**WEEK 2**

**Test driven development and Logging framework**

**JUnit Testing Exercises**

**Exercise 1: Setting Up Junit**

**CODE:**

**Calculator.test.java:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

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

}

}

**Calculator.java:**

public class Calculator {

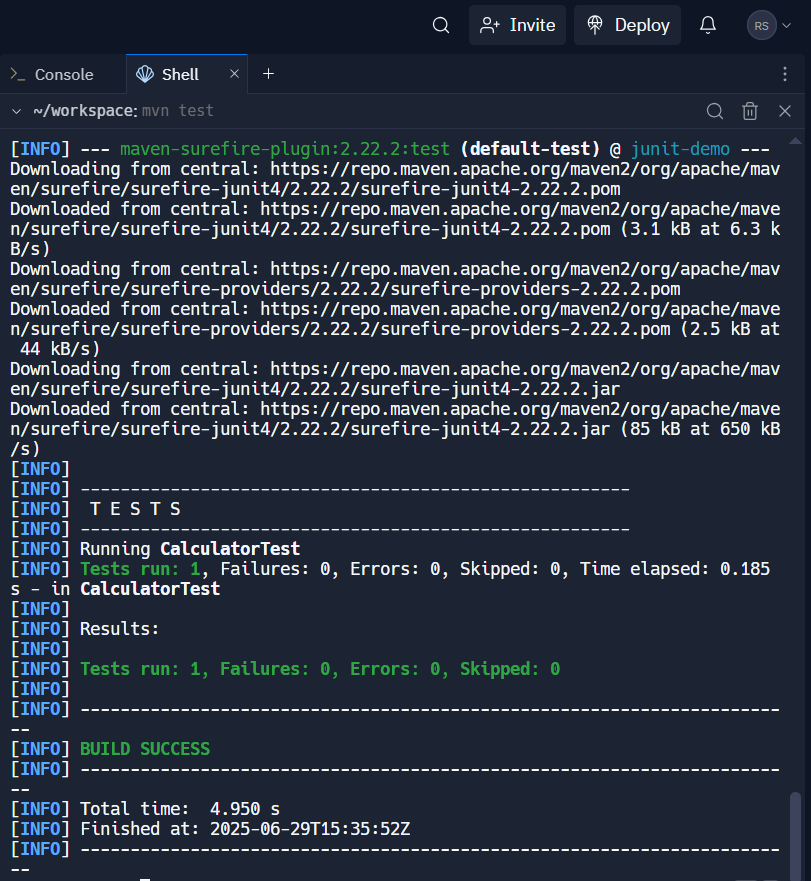
public int add(int a, int b) {

return a + b;

}

}

OUTPUT:



**Exercise 2: Writing Basic JUnit Tests**

**CODE:**

**MathUtilsTest.java:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathUtilsTest {

@Test

public void testAdd() {

MathUtils math = new MathUtils();

assertEquals(7, math.add(3, 4));

}

@Test

public void testSubtract() {

MathUtils math = new MathUtils();

assertEquals(2, math.subtract(5, 3));

}

@Test

public void testMultiply() {

MathUtils math = new MathUtils();

assertEquals(15, math.multiply(3, 5));

}

@Test

public void testDivide() {

MathUtils math = new MathUtils();

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

}

@Test(expected = IllegalArgumentException.class)

public void testDivideByZero() {

MathUtils math = new MathUtils();

math.divide(5, 0);

}

}

**MathUtils.java:**

public class MathUtils {

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) {

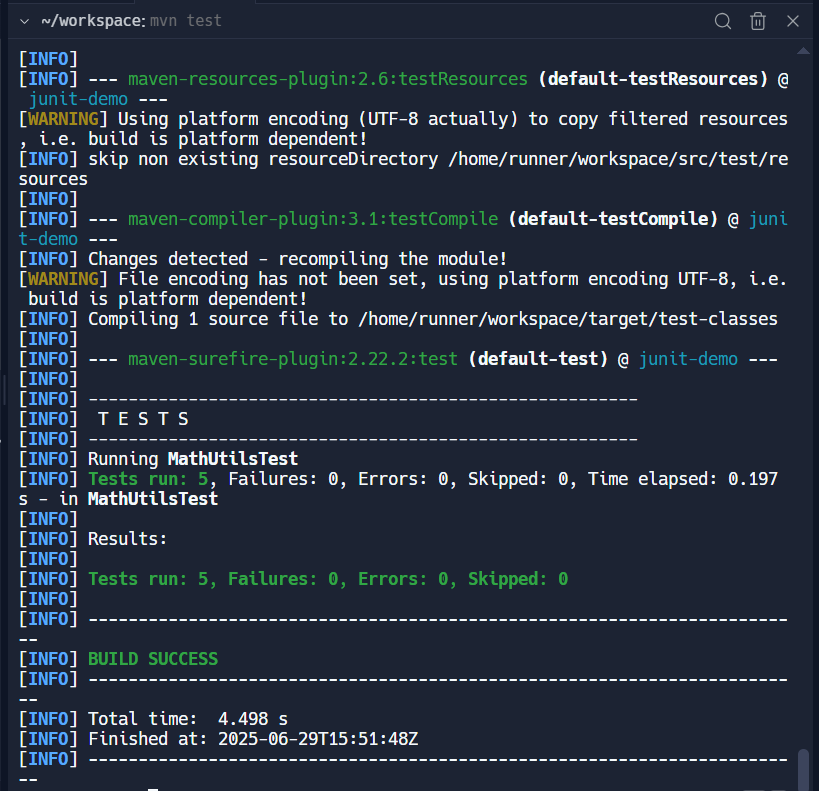
if (b == 0) throw new IllegalArgumentException("Cannot divide by zero");

return a / b;

}

}

**OUTPUT:**



**Exercise 3: Using Assertions in Junit**

**AssertionsTest.java:**

import org.junit.Test;

import static org.junit.Assert.\*;

Public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals

assertEquals(5, 2 + 3);

// Assert true

assertTrue(5 > 3);

// Assert false

assertFalse(5 < 3);

// Assert null

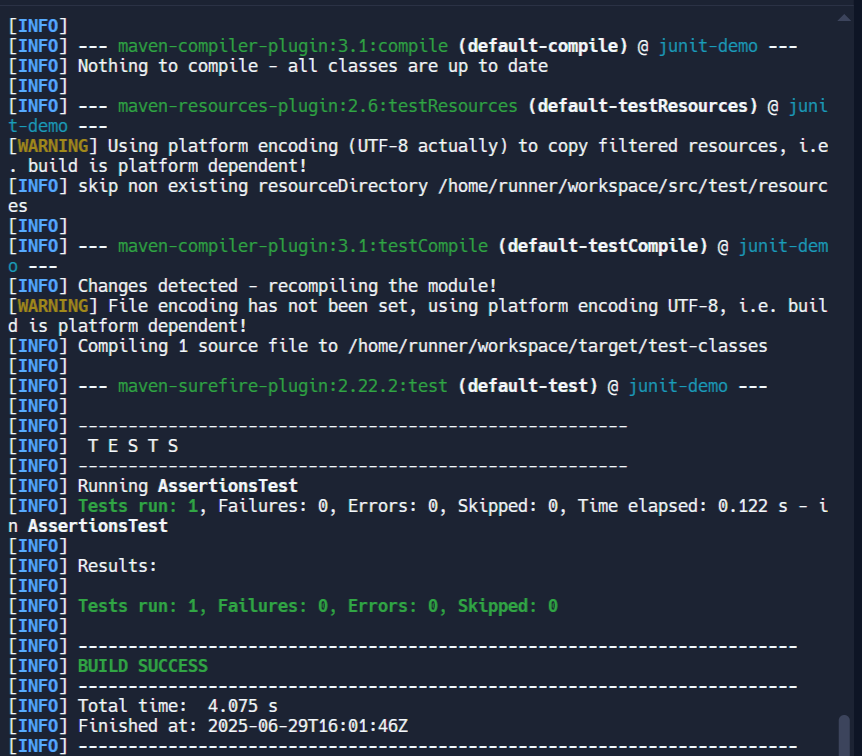
assertNull(null);

// Assert not null

assertNotNull(new Object());

}

}

**OUTPUT**

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

**CODE:**

**CalculatorLifecycleTest.java:**

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorLifecycleTest {

private Calculator calculator;

@Before

public void setUp() {

// Arrange: setup before each test

calculator = new Calculator();

System.out.println("Setup: Calculator initialized");

}

@After

public void tearDown() {

// Teardown: cleanup after each test

calculator = null;

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

}

@Test

public void testAddition() {

// Act

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

// Assert

assertEquals(5, result);

}

@Test

public void testSubtraction() {

// Act

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

// Assert

assertEquals(6, result);

}

}

**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; }

}

**OUTPUT:**

