**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.

**SOLUTION:**

Adding the dependency in 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>JUnitSetupDemo</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>  
 <!-- JUnit 4 Dependency -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
  
</project>

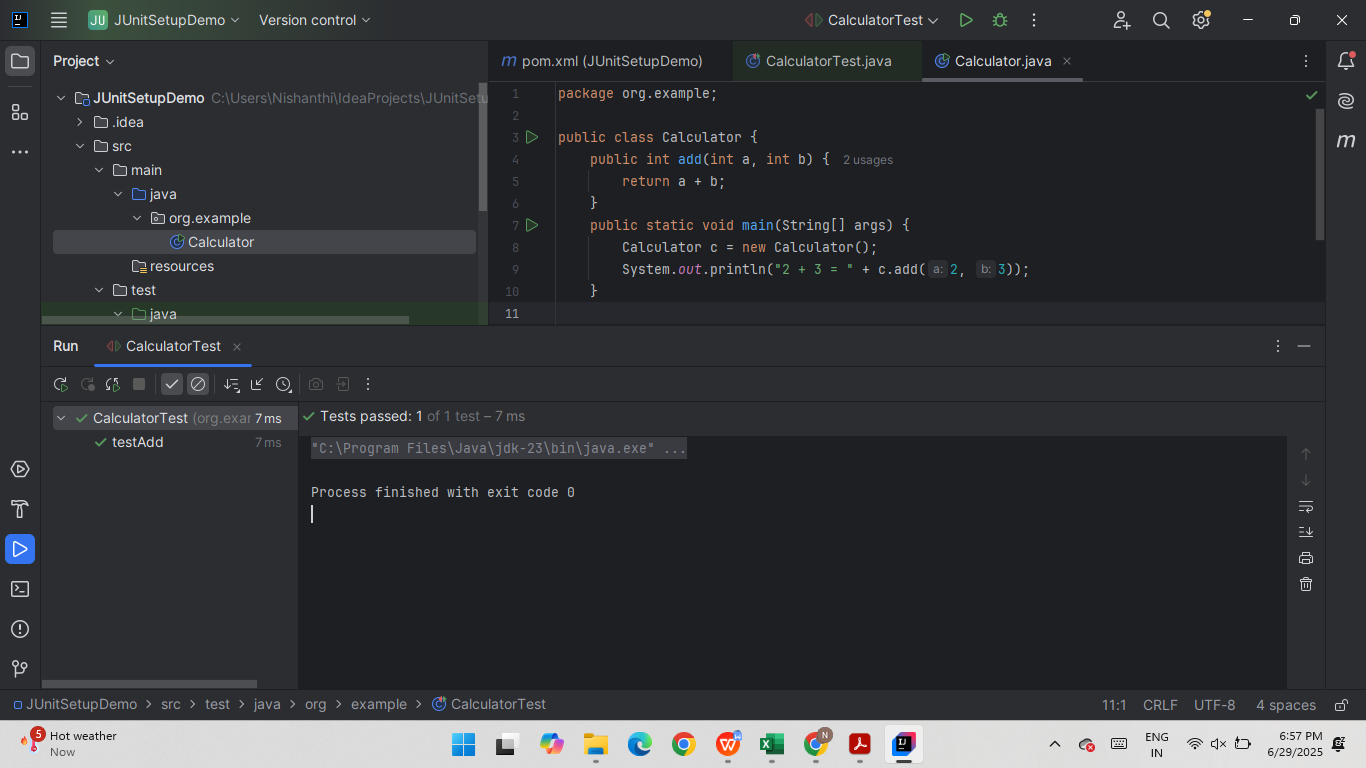
**TEST CASE:**

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

**MAIN CLASS:**

package org.example;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
 public static void main(String[] args) {  
 Calculator c = new Calculator();  
 System.*out*.println("2 + 3 = " + c.add(2, 3));  
 }  
  
}

OUTPUT



**Exercise 3: Assertions in JUnit**

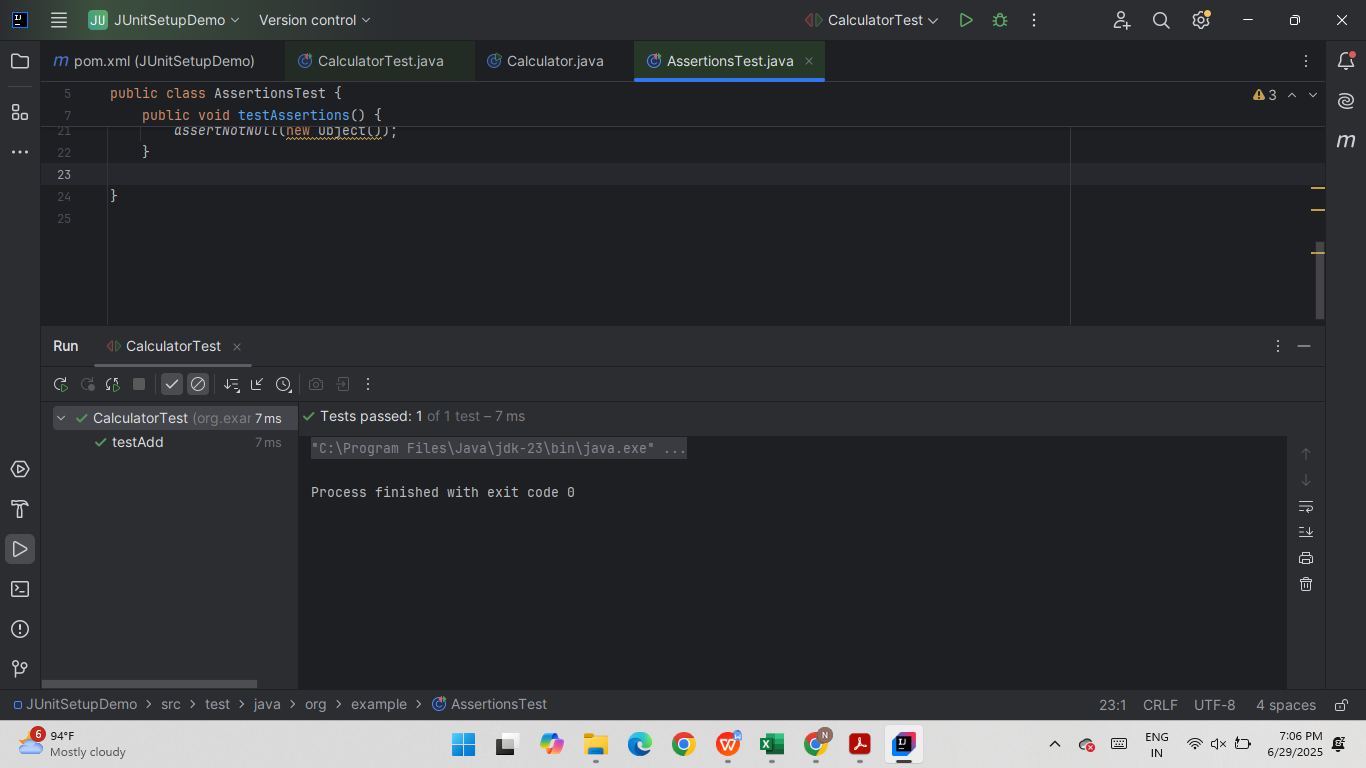
**Scenario:**

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

**SOLUTIONS:**

package org.example;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
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



**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:**

**Arrange** – Set up objects and data.

**Act** – Call the method being tested.

**Assert** – Verify the result.

CalculatorTest.java

package org.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: Creating Calculator object");  
 }  
  
 @After  
 public void tearDown() {  
 calculator = null;  
 System.*out*.println("Teardown: Cleaning up");  
 }  
  
 @Test  
 public void testAdd() {  
 int a = 10;  
 int b = 5;  
  
 // Act  
 int result = calculator.add(a, b);  
  
 // Assert  
 *assertEquals*(15, result);  
 }  
  
 @Test  
 public void testSubtract() {  
 // Arrange  
 int a = 10;  
 int b = 5;  
  
 // Act  
 int result = calculator.subtract(a, b);  
  
 // Assert  
 *assertEquals*(5, result);  
 }  
}

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;  
 }  
}

OUTPUT:

