**Exercise 1: Setting Up JUnit**

Sparshak Ghosh

Mandatory Hands-on

**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>**

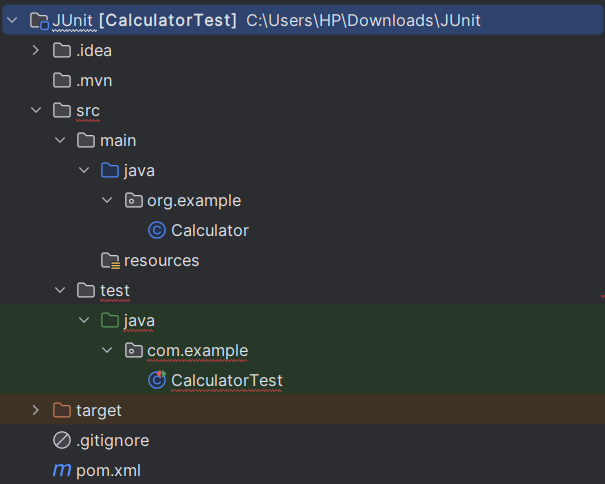
**<artifactId>junit</artifactId>**

**<version>4.13.2</version>**

**<scope>test</scope>**

**</dependency>**

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

1.

2. **pom.xml**

<?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>CalculatorTest</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</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>

Sparshak Ghosh

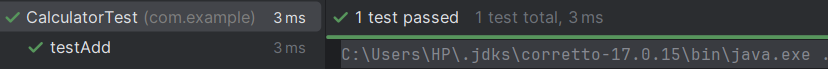
Mandatory Hands-on

3. **Calculator.java**

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

**CalculatorTest.java**

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



**Exercise 3: Assertions in JUnit**

Mandatory Hands-on

Sparshak Ghosh

**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() {**

**assertEquals(5, 2 + 3);**

**assertTrue(5 > 3);**

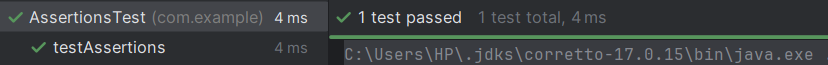
**assertFalse(5 < 3);**

**assertNull(null);**

**assertNotNull(new Object());**

**}**

**}**



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

Mandatory Hands-on

Sparshak Ghosh

**Scenario:**

**You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup**

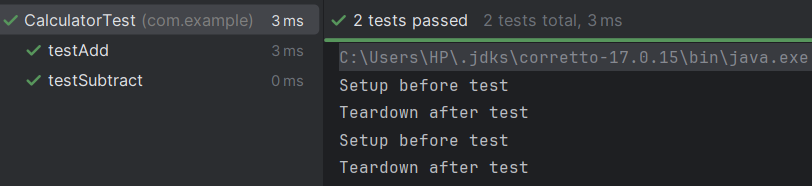
**and teardown methods.**

**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.After;  
import org.junit.Before;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
public class CalculatorTest {  
 private Calculator calculator;  
 @Before  
 public void setUp() {  
 calculator = new Calculator();  
 System.*out*.println("Setup before test");  
 }  
 @After  
 public void tearDown() {  
 calculator = null;  
 System.*out*.println("Teardown after test");  
 }  
 @Test  
 public void testAdd() {  
 int result = calculator.add(10, 5);  
 *assertEquals*(15, result);  
 }  
 @Test  
 public void testSubtract() {  
 int result = calculator.subtract(10, 5);  
 *assertEquals*(5, result);  
 }  
}

****