# JUnit Testing Exercises

**Exercise 1: Setting Up JUnit**

Scenario: You need to set up JUnit in your Java project to start writing unit tests.

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

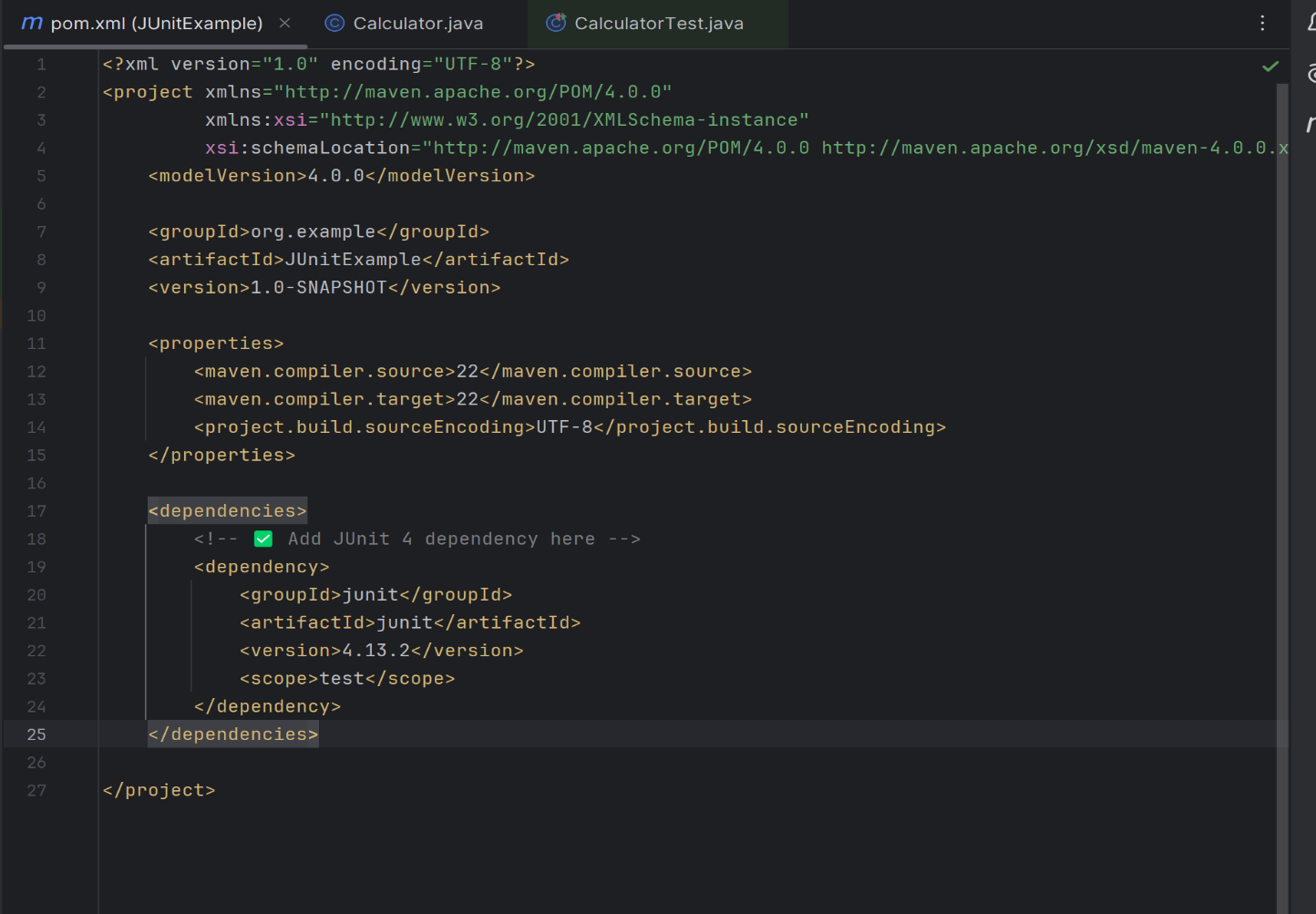
<version>4.13.2</version>

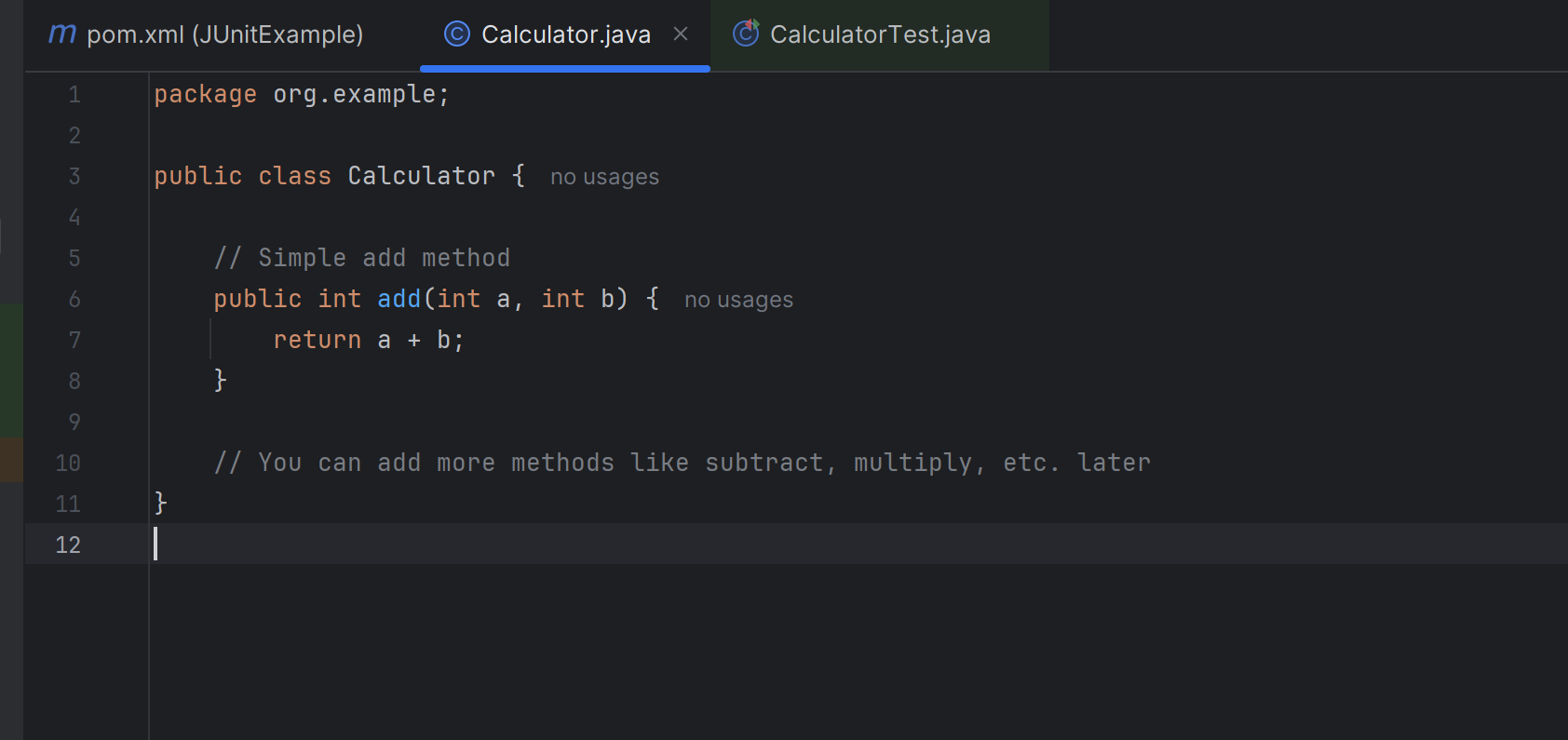
<scope>test</scope>

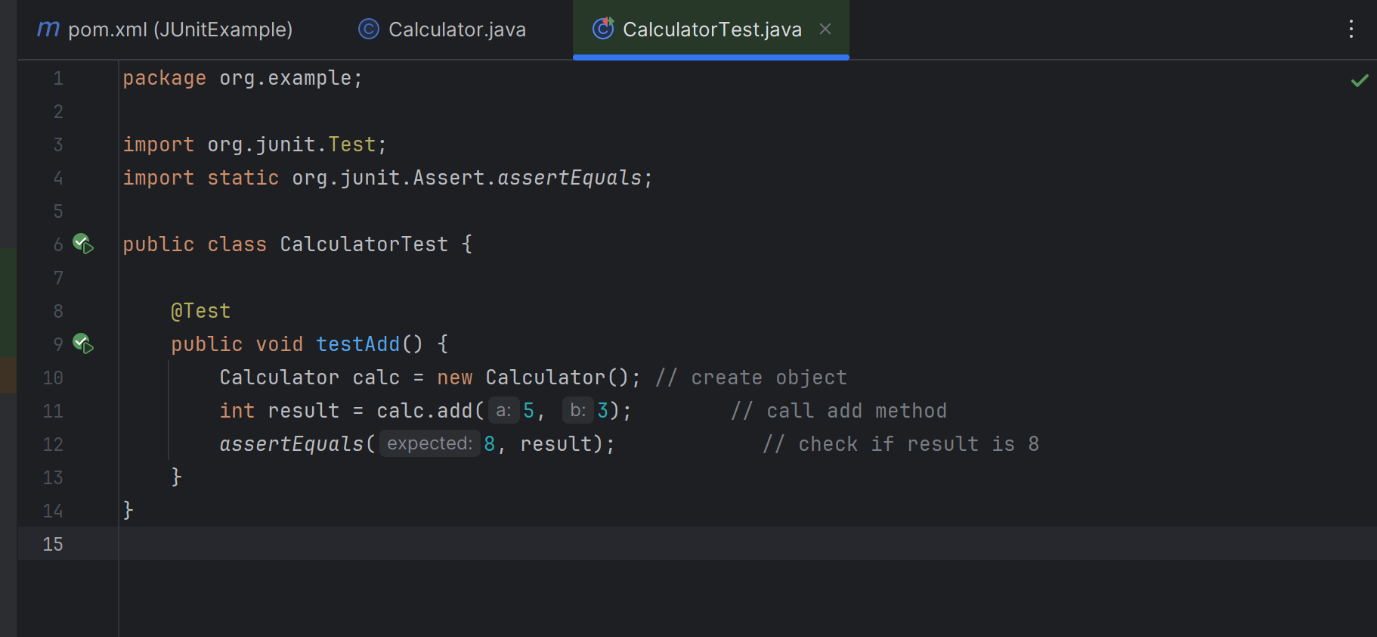
</dependency>

3. Create a new test class in your project.

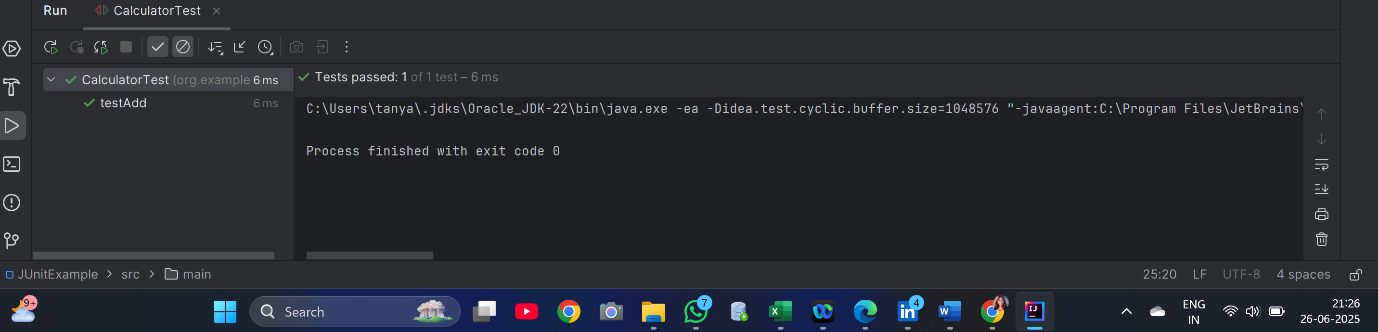
------------------------------------------------------------------------------------------------







**OUTPUT**



**Exercise 3: Assertions in JUnit**

Scenario: You need to use different assertions in JUnit to validate your test results.

Steps:

1. Write tests using various JUnit assertions.

Solution Code:

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals

assertEquals(5, 2 + 3);

// Assert true

assertTrue(5 > 3);

// Assert false

assertFalse(5 < 3);

// Assert null

assertNull(null);

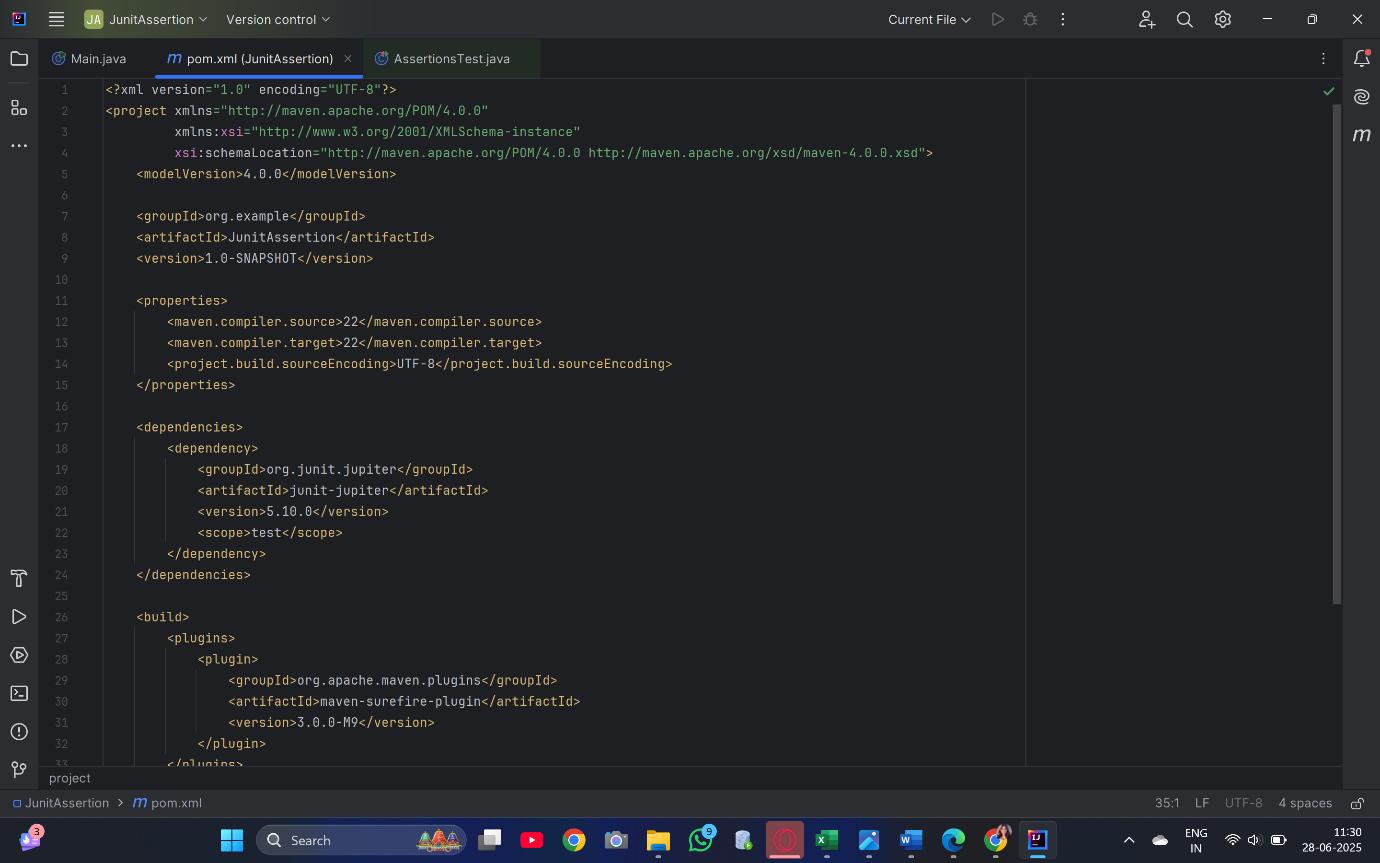
// Assert not null

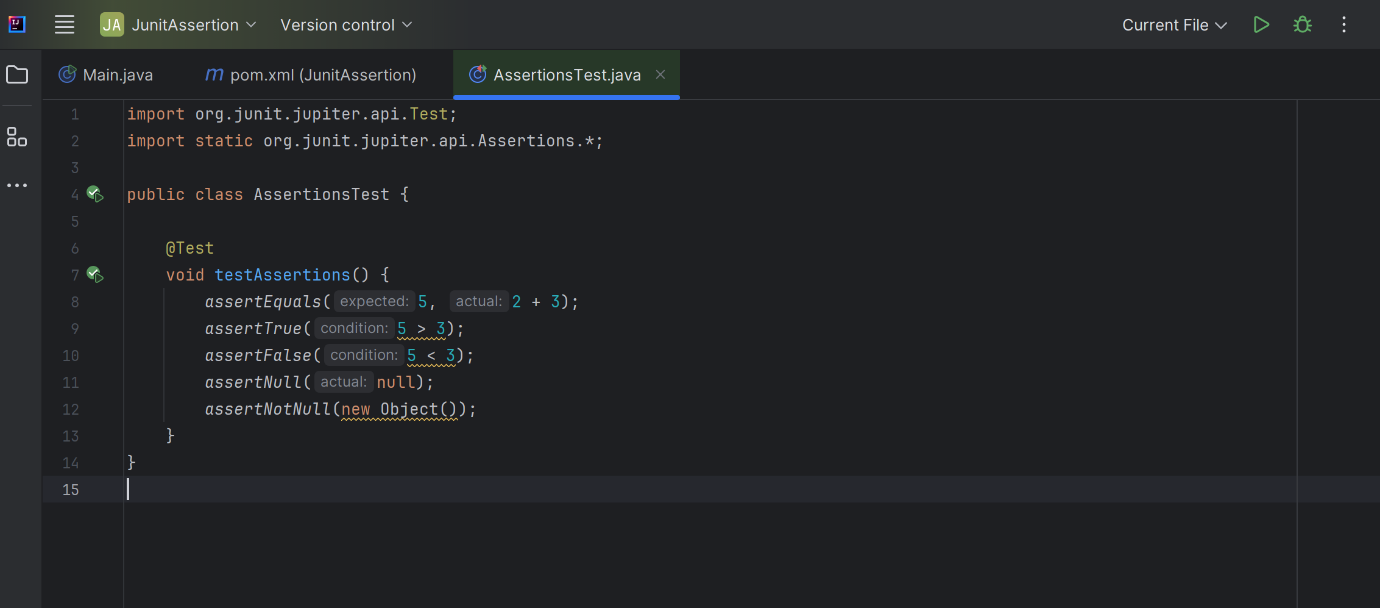
assertNotNull(new Object());

}

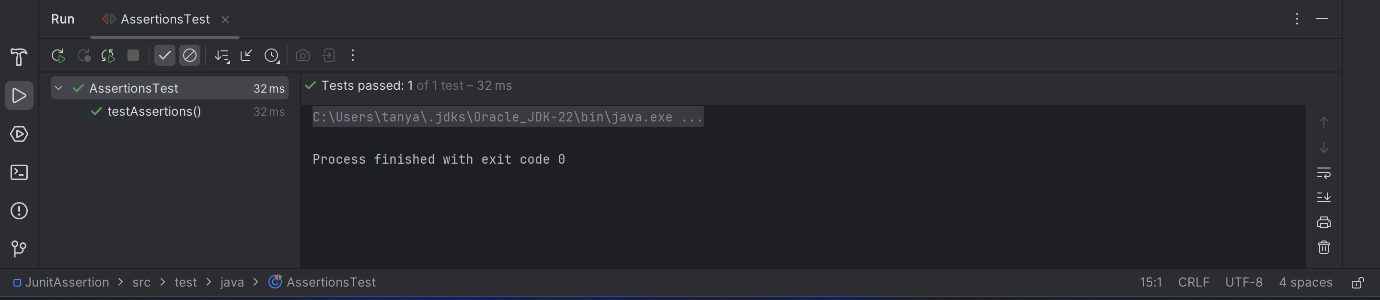
}

**SOLUTION**





**OUTPUT**

****

**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

Scenario: You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

Steps:

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods**.**

--------------------------------------------------------------------------------------------------------------------------

**SOLUTION**

**Breakdown of Key Concepts**

**✅ Arrange-Act-Assert (AAA):**

* **Arrange**: Set up test data and context.
* **Act**: Call the method being tested.
* **Assert**: Verify that the result is what you expected.

**✅ @Before**

* Runs **before each test method**.
* Used to **initialize test fixtures** (like calculator object).

**✅ @After**

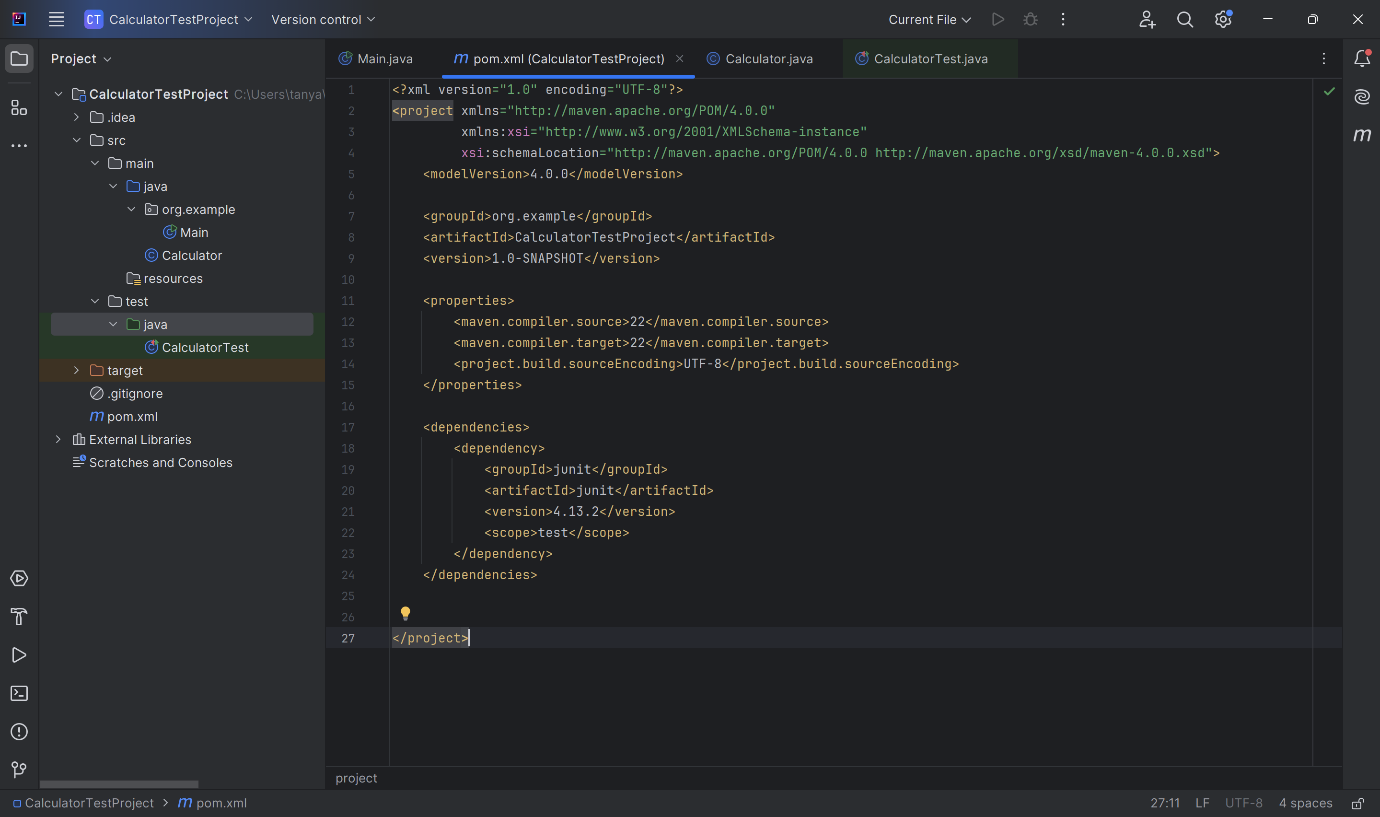
* Runs **after each test method**.
* Used to **release resources** or reset changes.

**Step 1: Create a Maven Project**

1. Open **IntelliJ IDEA**.
2. Go to **File > New > Project**.
3. Select **Maven** from the left panel.
4. Click **Next**, give it a name like CalculatorTestProject, and **Finish**.

**Step 2: Add JUnit 4 to pom.xml**

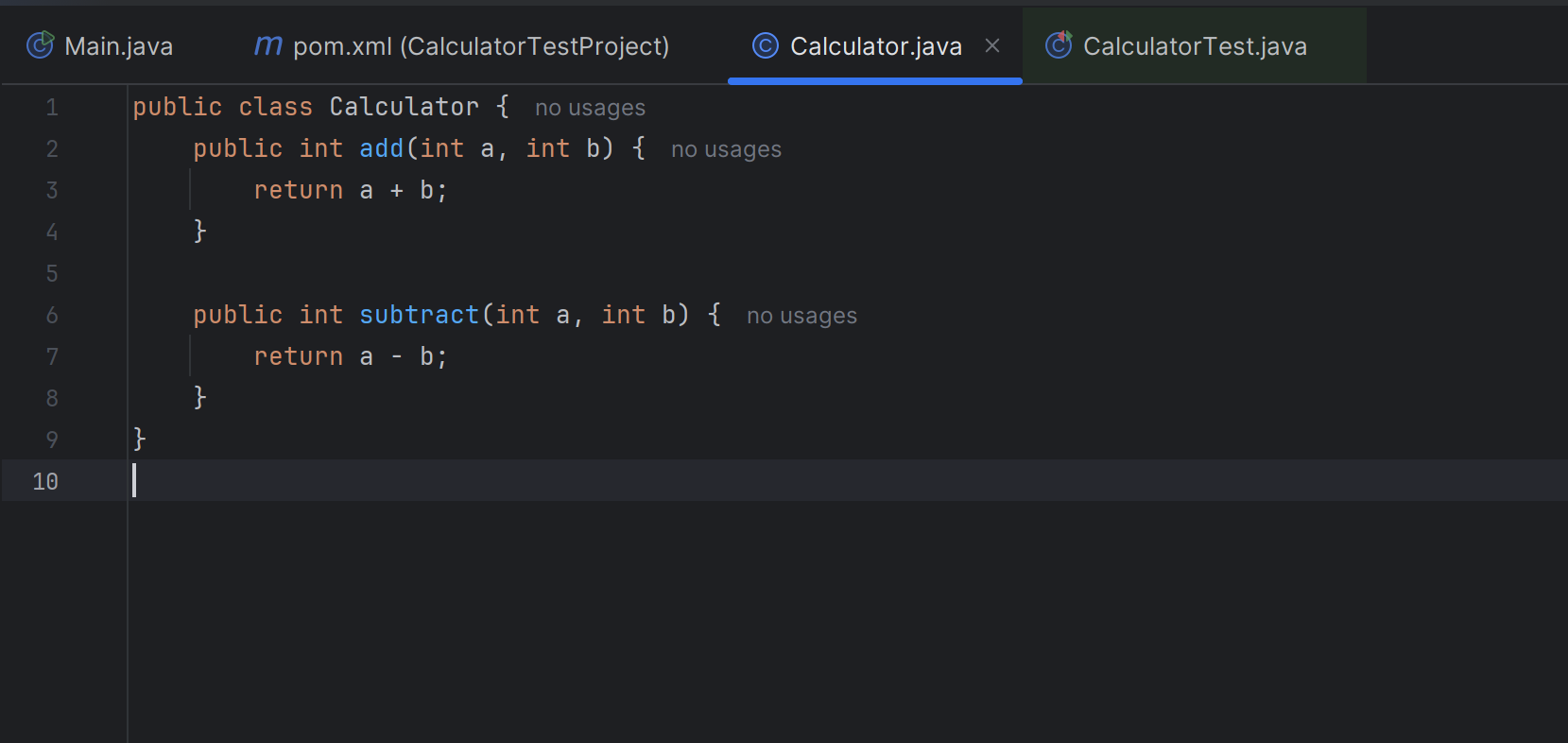
Open pom.xml and add the following inside the <dependencies> tag:

****

**Step 3: Create Java Classes**

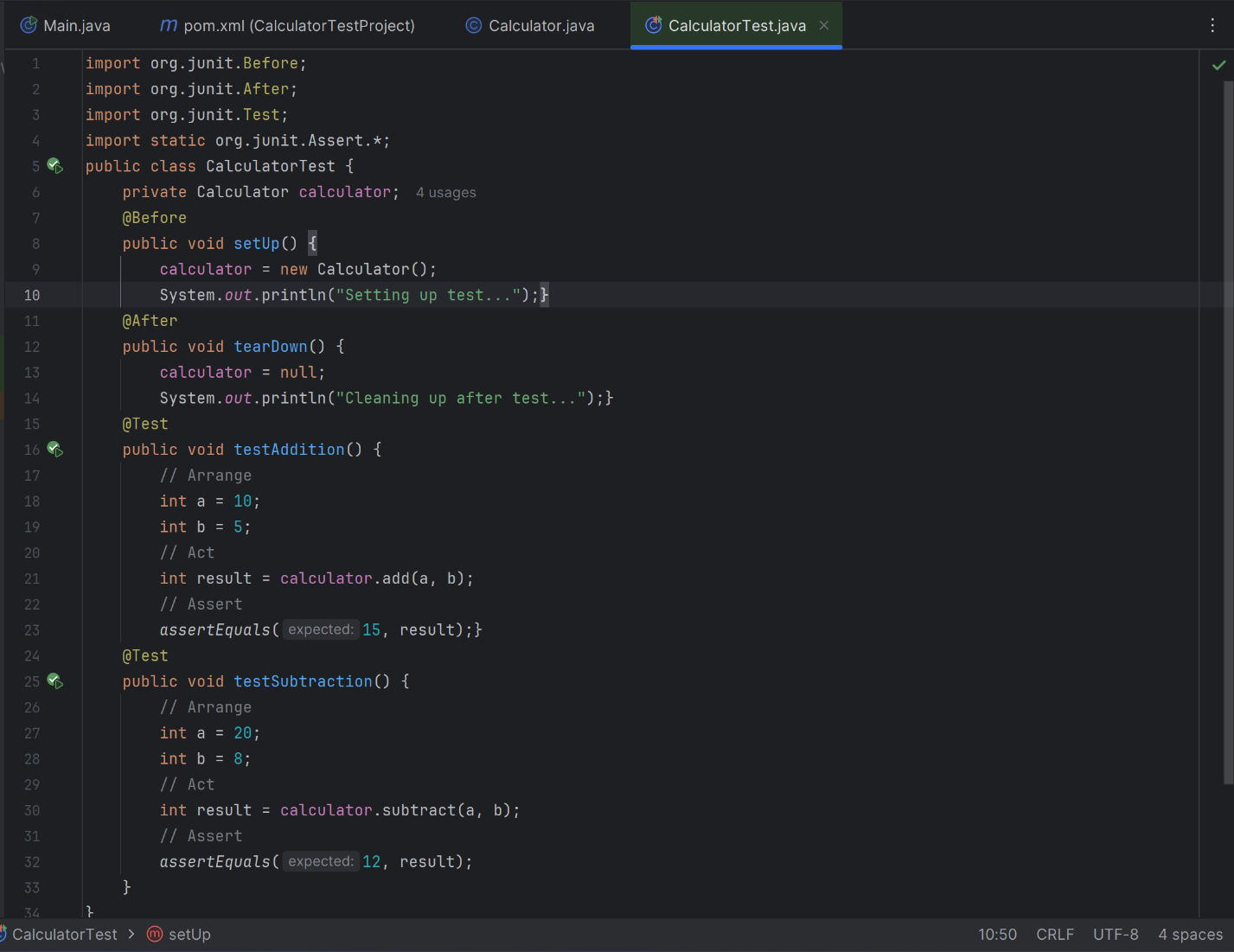
1. Create Calculator.java

Right-click src/main/java > New > Java Class > Name: Calculator

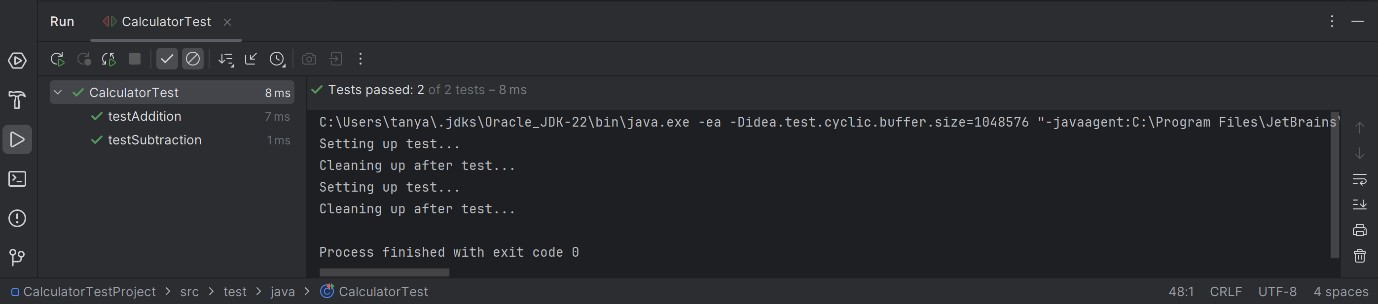
****

**2. Create CalculatorTest.java**

Right-click src/test/java > New > Java Class > Name: CalculatorTest

****

**OUTPUT**



=====================