# TDD using JUnit5 and Mockito

# JUnit\_Basic Testing Exercises

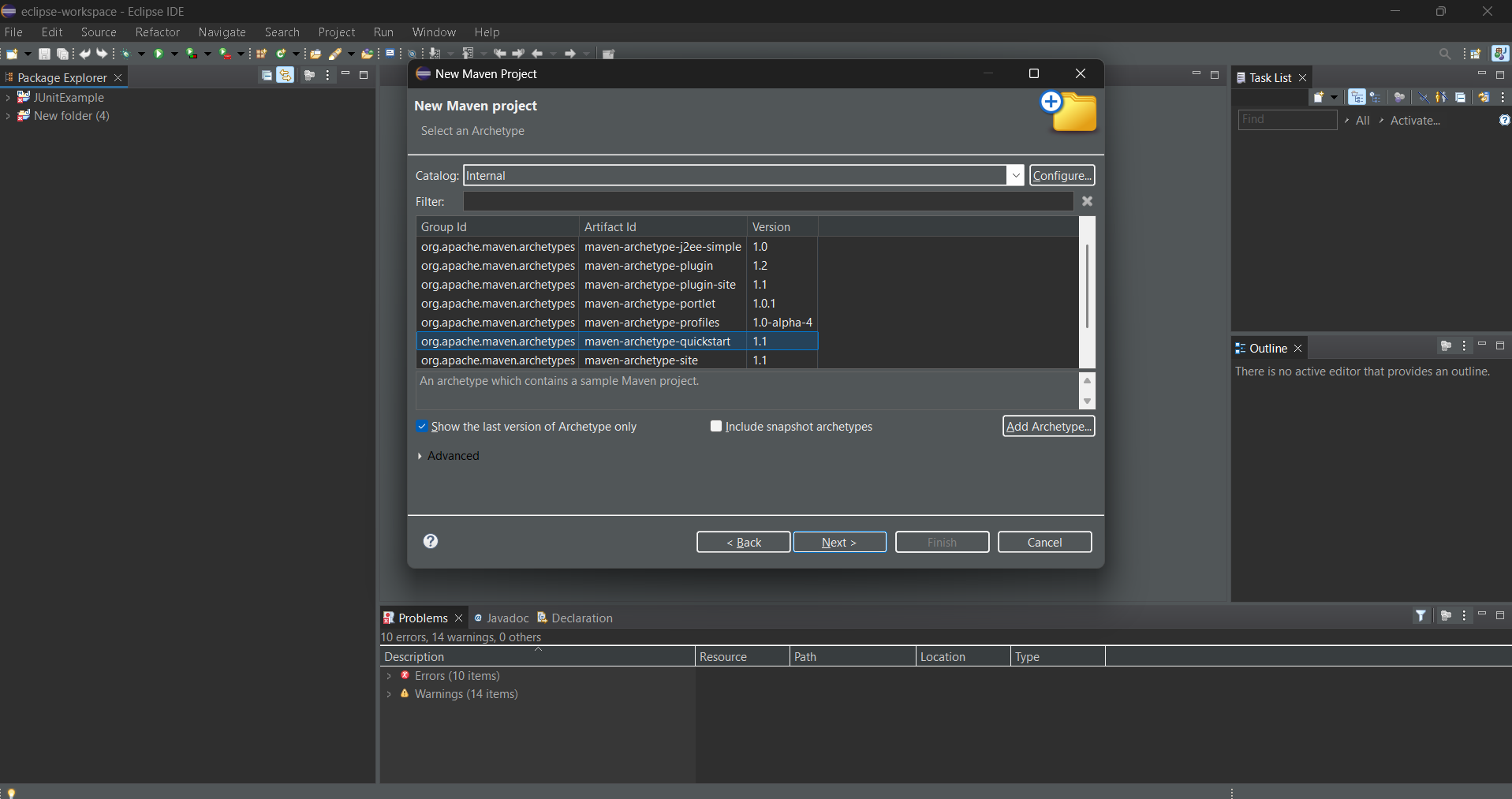
**Exercise 1: Setting Up JUnit**

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 dependency to your pom.xml .

<dependency>

<groupId>junit</groupId>

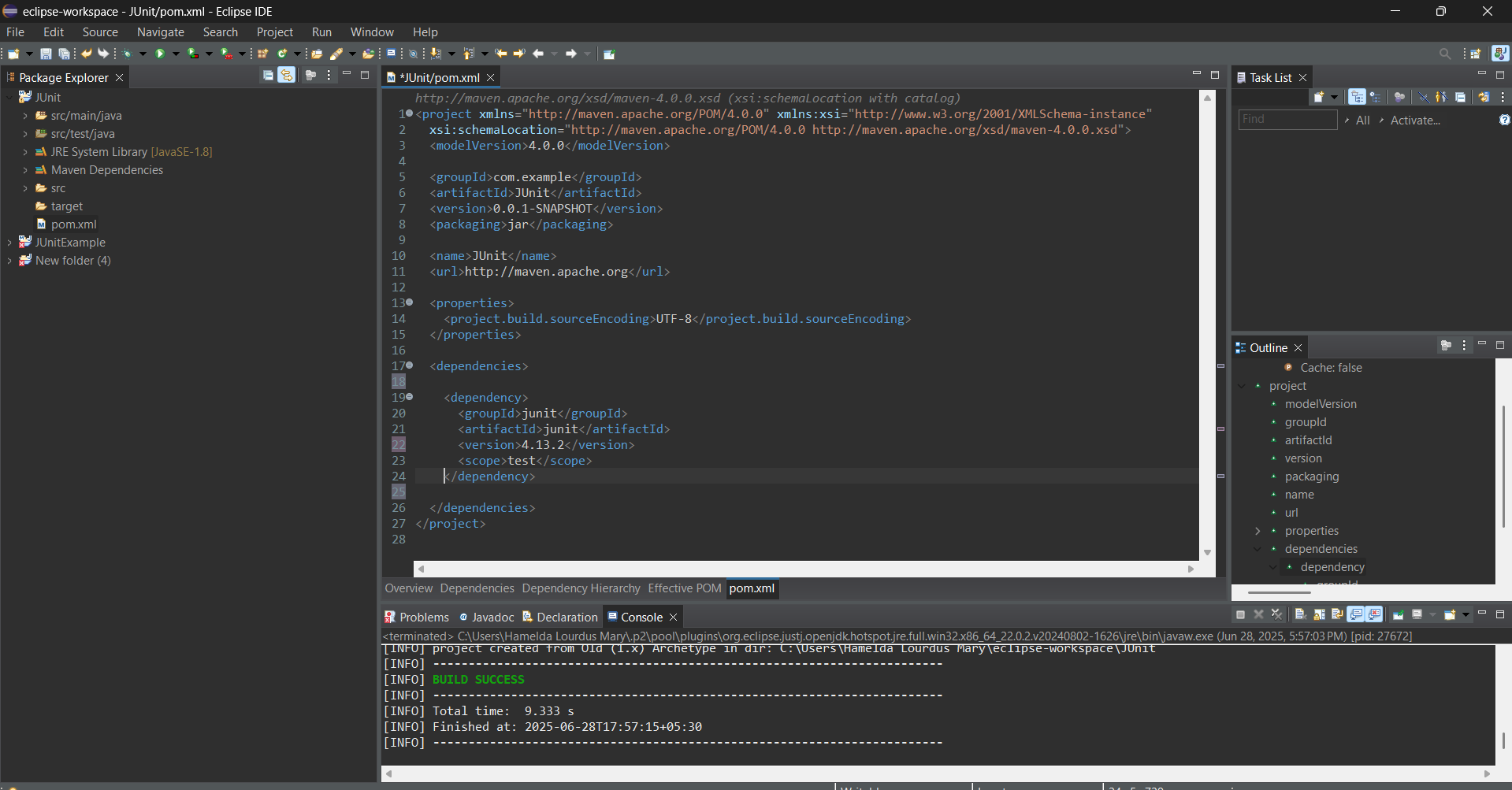
<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**Pom.xml**



3.Create a new test class in your project.

***Create a class- src/main/java/example/Calculator.java***

package 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;

}

}

***Create a class- src/test/java/example/CalculatorTest.java***

package example;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class CalculatorTest {

private Calculator calc;

// Setup method

@Before

public void setUp() {

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

calc = new Calculator();

}

// Teardown method

@After

public void tearDown() {

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

calc = null;

}

@Test

public void testAdd() {

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

}

@Test

public void testSubtract() {

assertEquals(1, calc.subtract(4, 3));

}

@Test

public void testMultiply() {

assertEquals(12, calc.multiply(3, 4));

}

@Test

public void testDivide() {

assertEquals(2, calc.divide(10, 5));

}

@Test(expected = IllegalArgumentException.class)

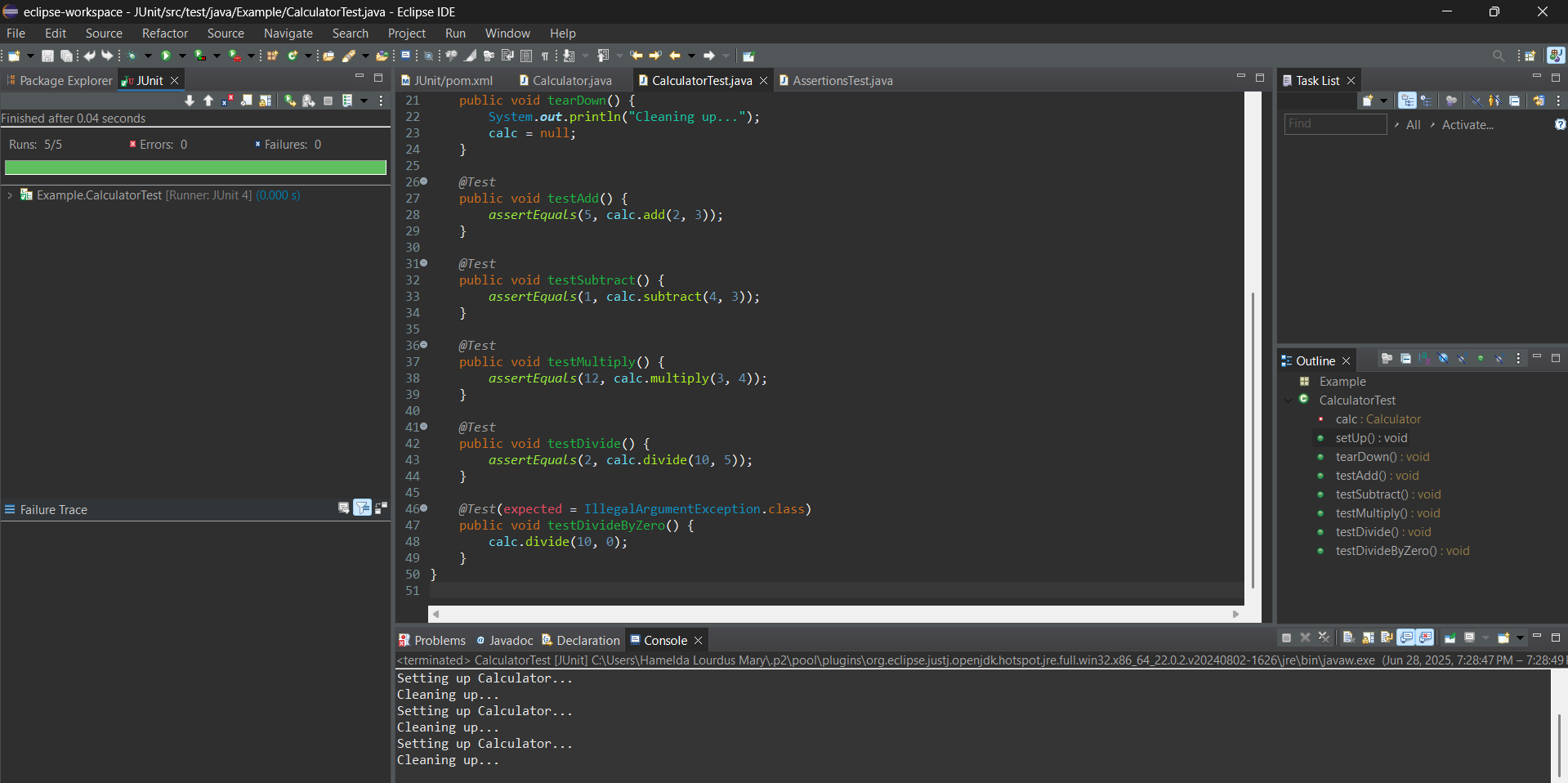
public void testDivideByZero() {

calc.divide(10, 0);

}

}

**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.

***Create a class - src/test/java/example/AssertionTest***

package example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testMixedAssertions() {

// assertEquals

assertEquals("Sum should be 5", 5, 2 + 3);

// assertTrue: condition must be true

assertTrue("Should be true because 10 is greater than 1", 10 > 1);

// assertFalse: condition must be false

assertFalse("Should be false because 2 is not greater than 5", 2 > 5);

// assertNull: object must be null

String str = null;

assertNull("Expected string to be null", str);

// assertNotNull: object must not be null

Object obj = new Object();

assertNotNull("Expected object to be not null", obj);

// assertArrayEquals: compare two arrays

int[] expected = {1, 2, 3};

int[] actual = {1, 2, 3};

assertArrayEquals("Arrays should be equal", expected, actual);

// assertSame: compare references

String name = "JUnit";

String sameRef = name;

assertSame("Both references should point to the same object", name, sameRef);

// assertNotSame: different references

String s1 = new String("Test");

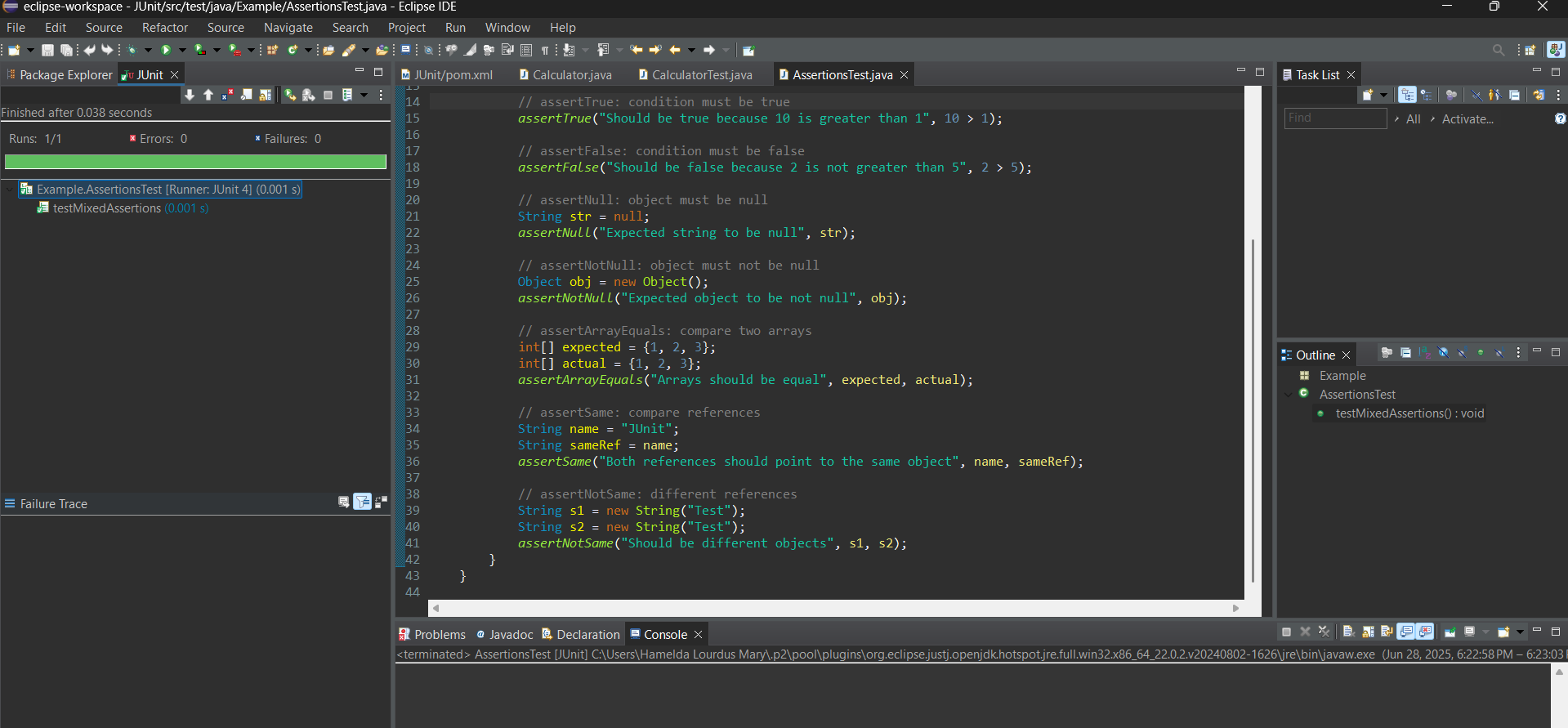
String s2 = new String("Test");

assertNotSame("Should be different objects", s1, s2);

}

}

**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.

***Create a class- src/main/java/example/Calculator.java***

package example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

***Create a class- 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;

*//SetUp method*

@Before

public void setUp() {

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

calculator = new Calculator(); // Arrange

}

*//Tear Down method*

@After

public void tearDown() {

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

calculator = null;

}

@Test

public void testAddition() {

// Arrange: Done in setUp()

// Act

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

// Assert

assertEquals("2 + 3 should equal 5", 5, result);

}

@Test

public void testSubtraction() {

// Arrange: Done in setUp()

// Act

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

// Assert

assertEquals("10 - 4 should equal 6", 6, result);

}

}

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

//SetUp method

@Before

public void setUp() {

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

calculator = new Calculator(); // Arrange

}

//Tear Down method

@After

public void tearDown() {

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

calculator = null;

}

**Output:**

