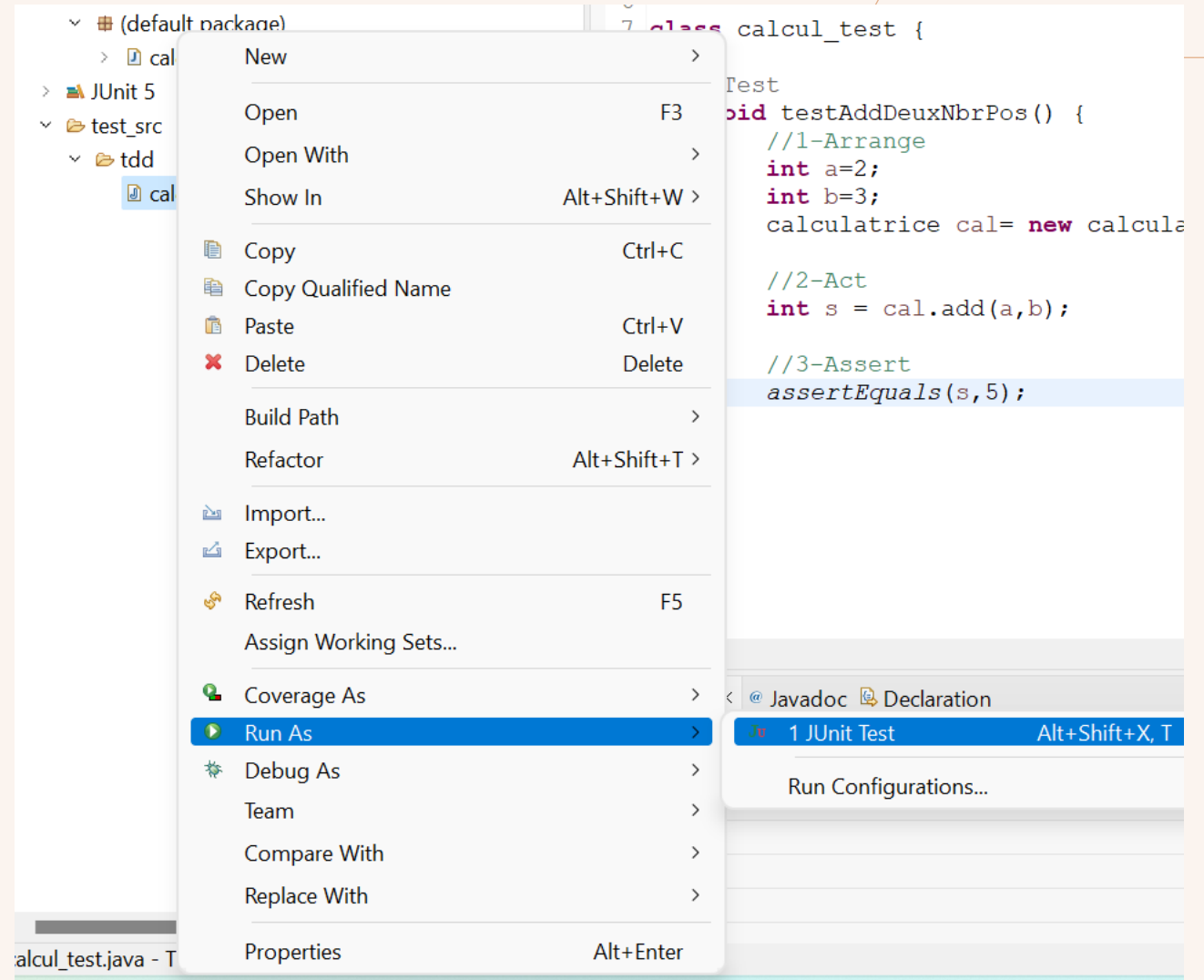
Saisir le code de

- ARRANGE
- ACT
- ASSERT

```
Calculatrice.java  *CalculatriceTest.java x
1 package tesrAppClasses;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class CalculatriceTest {
6
7     @Test
8     void testAddDeuxNbrPos() {
9         //1-Arrange
10        int a=2;
11        int b=3;
12        Calculatrice cal= new Calculatrice();
13
14        //2-Act
15        int s = cal.add(a,b);
16
17        //3-Assert
18        /*assertEquals(reultat_attendu, resultat_obtenu)*/
19        assertEquals(5,s);
20    }
21 }
```

MISE EN PLACE D'UN PREMIER CYCLE TDD

- Lance le test
- Bouton droit fichier test > run
As > JUnitTest



MISE EN PLACE D'UN PREMIER CYCLE TDD

ROUGE

- Test échoue vue que la méthode add n'a pas été implémentée et retourne zéro.

900_WorkSpaceTDD - TDD3/src/tesrAppClasses/CalculatriceTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit x

Finished after 0,104 seconds

Runs: 1/1 Errors: 0 Failures: 1

CalculatriceTest [Runner: JUnit 5] (0,001 s)

testAddDeuxNbrPos() (0,001 s)

Failure Trace

org.opentest4j.AssertionFailedError: expected: <0> but was: <5>
at tesrAppClasses.CalculatriceTest.testAddDeuxNbrPos(CalculatriceTest.java:23)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

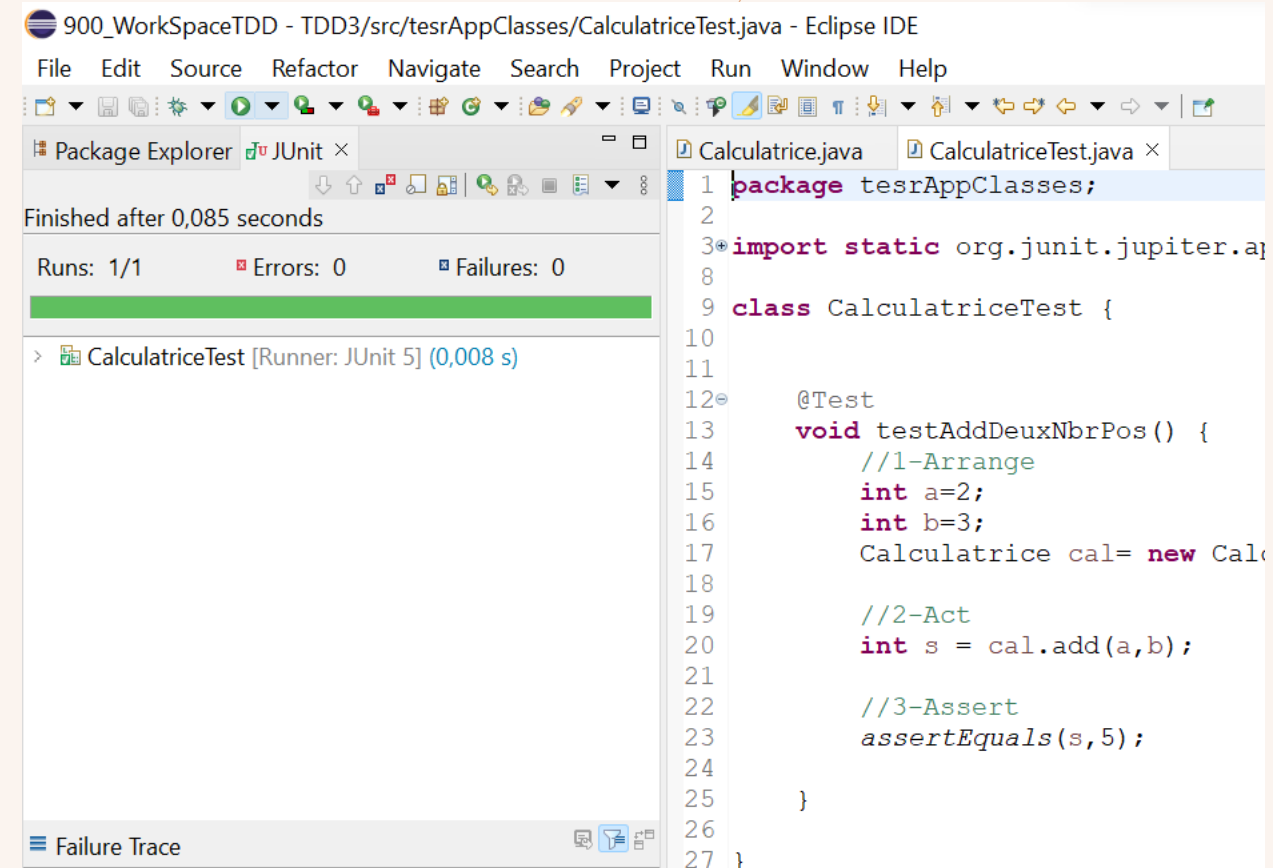
```
5 import org.junit.jupiter.api.Test;
6
7 import AppClasses.Calculatrice;
8
9 class CalculatriceTest {
10
11
12     @Test
13     void testAddDeuxNbrPos() {
14         //1-Arrange
15         int a=2;
16         int b=3;
17         Calculatrice cal= new Calculatrice();
18
19         //2-Act
20         int s = cal.add(a,b);
21
22         //3-Assert
23         assertEquals(s,5);
24
25     }
26
27 }
28
29
30
```

MISE EN PLACE D'UN PREMIER CYCLE TDD

VERT

- Ecrire le code la méthode add et relancer le test, le test réussi

```
Calculatrice.java × CalculatriceTest.java
1 package AppClasses;
2
3 public class Calculatrice {
4
5     public int add(int a, int b) {
6
7         return a+b;
8     }
9
10 }
11
```



MISE EN PLACE D'UN PREMIER CYCLE TDD

Remarques

- L'élément crucial à retenir par rapport au TDD, c'est que tous vos tests devraient échouer au début !
- Pour les classes de test utiliser un nom se terminant par Test. C'est une **convention** qui aidera les autres développeurs avec qui vous travaillez.
- Si l'assertion est fausse, le test est tout de suite en échec. S'il y a plusieurs assertions, toutes les assertions doivent être vraies. C'est pour cela qu'en général, il est préférable d'avoir une seule assertion par test, pour mieux cibler le test.

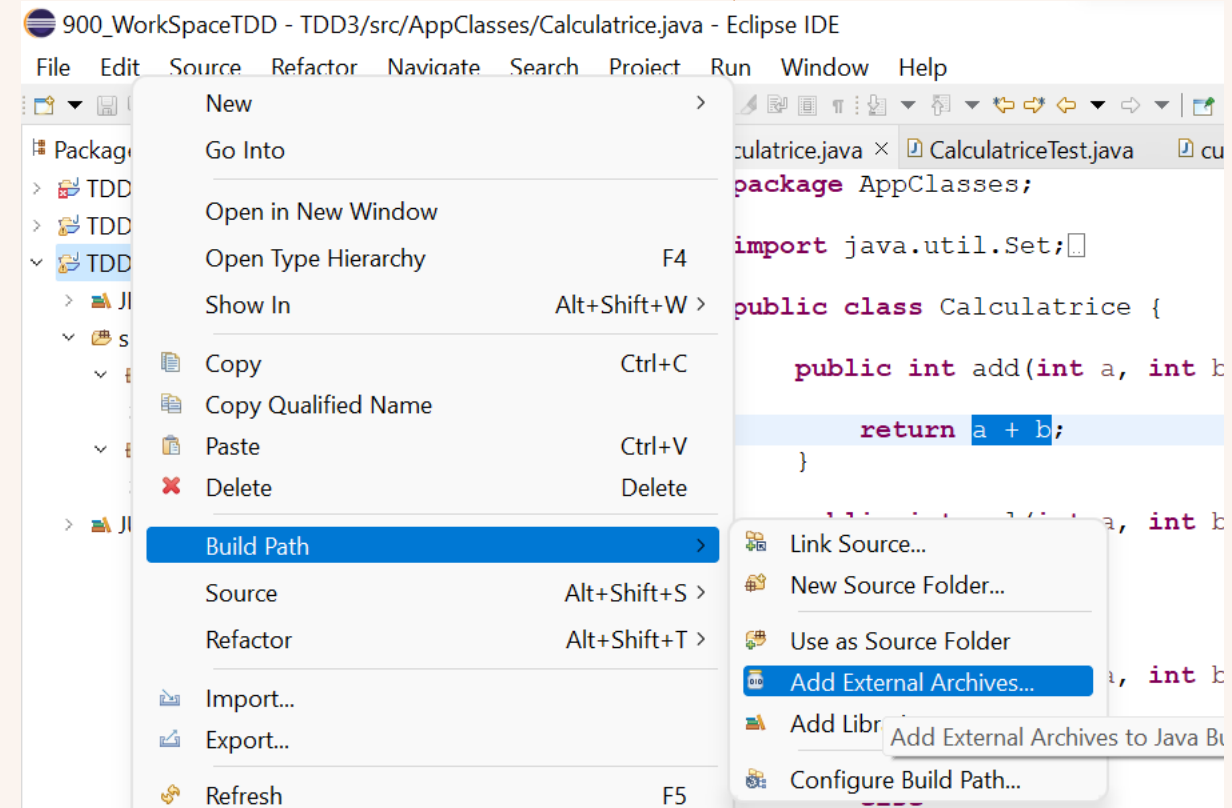
MISE EN PLACE D'UN PREMIER CYCLE TDD

Remarques

- Dans le TDD on va du moins spécifique au plus spécifique
- Les tests deviendront de plus en plus spécifiques lorsque on les code pour des scénarios plus spécifiques.
- Or, cela signifie que le code devient de plus en plus général, parce qu'il couvre plus de situations.

TP1

- Pour l'installation de Junit télécharger le jar file
- <https://mvnrepository.com/artifact/org.assertj/assertj-core/3.24.2>
- Enregistrer le fichier
- Click droit sur le projet > BuildPath > Add External Archine



TP1

Exercice 1

- Reprendre l'exemple de la calculatrice.
- Ajouter en TDD la multiplication de 2 nombres
- Ajouter la division de 2 nombres
- Tester le cas où le dénominateur est nul. Commenter ce cas.

Exercice 2

- Ecrire en TDD une fonctionnalité de conversion de devise (dollar-euro, euro-dollar, ...)
- Laisser trace de votre cycle de développement.