JUnit Testing Exercises

# Exercise 1: Setting Up JUnit

Scenario:

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

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>

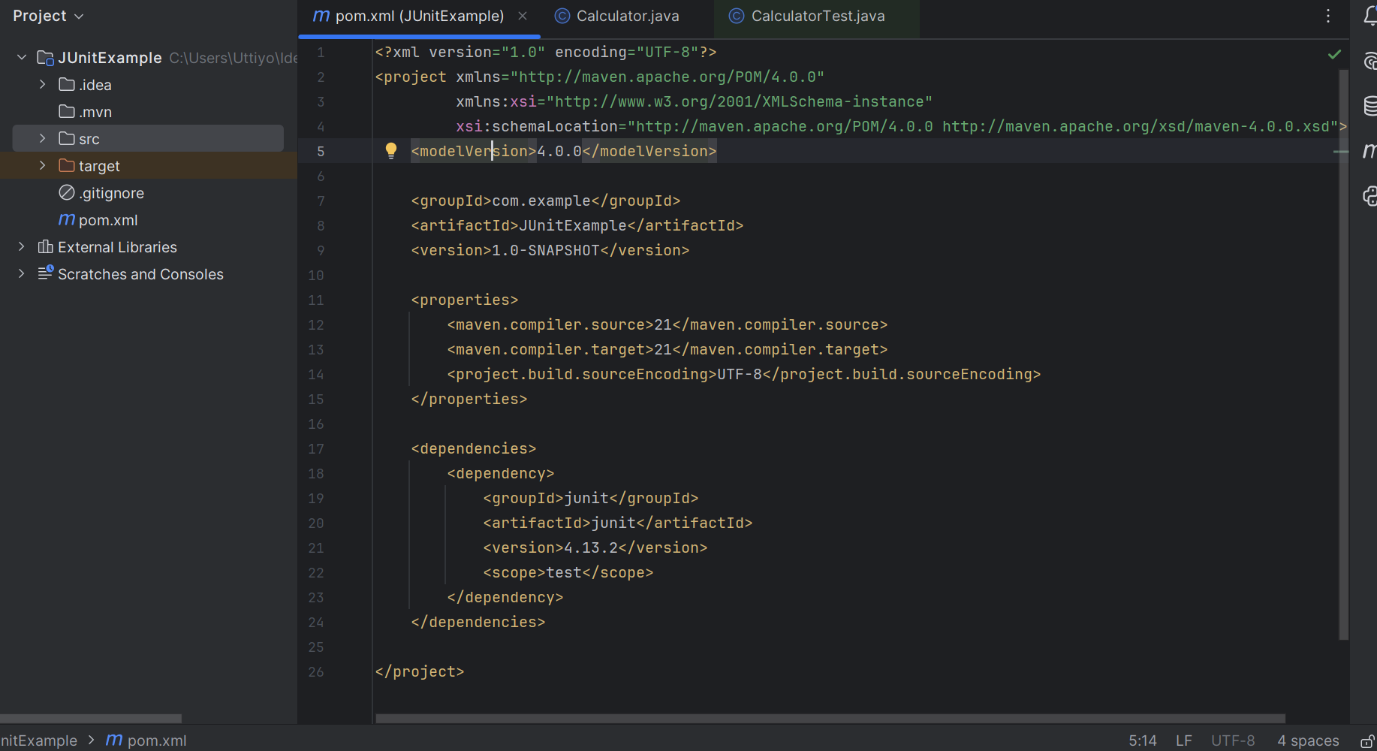
<groupId>junit</groupId>

<artifactId>junit</artifactId>

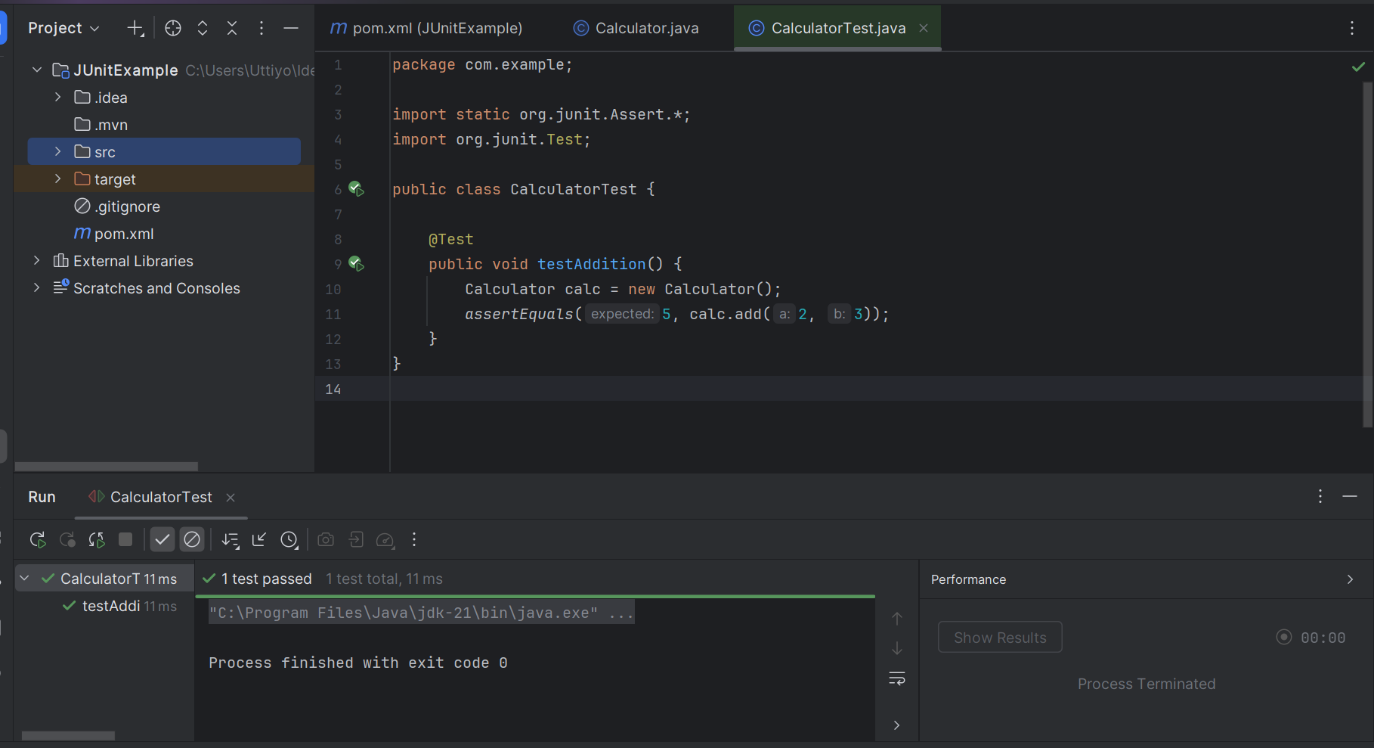
<version>4.13.2</version>

<scope>test</scope>

</dependency>



1. Create a new test class in your project.



# Exercise 2: Writing Basic JUnit Tests

Scenario:

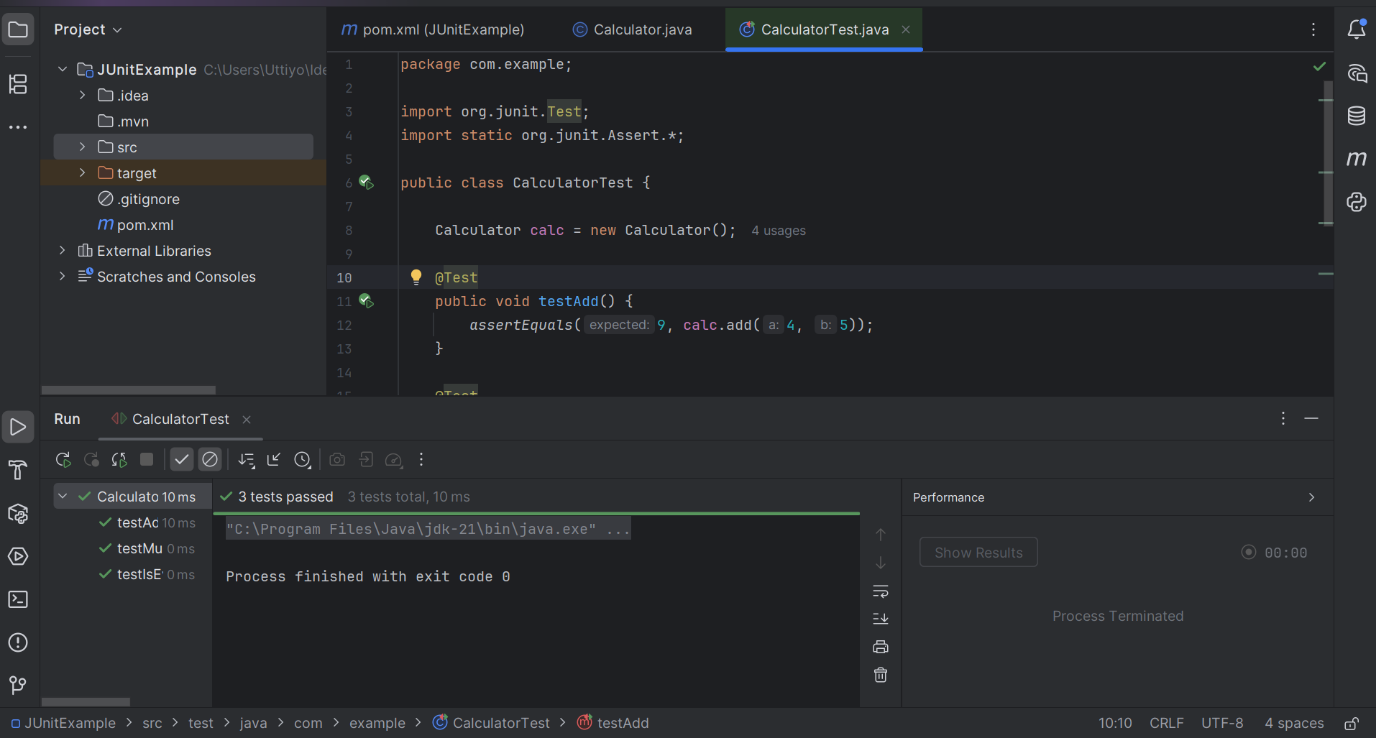
You need to write basic JUnit tests for a simple Java class. Steps:

1. Create a new Java class with some methods to test.

**package com.example;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
  
 public int multiply(int a, int b) {  
 return a \* b;  
 }  
  
 public boolean isEven(int num) {  
 return num % 2 == 0;  
 }  
}**

1. Write JUnit tests for these methods.

package com.example;  
  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class CalculatorTest {  
  
 Calculator calc = new Calculator();  
  
 @Test  
 public void testAdd() {  
 *assertEquals*(9, calc.add(4, 5));  
 }  
  
 @Test  
 public void testMultiply() {  
 *assertEquals*(20, calc.multiply(4, 5));  
 }  
  
 @Test  
 public void testIsEven() {  
 *assertTrue*(calc.isEven(6));  
 *assertFalse*(calc.isEven(7));  
 }  
}



# 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);

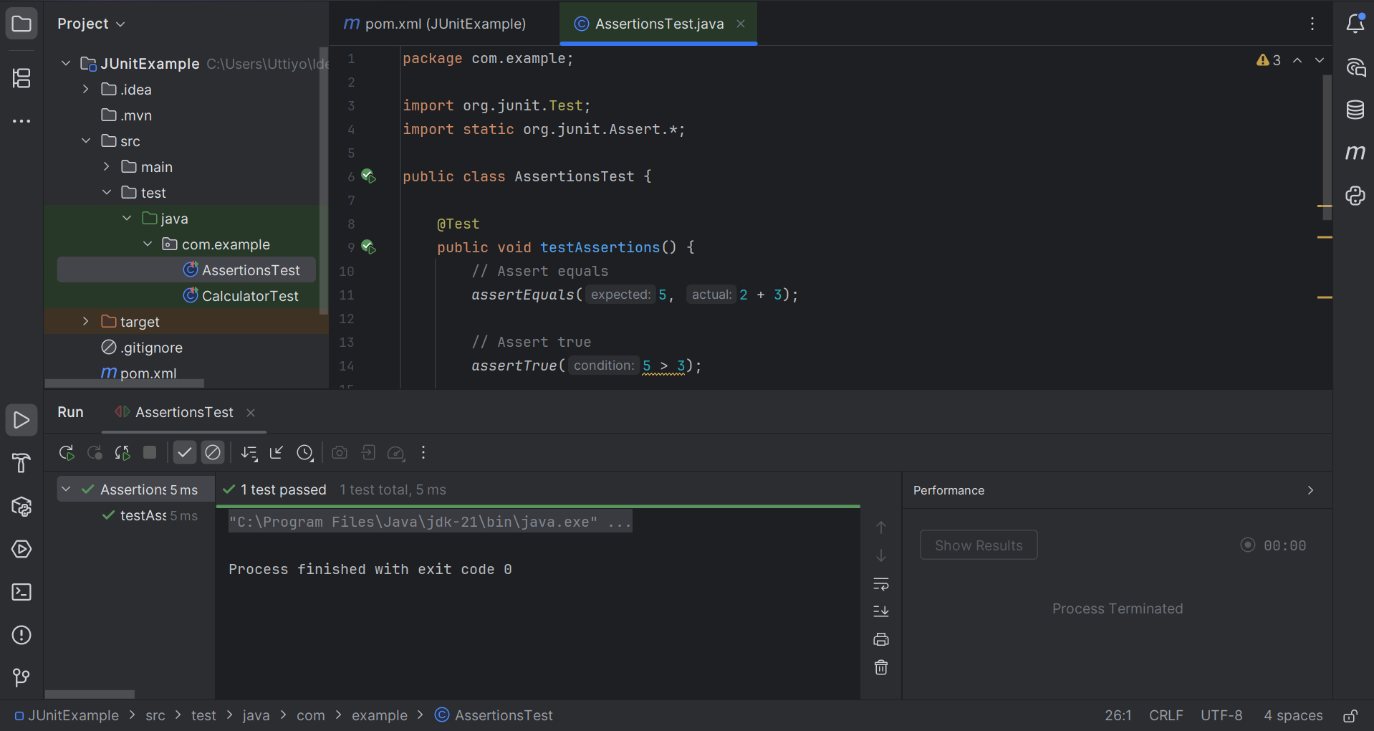
// Assert false assertFalse(5 < 3);

// Assert null assertNull(null);

// Assert not null assertNotNull(new Object());

}

}



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

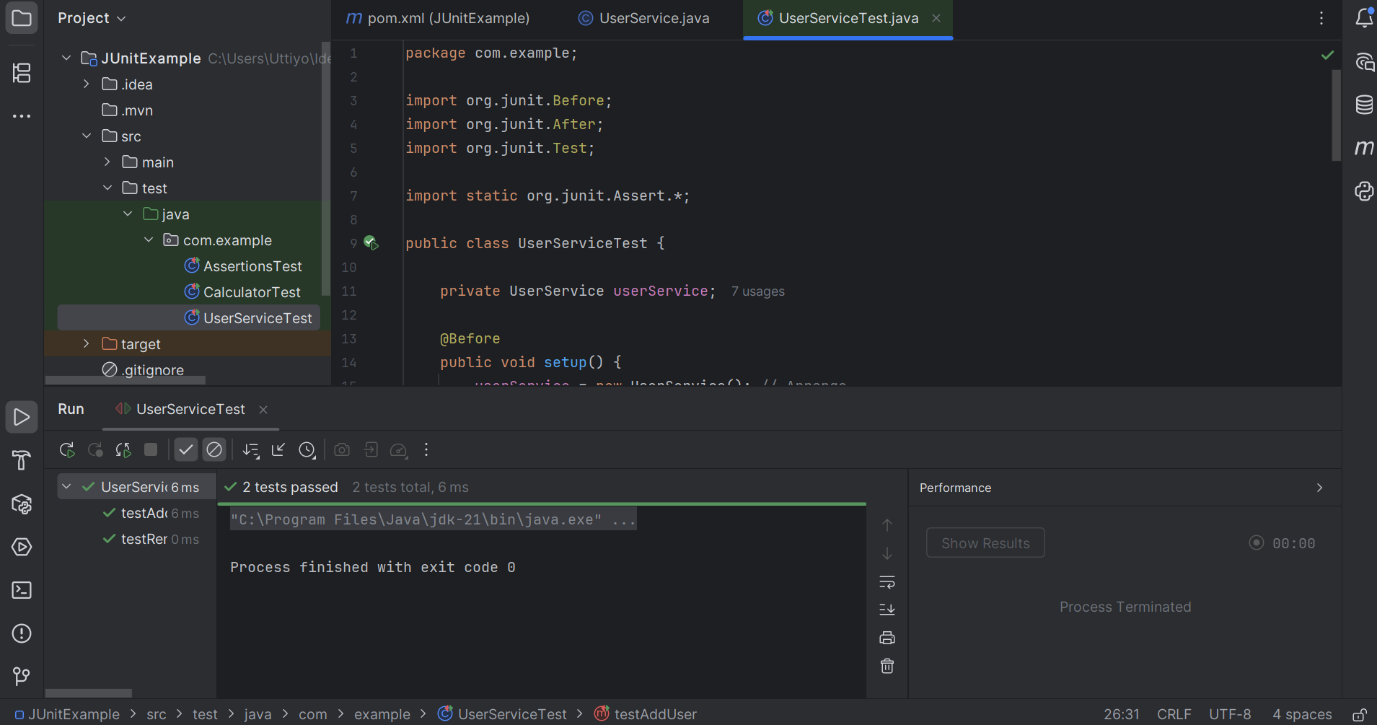
Scenario:

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.

**package com.example;  
  
import org.junit.Before;  
import org.junit.After;  
import org.junit.Test;  
  
import static org.junit.Assert.\*;  
  
public class UserServiceTest {  
  
 private UserService userService;  
  
 @Before  
 public void setup() {  
 userService = new UserService(); // Arrange  
 }  
  
 @After  
 public void teardown() {  
 userService = null; // Clean up  
 }  
  
 @Test  
 public void testAddUser() {  
 // Act  
 userService.addUser();  
  
 // Assert  
 *assertEquals*(1, userService.getUserCount());  
 }  
  
 @Test  
 public void testRemoveUser() {  
 // Arrange  
 userService.addUser();  
  
 // Act  
 userService.removeUser();  
  
 // Assert  
 *assertEquals*(0, userService.getUserCount());  
 }  
}**

****