**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: junit junit 4.13.2 test

3. Create a new test class in your project.

**Code:**

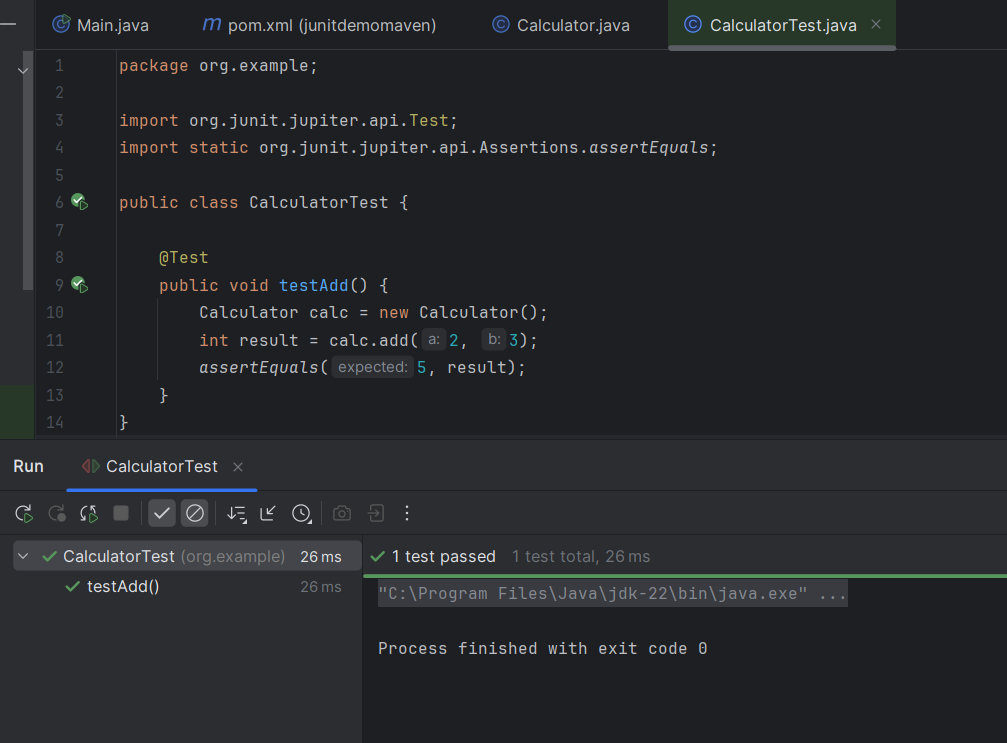
**Calculator.java**

package org.example;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}

**CalculatorTest.java**

package org.example;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class CalculatorTest {  
  
 public void testAdd() {  
 Calculator calc = new Calculator();  
 int result = calc.add(2, 3);  
 *assertEquals*(5, result);  
 }  
}

**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.

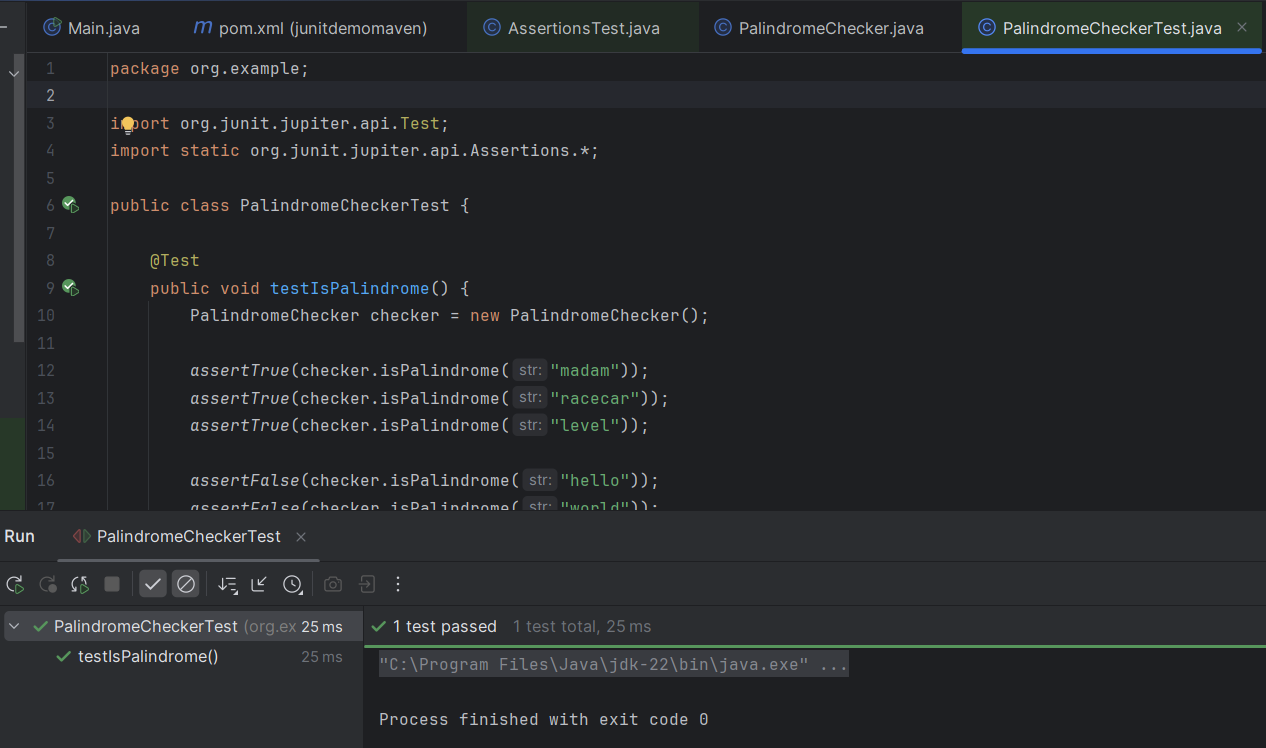
**PalindromeChecker.java**

package org.example;  
public class PalindromeChecker {  
  
 public boolean isPalindrome(String str) {  
 if (str == null) {  
 return false;  
 }  
 String reversed = new StringBuilder(str).reverse().toString();  
 return str.equals(reversed);  
 }  
}

**PalindromeCheckerTest.java**

package org.example;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class PalindromeCheckerTest {  
  
 @Test  
 public void testIsPalindrome() {  
 PalindromeChecker checker = new PalindromeChecker();  
  
 *assertTrue*(checker.isPalindrome("madam"));  
 *assertTrue*(checker.isPalindrome("racecar"));  
 *assertTrue*(checker.isPalindrome("level"));  
  
 *assertFalse*(checker.isPalindrome("hello"));  
 *assertFalse*(checker.isPalindrome("world"));  
 *assertFalse*(checker.isPalindrome(null)); // Checks null handling  
 }  
}

**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.

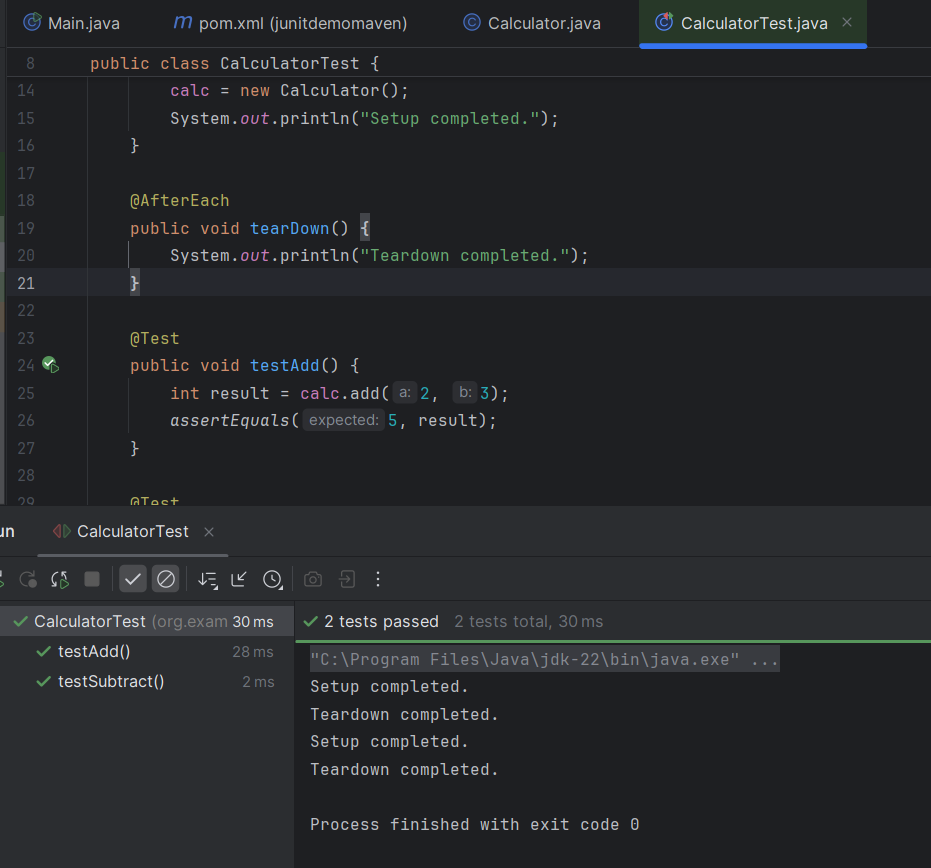
**Calculator.java**

package org.example;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
 public int subtract(int a, int b) {  
 return a - b;  
 }  
}

**CalculatorTest.java:**

package org.example;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.AfterEach;  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class CalculatorTest {  
  
 private Calculator calc;  
  
   
 public void setUp() {  
 calc = new Calculator();  
 System.*out*.println("Setup completed.");  
 }  
  
 public void tearDown() {  
 System.*out*.println("Teardown completed.");  
 }  
  
 public void testAdd() {  
 int result = calc.add(2, 3);  
 *assertEquals*(5, result);  
 }  
  
 public void testSubtract() {  
 int result = calc.subtract(5, 2);  
 *assertEquals*(3, result);  
 }  
}

**Output:**

****