**Cognizant Digital Nurture 4.0**

**Name: Siri Chandana Chittipolu**

**Email:** [**sirichittipolu11@gmail.com**](mailto:sirichittipolu11@gmail.com)

**Superset ID: 6386277**

**Mandatory Hands-On Exercises**

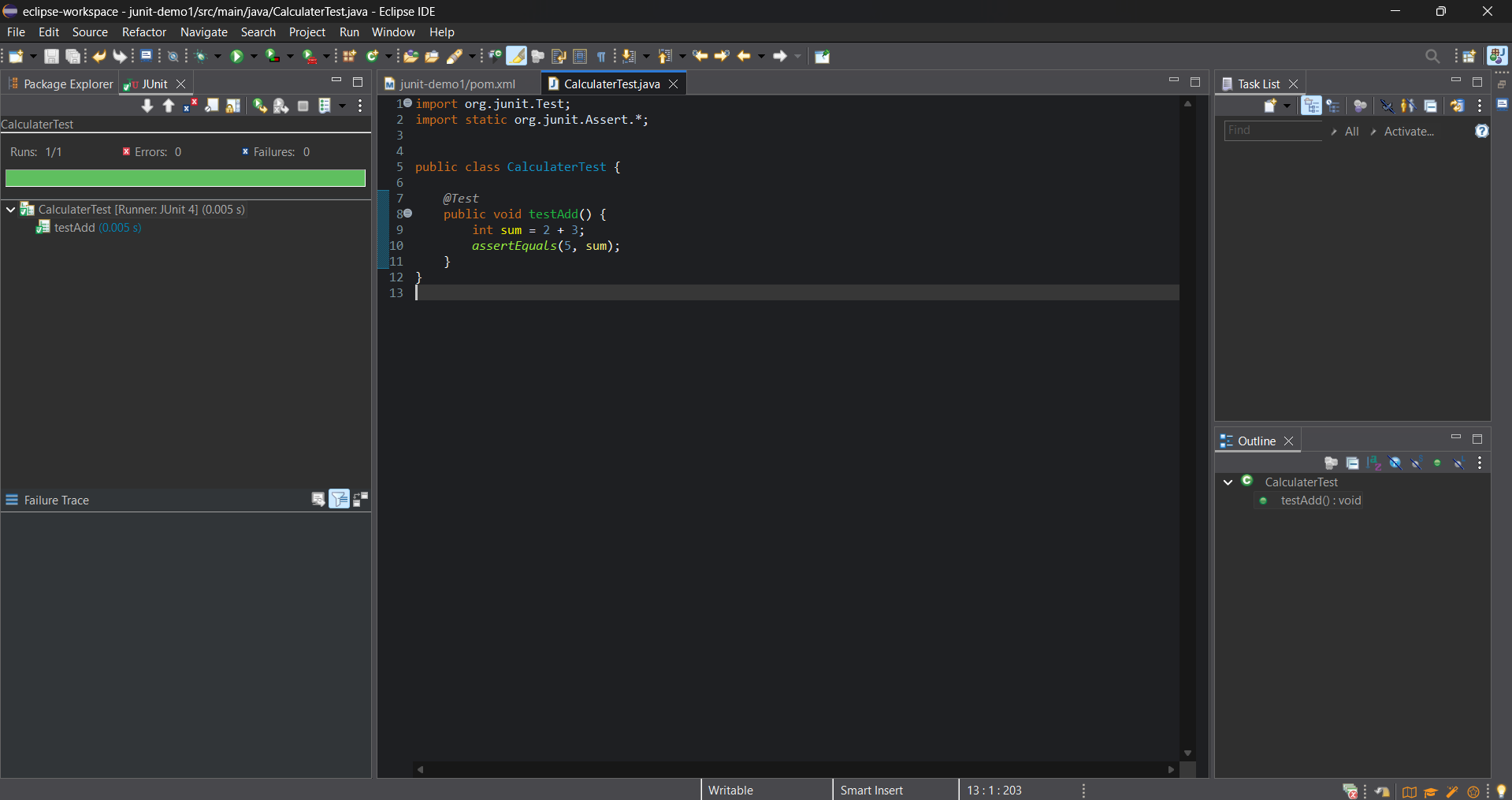
**JUnit Testing Exercises:**

**Exercise 1: Setting Up JUnit**

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

**Solution:**

* After setting up the Java project in Eclipse and adding the required JUnit dependency to my pom.xml file, I created a simple test class named CalculatorTest.
* In this class, I wrote a unit test method using the JUnit framework to validate a basic addition operation.
* I successfully ran the test using Eclipse’s JUnit test runner, and the test passed, confirming that the addition logic works as expected.

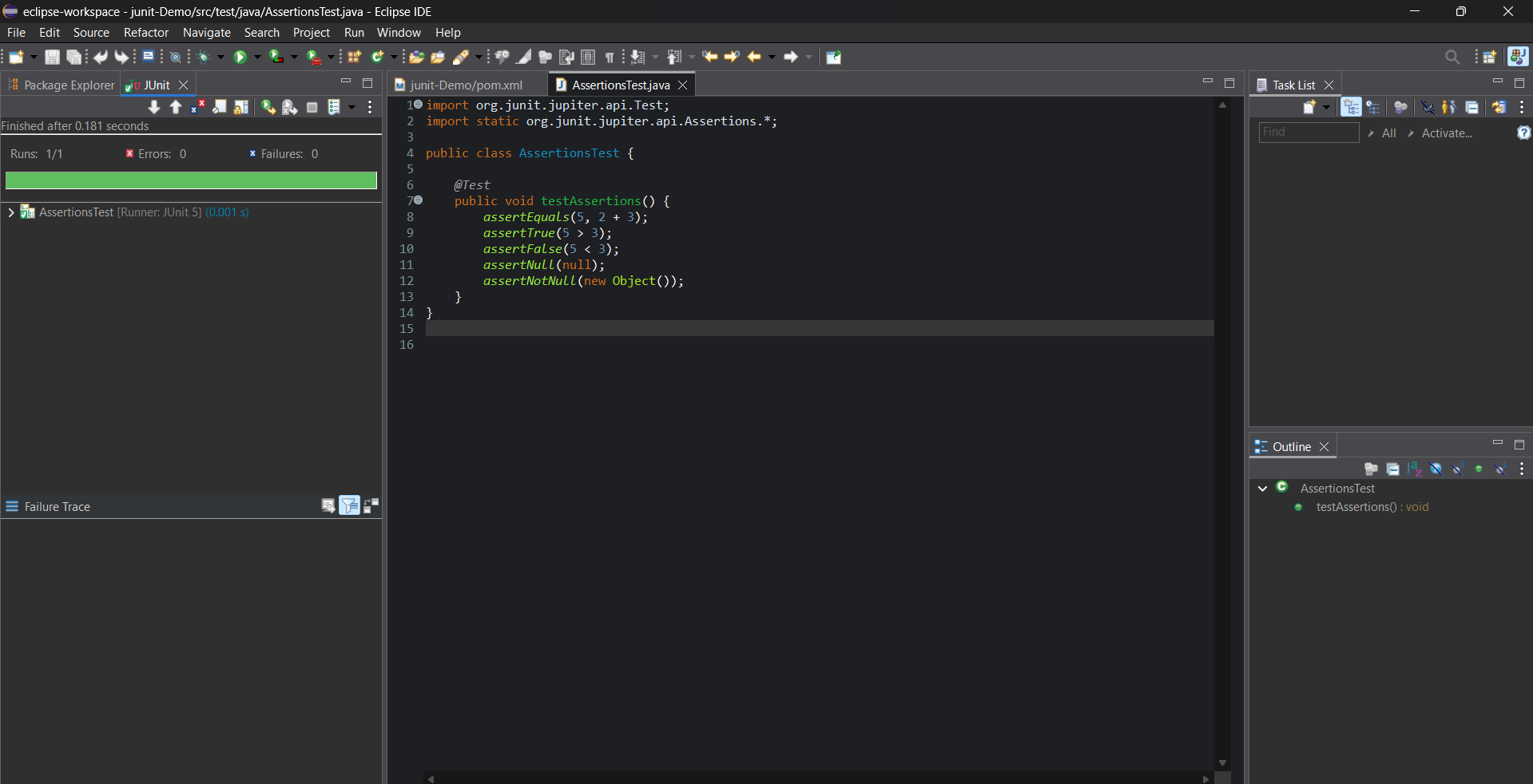


**Exercise 3: Assertions in JUnit**

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

**Solution:**

* After setting up the Maven project and updating the pom.xml with JUnit dependencies, I created a test class with various assertions using @Test from org.junit.jupiter.api.
* I placed the class inside src/test/java, refreshed the project, and successfully ran the tests using the JUnit runner in Eclipse.



**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.**

**Solution:**

**Calculator.java:**

package com.example.junit\_Demo;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

}

CalculatroTest.java :

package com.example.junit\_Demo;

import org.junit.jupiter.api.\*;

import static org.junit.jupiter.api.Assertions.\*;

public class CalculatorTest {

private Calculator calculator;

*@BeforeEach*

void setUp() {

calculator = new Calculator();

System.***out***.println("Setup: Calculator created");

}

*@AfterEach*

void tearDown() {

calculator = null;

System.***out***.println("Teardown: Calculator cleared");

}

*@Test*

void testAddition() {

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

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

}

*@Test*

void testMultiplication() {

int result = calculator.multiply(4, 2);

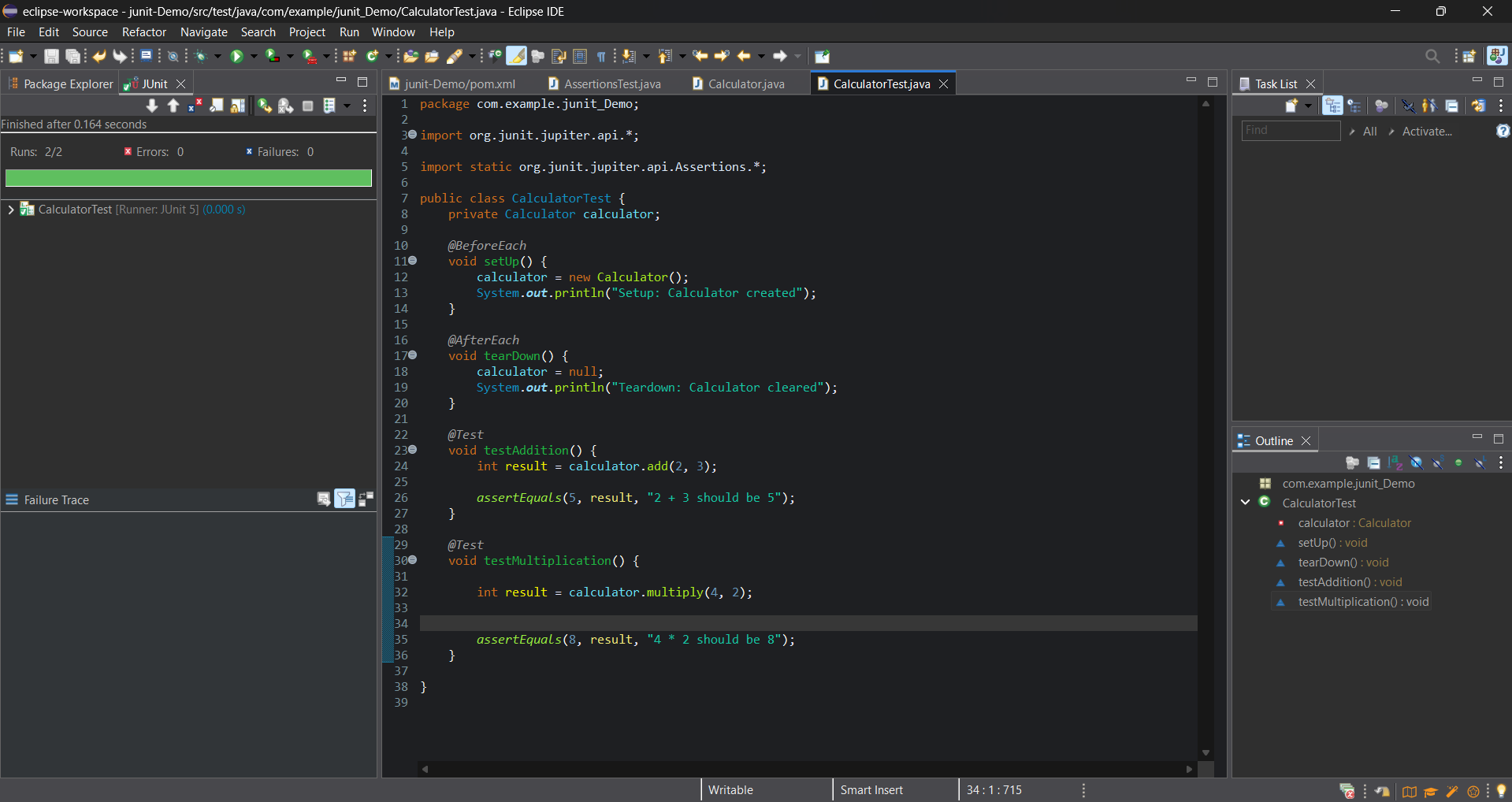
*assertEquals*(8, result, "4 \* 2 should be 8");

}

}

* I followed the Arrange-Act-Assert (AAA) pattern by creating a Calculator class and organizing the test logic accordingly.
* Using @BeforeEach and @AfterEach, I implemented setup and teardown methods to initialize and clean up resources before and after each test.

**Output:**

****