**Week2\_SettingUpJunit**

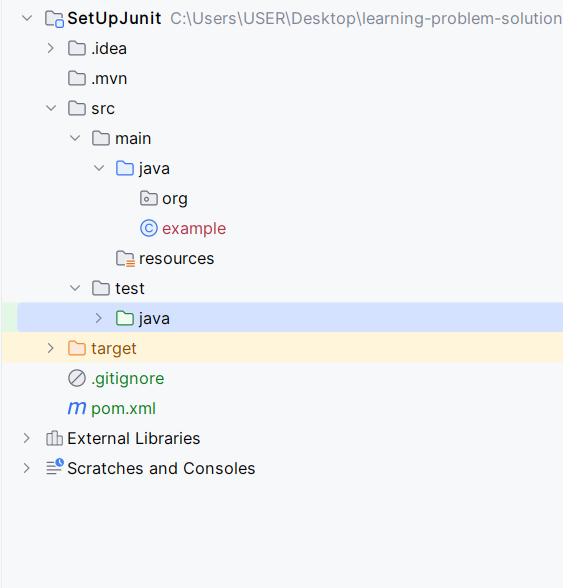
## Exercise 1: Setting Up JUnit

**Scenario:**

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

**Steps:**

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



**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>**

*<?*xml version="1.0" encoding="UTF-8"*?>*<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>SetUpJunit</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>24</maven.compiler.source>  
 <maven.compiler.target>24</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
</project>

**Create a new test class in your project.**

public class CalculatorTest {  
}

**Week2\_Assertions in JUnit**

## Exercise 3: Assertions in JUnit

**Scenario:**

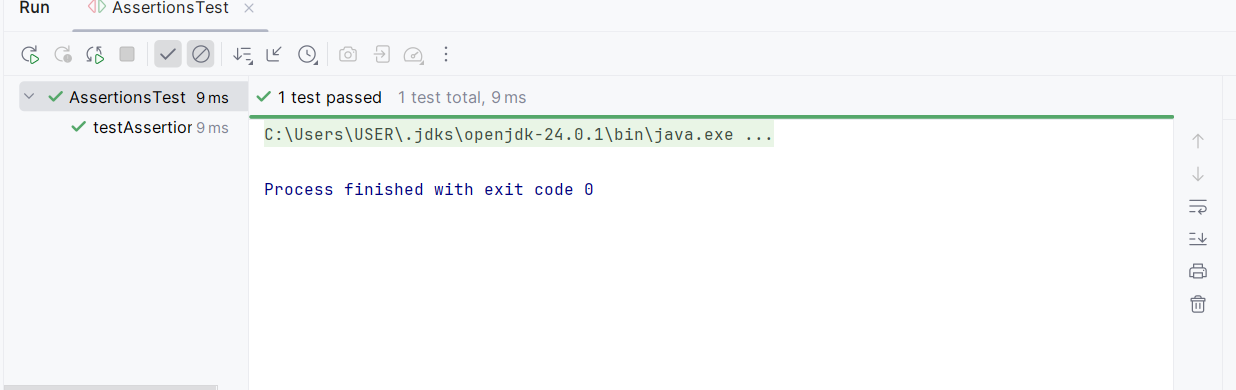
**You need to use different assertions in JUnit to validate your test results.**

**Steps:**

**Write tests using various JUnit assertions.**

// Import static methods to make the code cleaner  
import static org.junit.Assert.\*;  
import org.junit.Test;  
  
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());  
 }  
}

Output



**Week2\_AAA Pattern & Teardown Methods**

## 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:**

### --**Create a Simple Class to Test**

*// In src/main/java/example/Calculator.java*package example;  
  
public class Calculator {  
 private int currentValue = 0;  
  
 public void add(int number) {  
 this.currentValue += number;  
 }  
  
 public void subtract(int number) {  
 this.currentValue -= number;  
 }  
  
 public int getCurrentValue() {  
 return this.currentValue;  
 }  
  
 public void reset() {  
 this.currentValue = 0;  
 }  
}

**Write tests using the AAA pattern.**

@Test  
public void testAddition() {  
 *// Arrange: The setup is done by the @Before method. No further arrangement is needed.* System.*out*.println("Executing testAddition...");  
  
 *// Act: Perform the action to be tested.* calculator.add(5);  
  
 *// Assert: Check if the result is as expected.* assertEquals(5, calculator.getCurrentValue());  
}  
  
@Test  
public void testSubtraction() {  
 *// Arrange: The setup is done by the @Before method.* System.*out*.println("Executing testSubtraction...");  
  
 *// Act: Perform the action to be tested.* calculator.subtract(3);  
  
 *// Assert: Check if the result is as expected.* assertEquals(-3, calculator.getCurrentValue());  
}

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

*// In src/test/java/example/CalculatorTest.java*package example;  
  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class CalculatorTest {  
  
 private Calculator calculator; *// This is our test fixture* @Before  
 public void setUp() {  
 *// This method runs BEFORE each test.* System.*out*.println("Running @Before: Setting up a new Calculator instance.");  
 calculator = new Calculator();  
 }  
  
 @After  
 public void tearDown() {  
 *// This method runs AFTER each test.* System.*out*.println("Running @After: Tearing down the instance.");  
 calculator = null; *// Clean up the object to ensure no state leaks between tests.* }  
  
 @Test  
 public void testAddition() {  
 *// Arrange: The setup is done by the @Before method. No further arrangement is needed.* System.*out*.println("Executing testAddition...");  
  
 *// Act: Perform the action to be tested.* calculator.add(5);  
  
 *// Assert: Check if the result is as expected.* assertEquals(5, calculator.getCurrentValue());  
 }  
  
 @Test  
 public void testSubtraction() {  
 *// Arrange: The setup is done by the @Before method.* System.*out*.println("Executing testSubtraction...");  
  
 *// Act: Perform the action to be tested.* calculator.subtract(3);  
  
 *// Assert: Check if the result is as expected.* assertEquals(-3, calculator.getCurrentValue());  
 }}

Output

