**Week2\_Junit Basic Testing Exercises**

**Exercise 1: Setting up Junit**

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

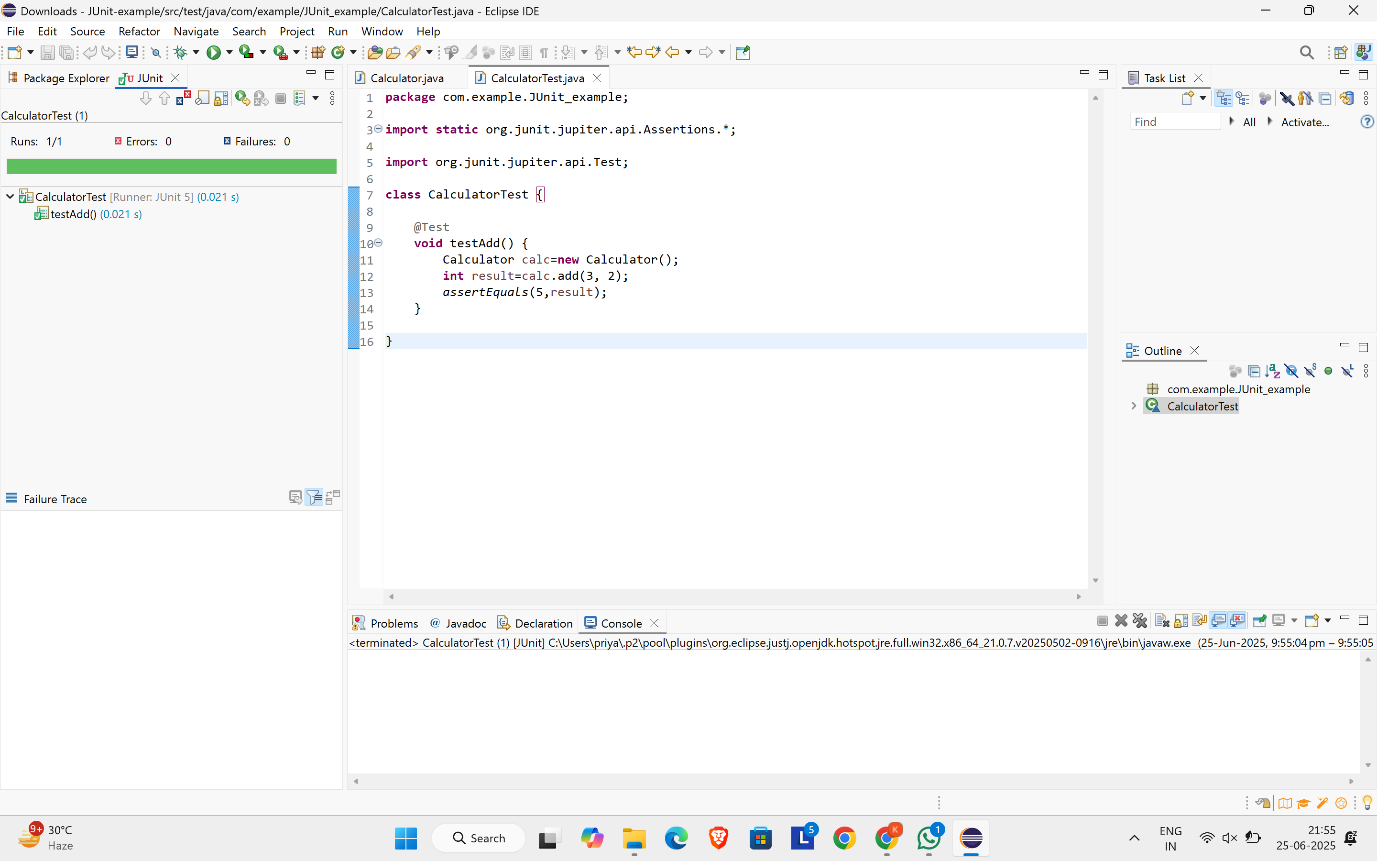
int result = calc.add(2, 3);

assertEquals(5, result);

}

}

**OUTPUT**



**Exercise 2 : Writing Basic Junit Tests**

**Calculator.java**

package com.example.calculator;

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.calculator;

import org.junit.jupiter.api.Test;

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

class CalculatorTest {

Calculator calculator = new Calculator();

@Test

void testAdd() {

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

}

@Test

void testSubtract() {

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

}

@Test

void testMultiply() {

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

}

@Test

void testDivide() {

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

}

@Test

void testDivideByZero() {

Exception exception = assertThrows(IllegalArgumentException.class, () -> {

calculator.divide(10, 0);

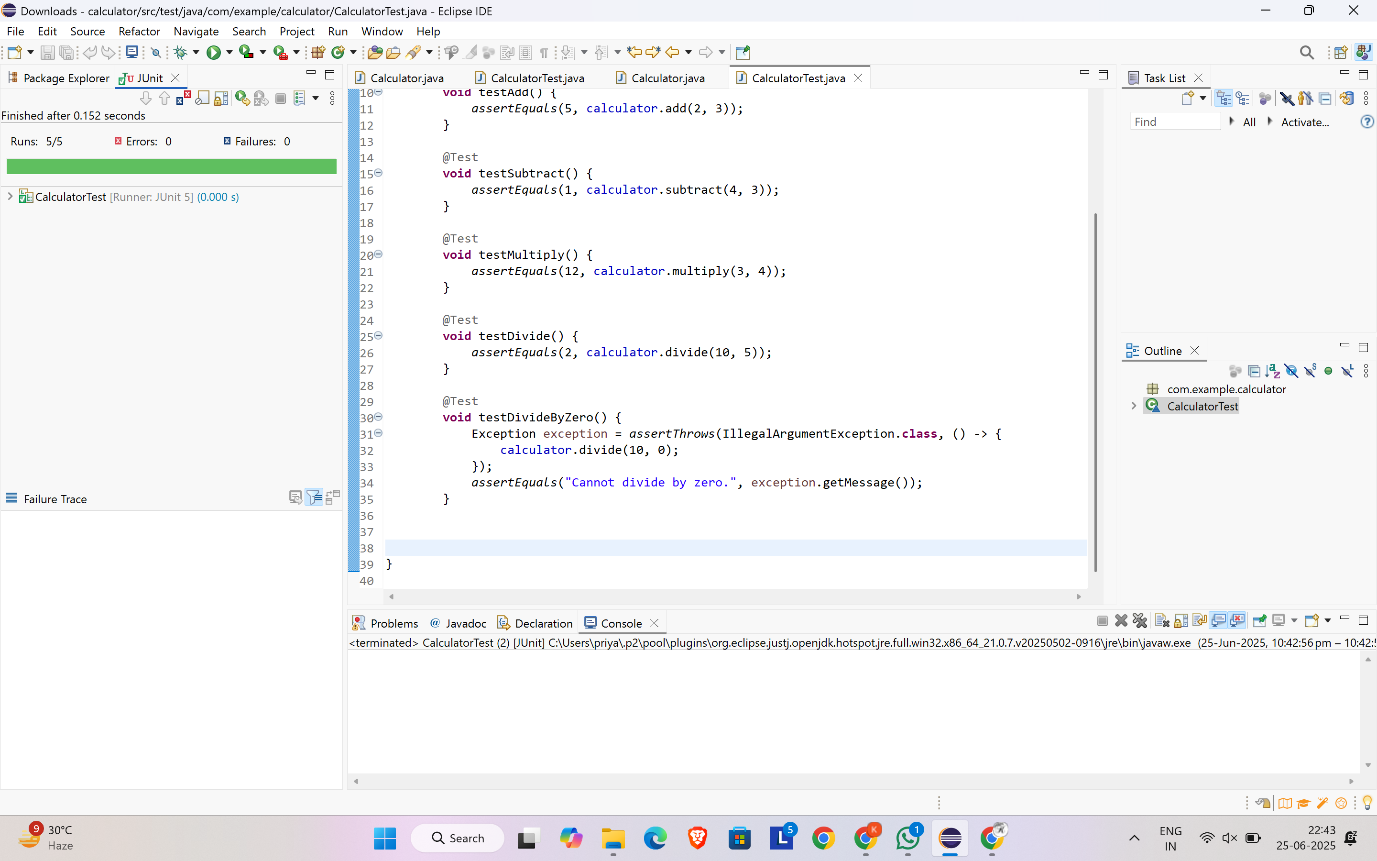
});

assertEquals("Cannot divide by zero.", exception.getMessage());

}

}

**OUTPUT**



**Exercise 3 : Assertions in Junit**

**AssertionsTest.java**

import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

@Test

public void testAssertions() {

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

assertTrue(5 > 3, "5 should be greater than 3");

assertFalse(5 < 3, "5 should not be less than 3");

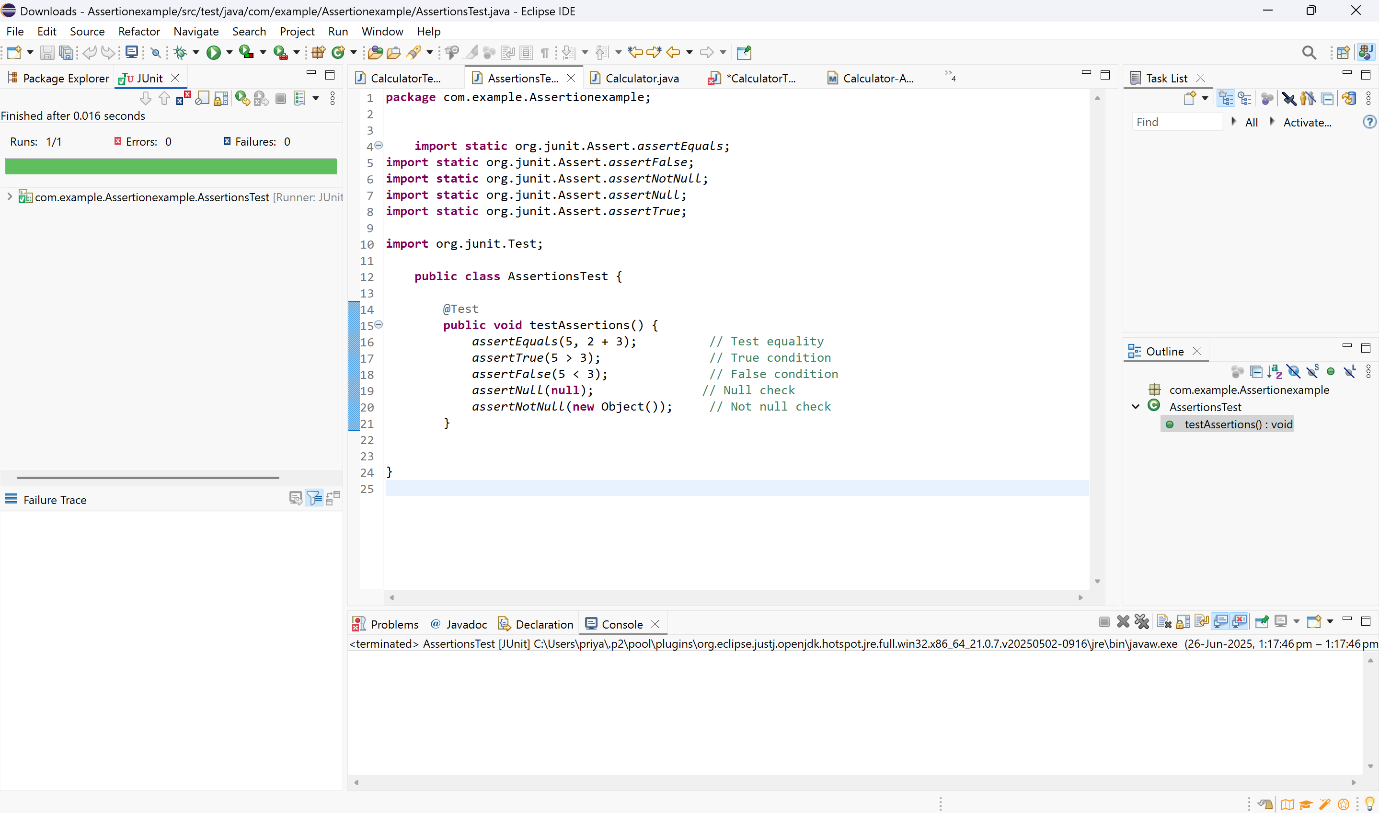
assertNull(null, "Object should be null");

assertNotNull(new Object(), "Object should not be null");

}

}

**OUTPUT**

****

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

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

**CalculatorTest.java**

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.Test;

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

public class CalculatorTest {

private Calculator calculator;

@BeforeEach

public void setUp() {

calculator = new Calculator();

System.out.println("Setup complete.");

}

@AfterEach

public void tearDown() {

calculator = null;

System.out.println("Teardown complete.");

}

@Test

public void testAdd() {

int a = 2;

int b = 3;

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

assertEquals(5, result, "Addition should be 5");

}

@Test

public void testSubtract() {

int a = 7;

int b = 4;

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

assertEquals(3, result, "Subtraction should be 3");

}

}

**OUTPUT**

