JUnit Testing Exercises

Exercise 1: Setting Up Junit

Exercise 1: Setting Up Junit(Mandatory)

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).

package com.example;

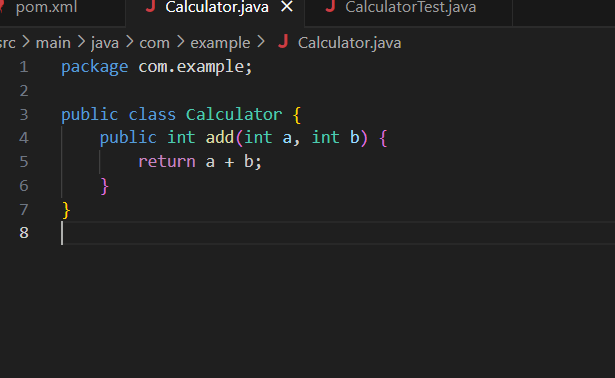
public class Calculator {

public int add(int a, int b) {

return a + b;

}

}



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.

package com.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class CalculatorTest {

@Test

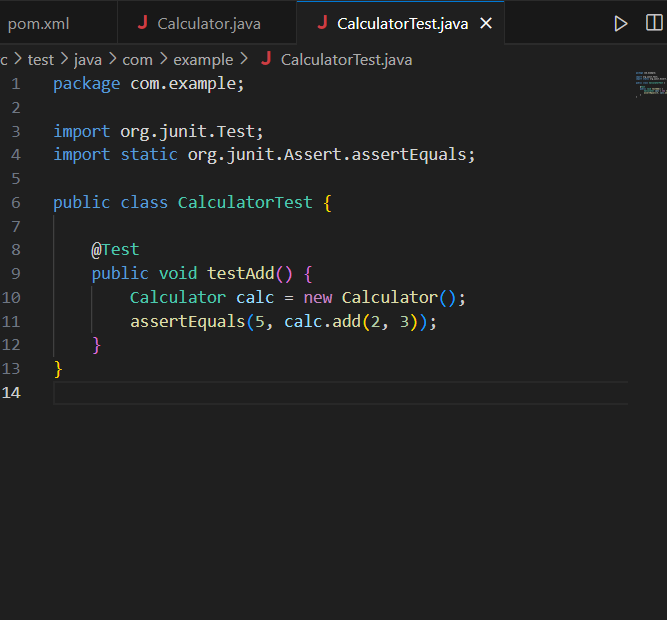
void testAdd() {

Calculator calc = new Calculator();

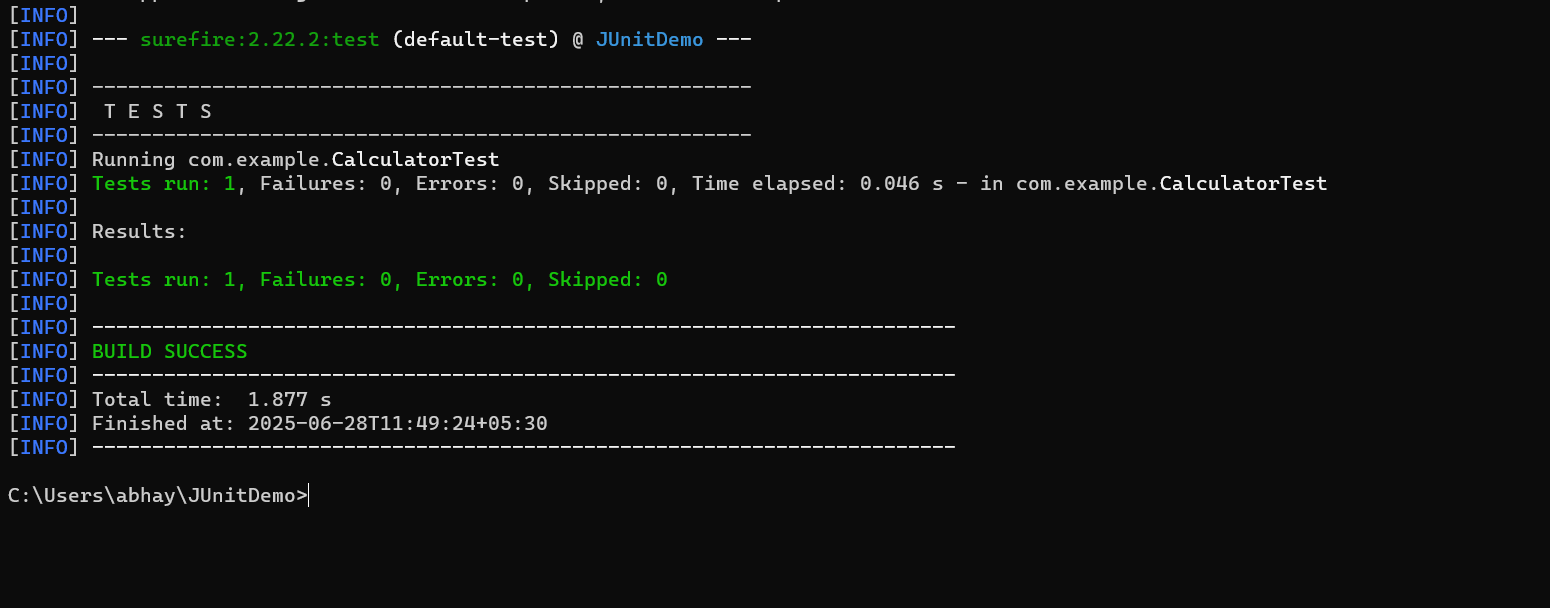
assertEquals(5, calc.add(2, 3));

}

}



Output:



Exercise 3: Assertions in JUnit(Mandatory)

Scenario:

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

Steps:

1. Write tests using various JUnit assertions.

Solution Code:

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

AssertionsDemo demo = new AssertionsDemo();

@Test

    public void testAssertions() {

        // Assert equals

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

// Assert true

  assertTrue(demo.isEven(4));

// Assert false

assertFalse(demo.isEven(5));

// Assert null

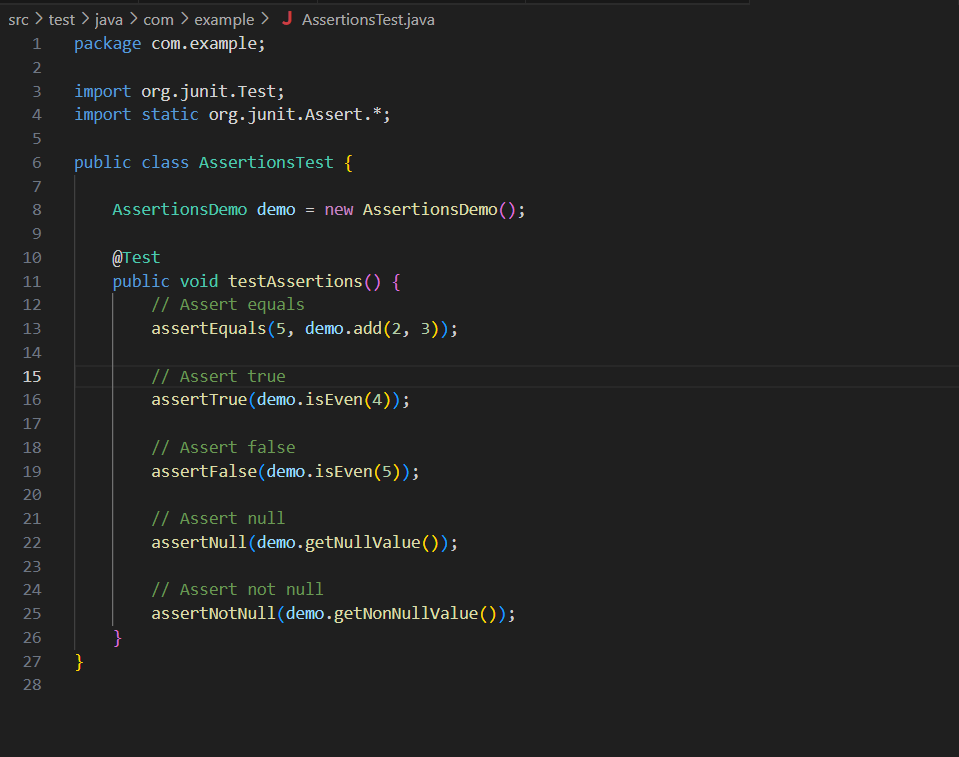
assertNull(demo.getNullValue());

// Assert not null

assertNotNull(demo.getNonNullValue());

    }

}



package com.example;

public class AssertionsDemo {

public int add(int a, int b) {

return a + b;

}

public boolean isEven(int n) {

return n % 2 == 0;

}

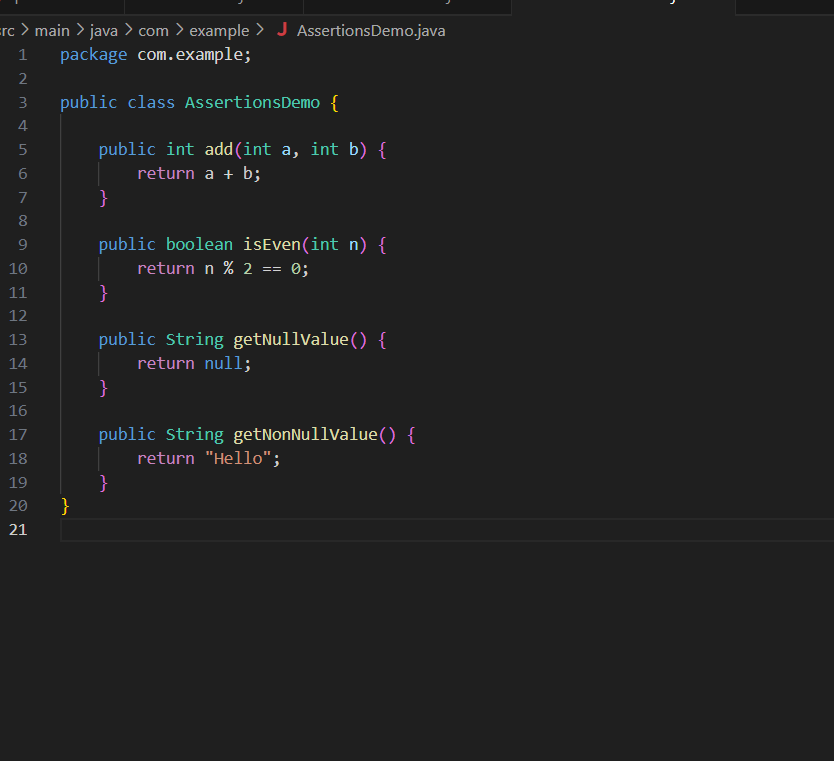
public String getNullValue() {

return null;

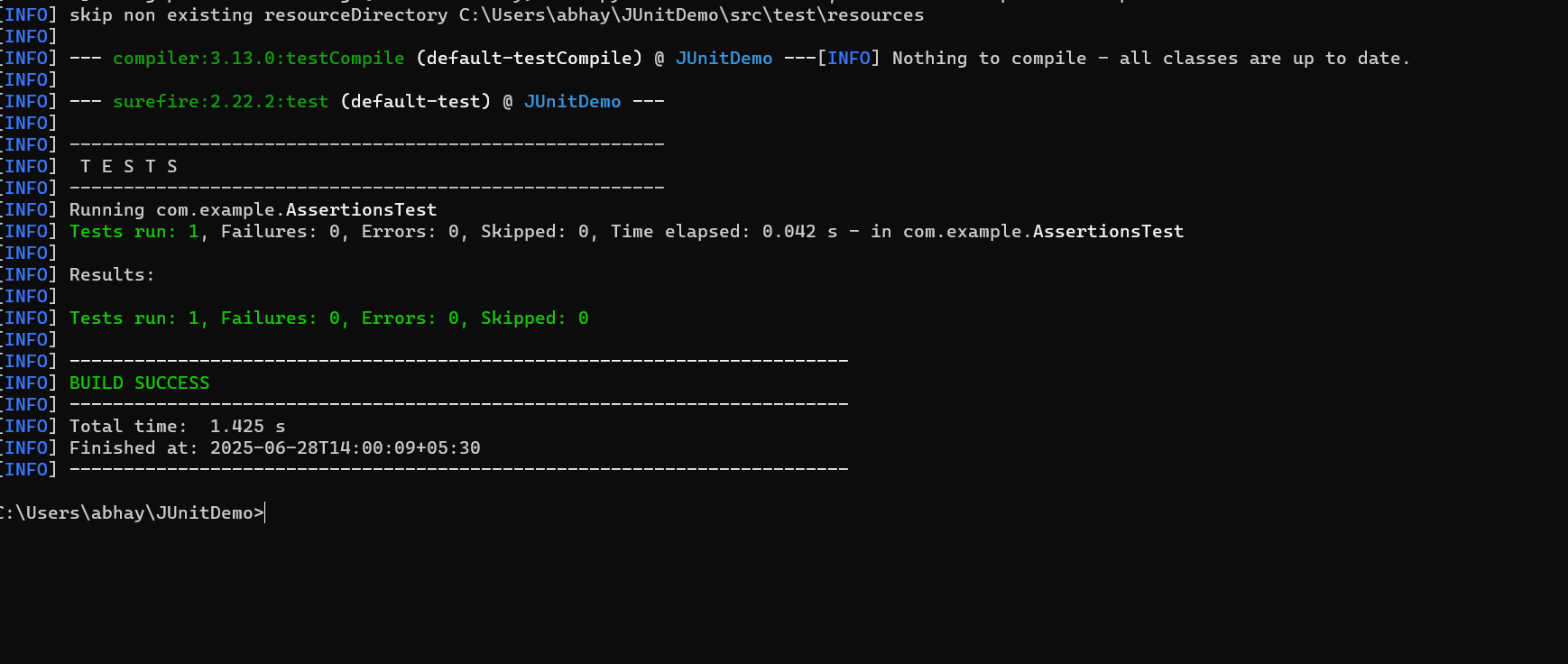
}

public String getNonNullValue() {

return "Hello"; }}



Output:



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

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.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

private Calculator calculator;

@Before

public void setUp() {

System.out.println("before calculator test");

calculator = new Calculator();

}

@After

public void tearDown() {

System.out.println("after calculator test");

calculator = null;

}

@Test

public void testAdd() {

int a = 5;

int b = 3;

int result = calculator.add(a, b);

assertEquals(8, result);

}

@Test

public void testSubtract() {

int a = 10;

int b = 4;

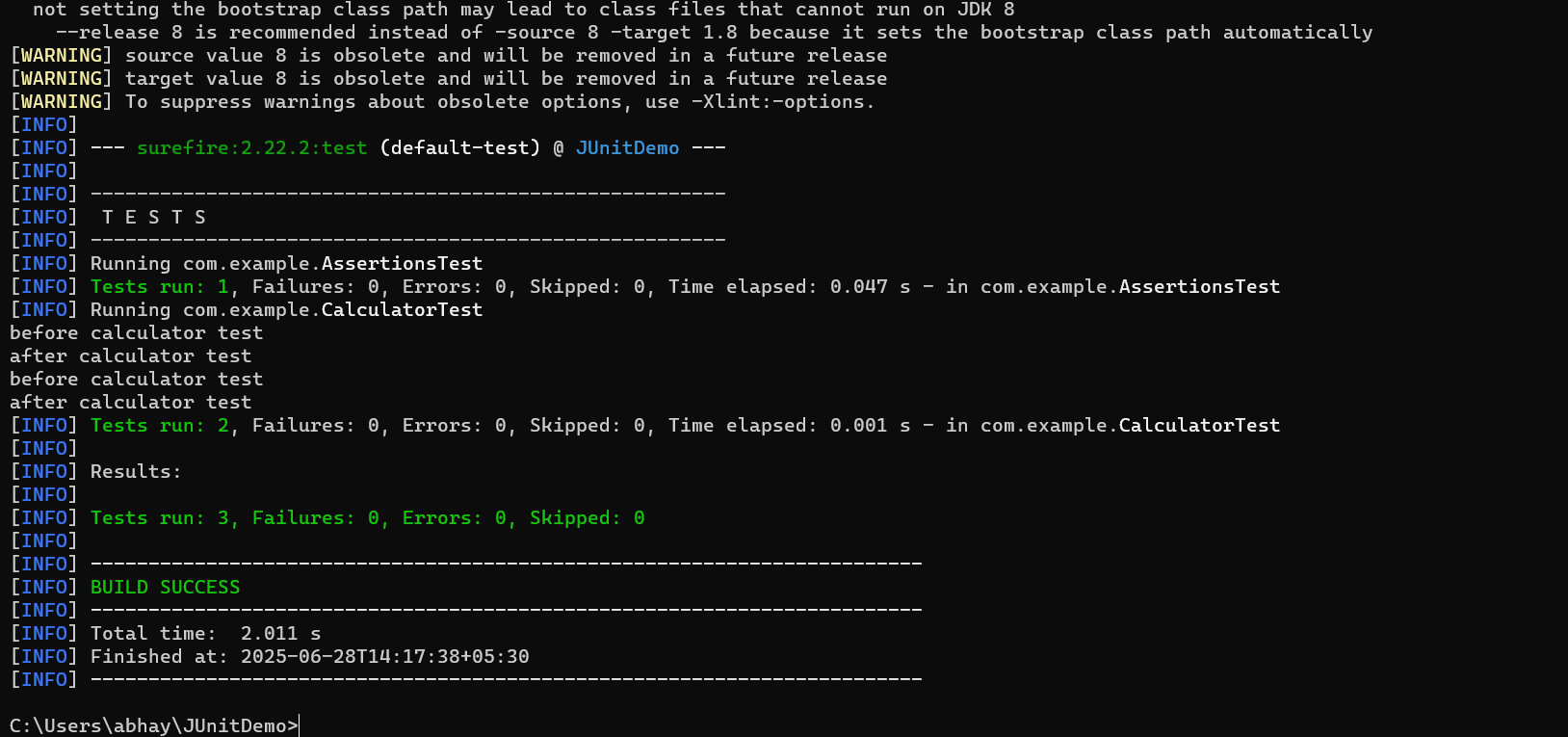
int result = calculator.subtract(a, b);

assertEquals(6, result);

}

}

Output:



Exercise 2: Writing Basic JUnit Tests(Additional)

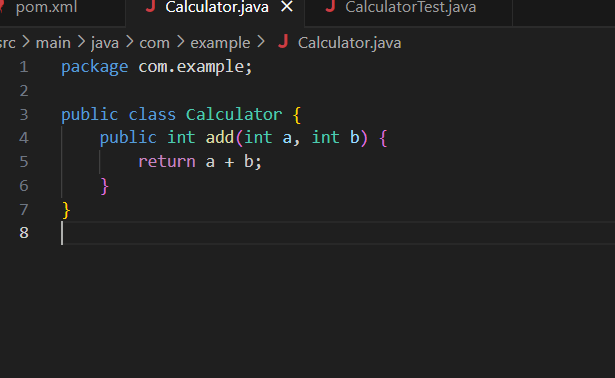
Scenario:

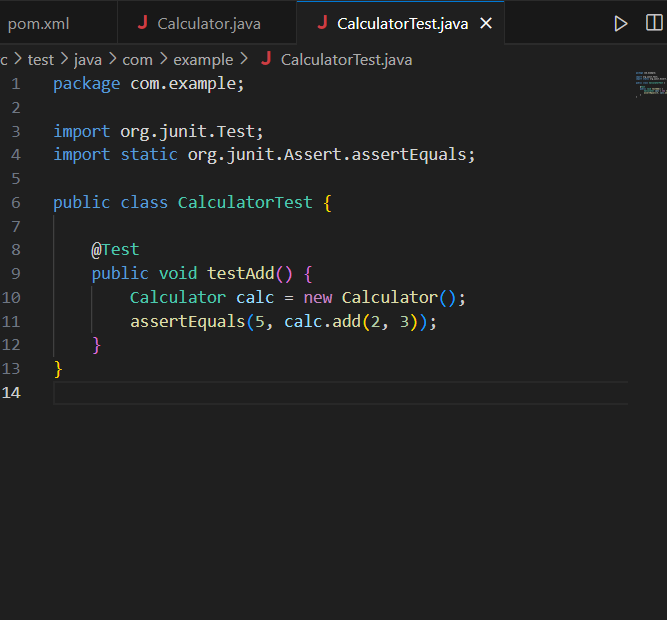
You need to write basic JUnit tests for a simple Java class.

Steps:

1. Create a new Java class with some methods to test.

2. Write JUnit tests for these methods.





Output:

