**Exercise 1: Setting Up JUnit**

<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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitExample</artifactId>

<packaging>jar</packaging>

<version>1.0-SNAPSHOT</version>

<name>JUnitExample</name>

<url>http://maven.apache.org</url>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

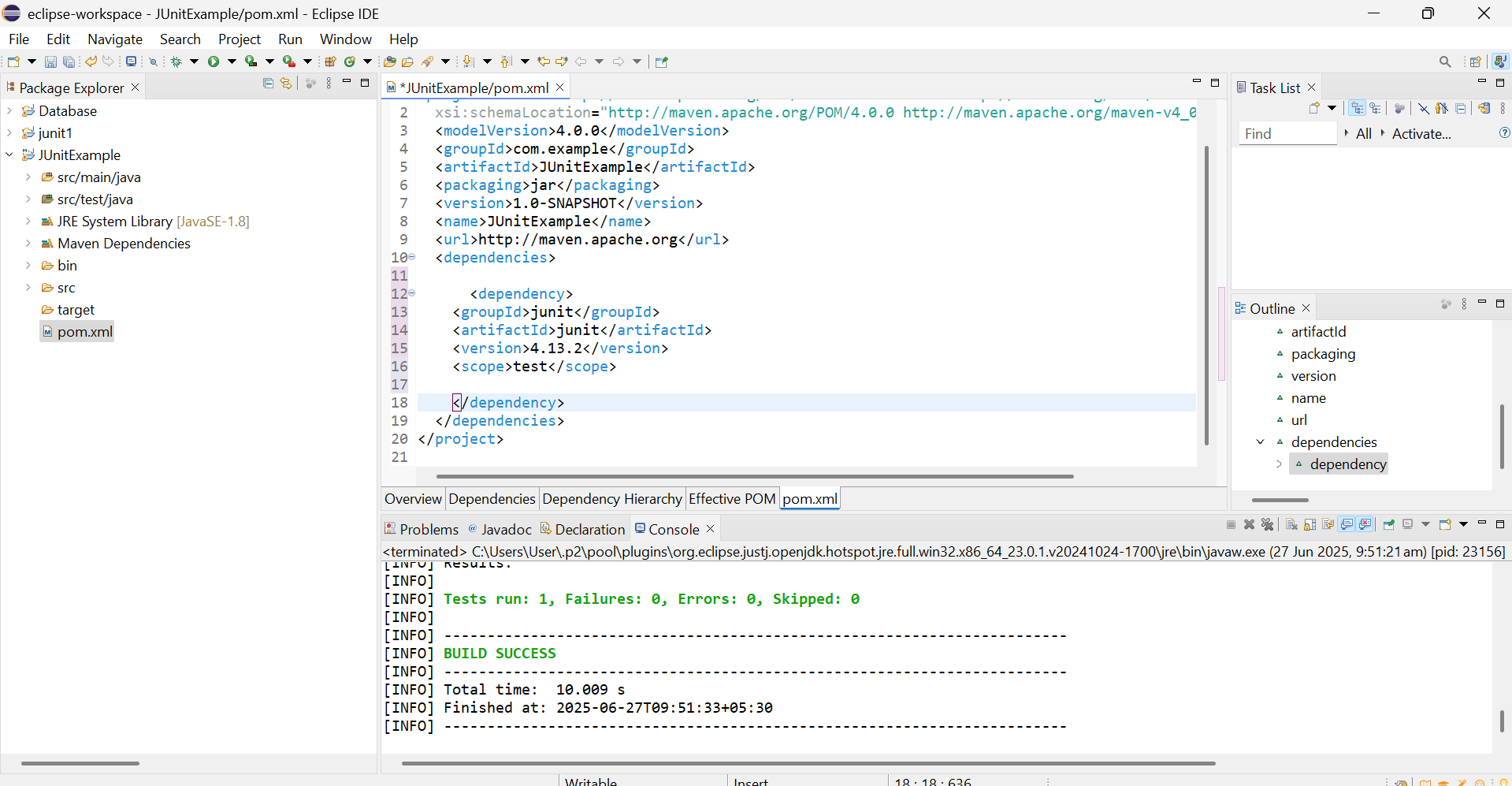
<scope>test</scope>

</dependency>

</dependencies>

</project>

**Output**

****

**Exercise 2: Writing Basic JUnit Tests**

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) { return a + b; }

public int subtract(int a, int b) { return a - b; }

public int multiply(int a, int b) { return a \* b; }

public int divide(int a, int b) {

if (b == 0) throw new IllegalArgumentException("Cannot divide by zero");

return a / b;

}

}

**CalculatorTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

Calculator calc = new Calculator();

@Test

public void testAdd() {

assertEquals(5, calc.add(2, 3));

}

@Test

public void testSubtract() {

assertEquals(1, calc.subtract(3, 2));

}

@Test

public void testMultiply() {

assertEquals(6, calc.multiply(2, 3));

}

@Test

public void testDivide() {

assertEquals(2, calc.divide(6, 3));

}

@Test(expected = IllegalArgumentException.class)

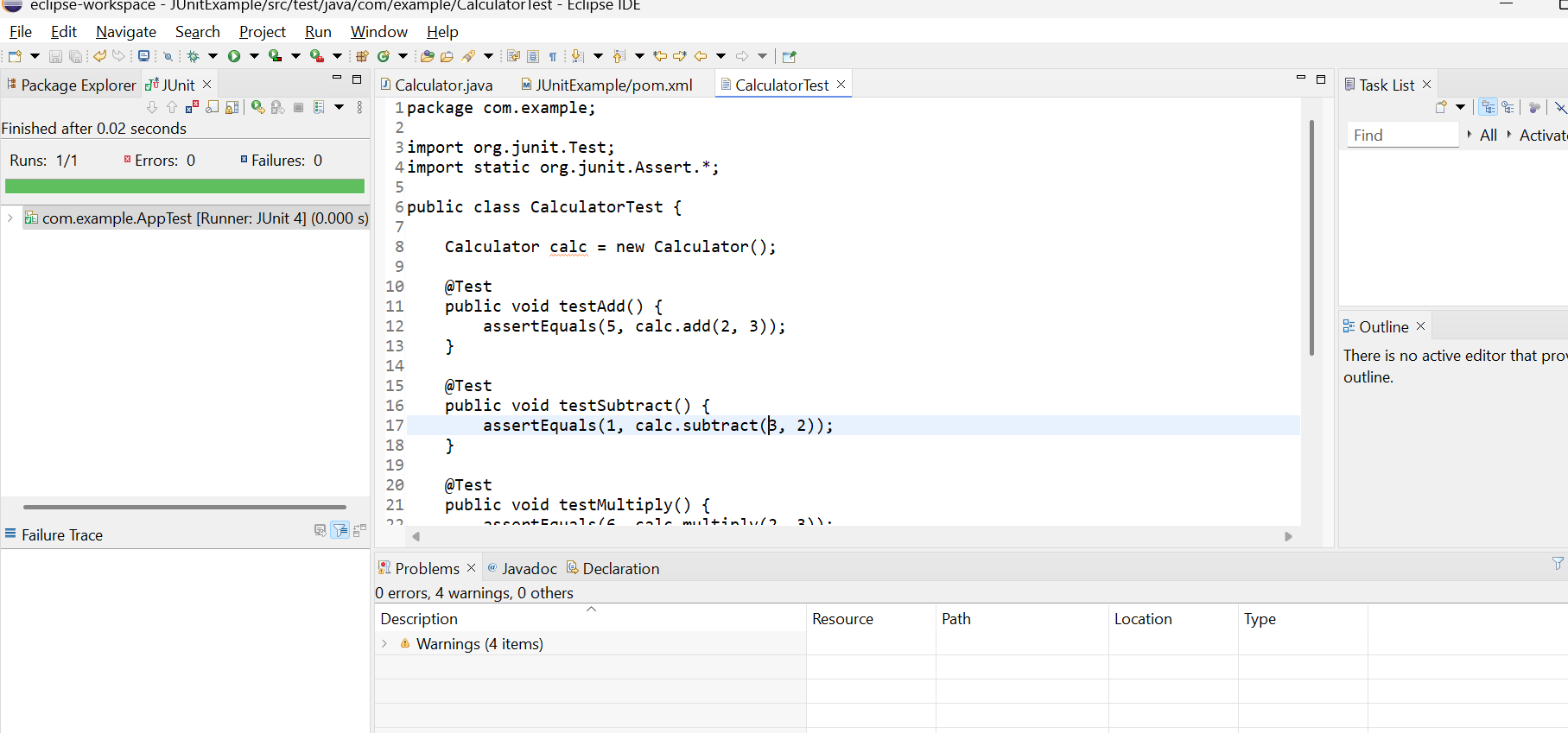
public void testDivideByZero() {

calc.divide(10, 0);

}

}

Output:



**Exercise 3: Assertions in JUnit**

**AssertionsTest.java**

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

// Check equality

assertEquals(5, 2 + 3);

// Check condition is true

assertTrue(5 > 3);

// Check condition is false

assertFalse(5 < 3);

// Check object is null

assertNull(null);

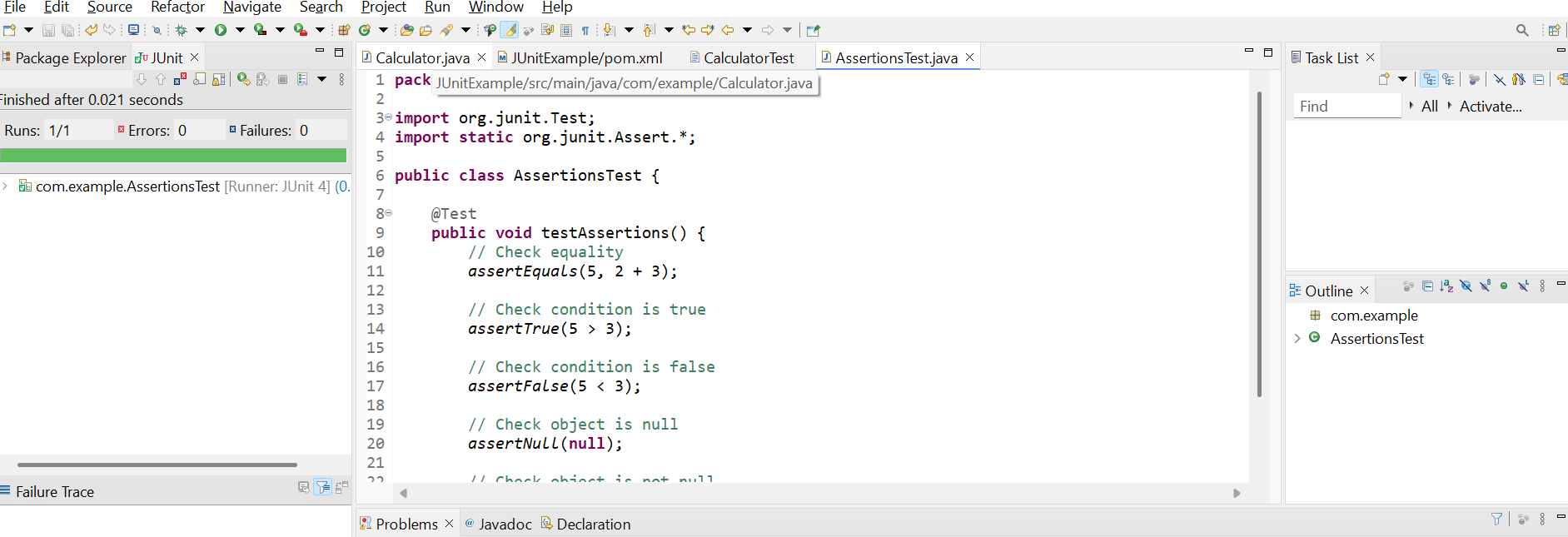
// Check object is not null

assertNotNull(new Object());

}

}

**Output:**

****

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

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) { return a + b; }

public int subtract(int a, int b) { return a - b; }

}

**CalculatorTestWithSetup.java**

package com.example;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTestWithSetup {

private Calculator calc;

// Setup method - runs before each test

@Before

public void setUp() {

System.out.println("Setting up...");

calc = new Calculator(); // Arrange

}

// Teardown method - runs after each test

@After

public void tearDown() {

System.out.println("Cleaning up...");

calc = null;

}

@Test

public void testAdd() {

// Act

int result = calc.add(3, 2);

// Assert

assertEquals(5, result);

}

@Test

public void testSubtract() {

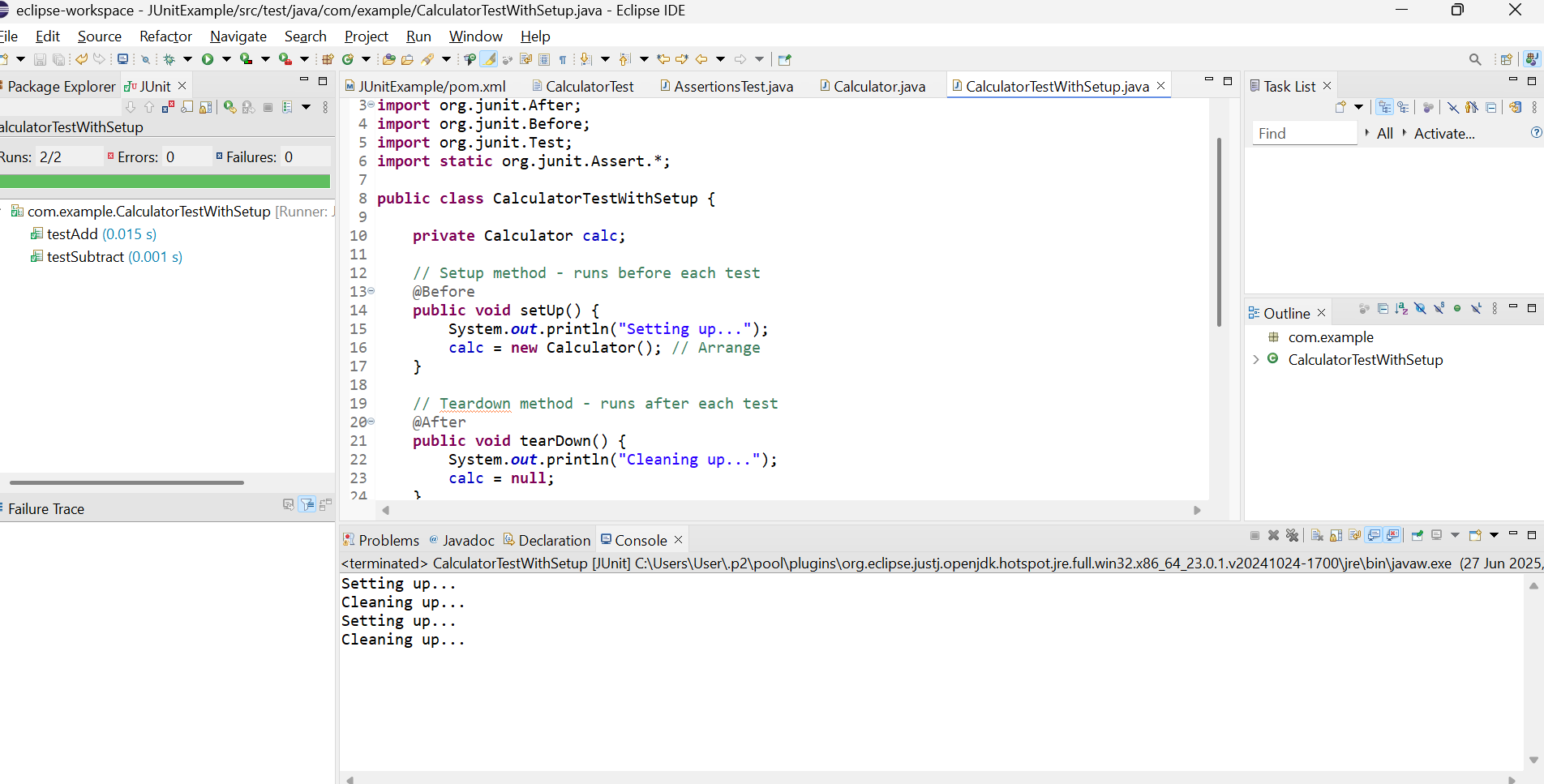
int result = calc.subtract(10, 4);

assertEquals(6, result);

}

}

**Output:**

****