Exercise-1:Setting Up the junit

//GreetTest.java is tested and compared with the resultant output from greet.java .Tested using maven and added a dependency to the file (pom.xml)

**Greet.java**

package com.example;

public class Greet {

    public void says(){

        System.out.println("Hello World");

    }

}

**GreeTest.java**

package com.example;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

public class GreeTest {

    @Test

    public void testGreet() {

        Greet greet = new Greet();

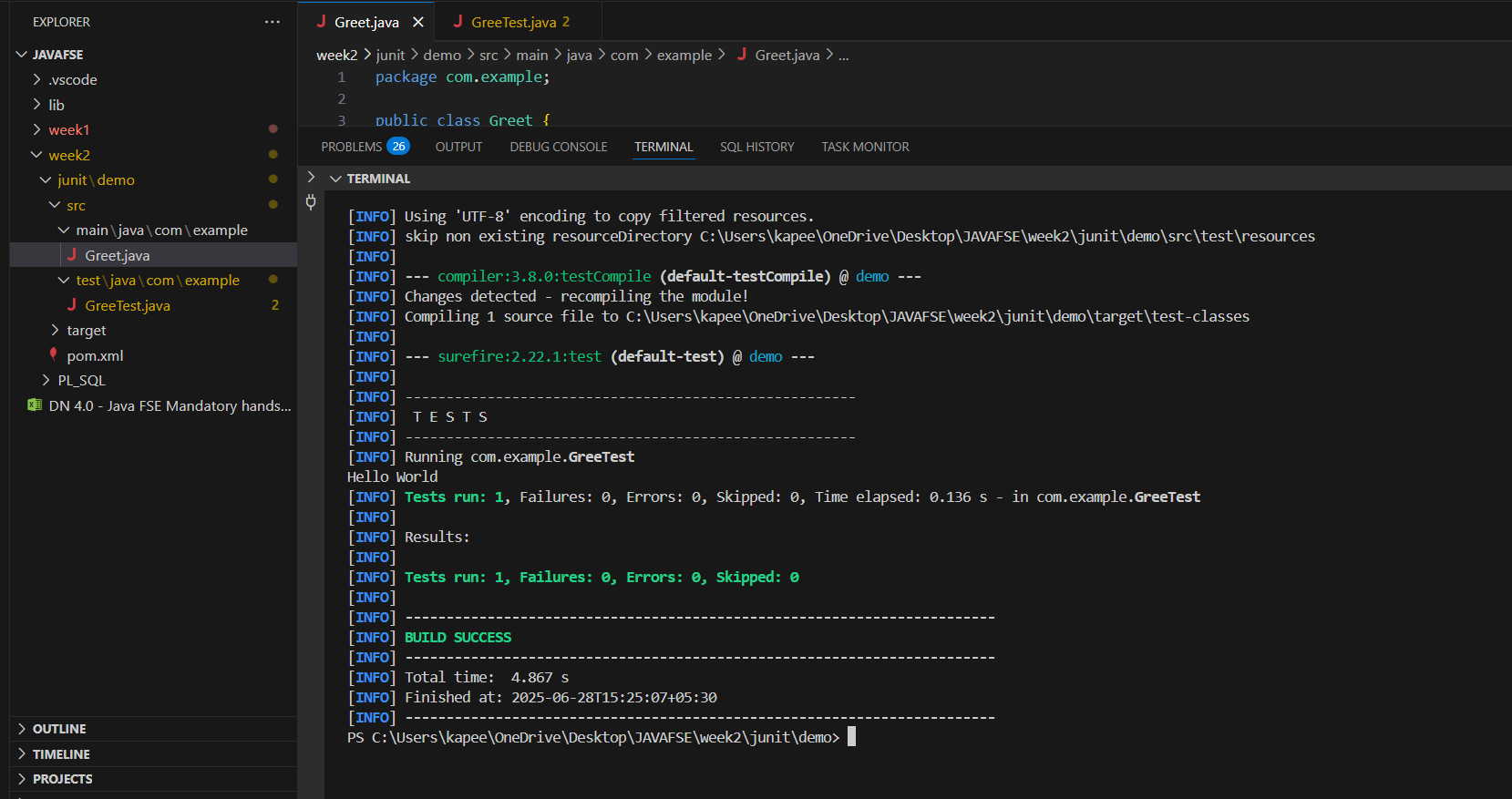
        greet.says(); // This will print "Hello World" to the console

        // No assertion needed, just checking if it runs without exceptions

    }

    }

Output:



Exercise-2:Writing Basic junitTests:

**Calculator.java**

package com.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("Division by zero is not allowed.");

        }

        return a / b;

    }

}

**CalculatorTest.java**

package com.example;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class CalculatorTest {

    @Test

    public void testAdd() {

        Calculator calc = new Calculator();

        int result = calc.add(3, 4);

        assertEquals(7, result);

    }

    @Test

    public void testSubtract() {

        Calculator calc = new Calculator();

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

        assertEquals(6, result);

    }

    @Test

    public void testMultiply() {

        Calculator calc = new Calculator();

        int result = calc.multiply(3, 4);

        assertEquals(12, result);

    }

    @Test

    public void testDivide() {

        Calculator calc = new Calculator();

        int result = calc.divide(12, 4);

        assertEquals(3, result);

    }

    @Test(expected = IllegalArgumentException.class)

    public void testDivideByZero() {

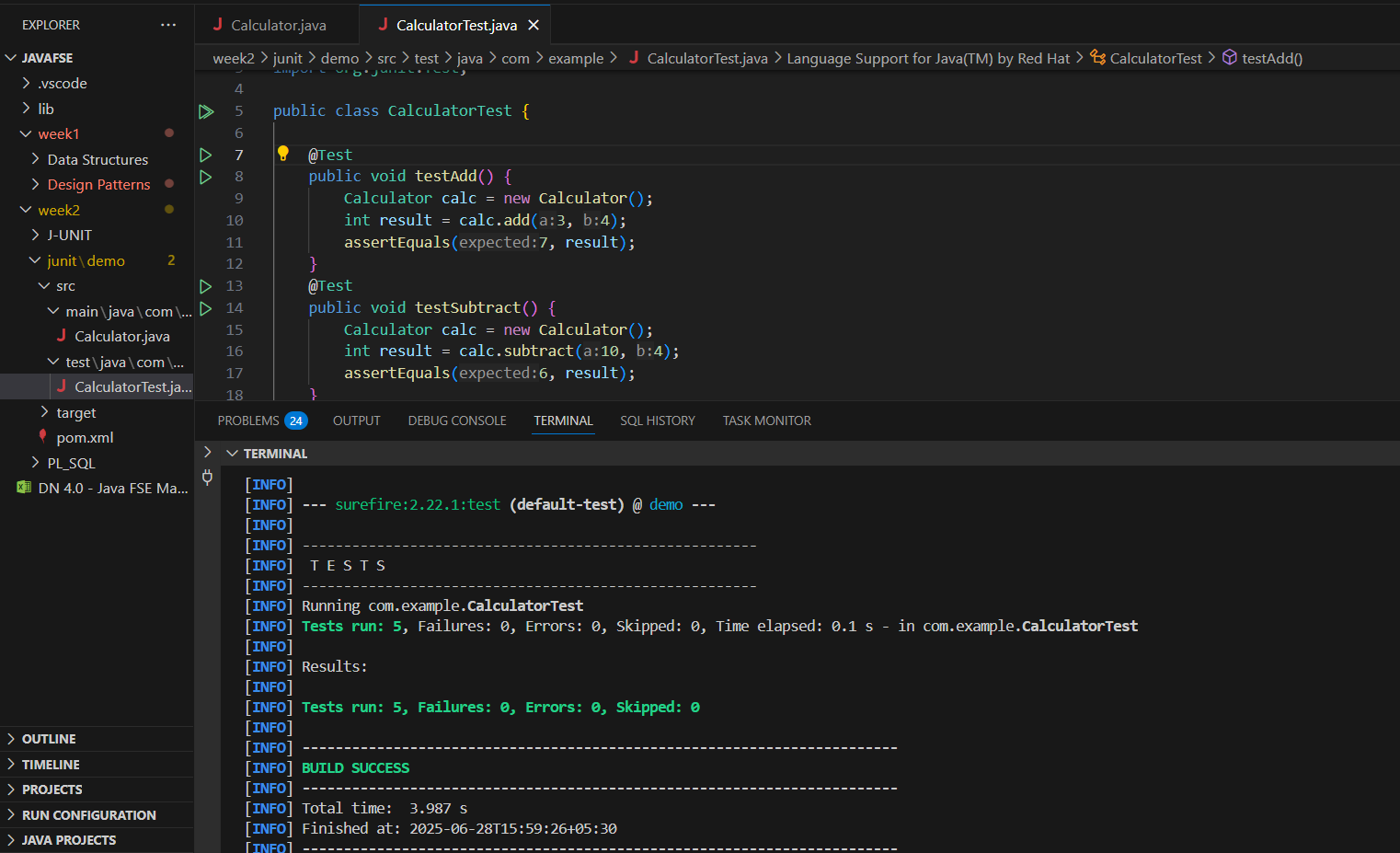
        Calculator calc = new Calculator();

        calc.divide(12, 0); // This should throw an IllegalArgumentException

    }

}

Output:



Exercise-3:Assertion in Junit

**AssertionTest.java**

package com.example;

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

import org.junit.jupiter.api.Test;

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

assertNull(null);

assertNotNull(new Object());

}

}

//to compile the assertion test we need to add a dependency in pom.xml file

<dependency>

<groupId>org.junit.jupiter</groupId>

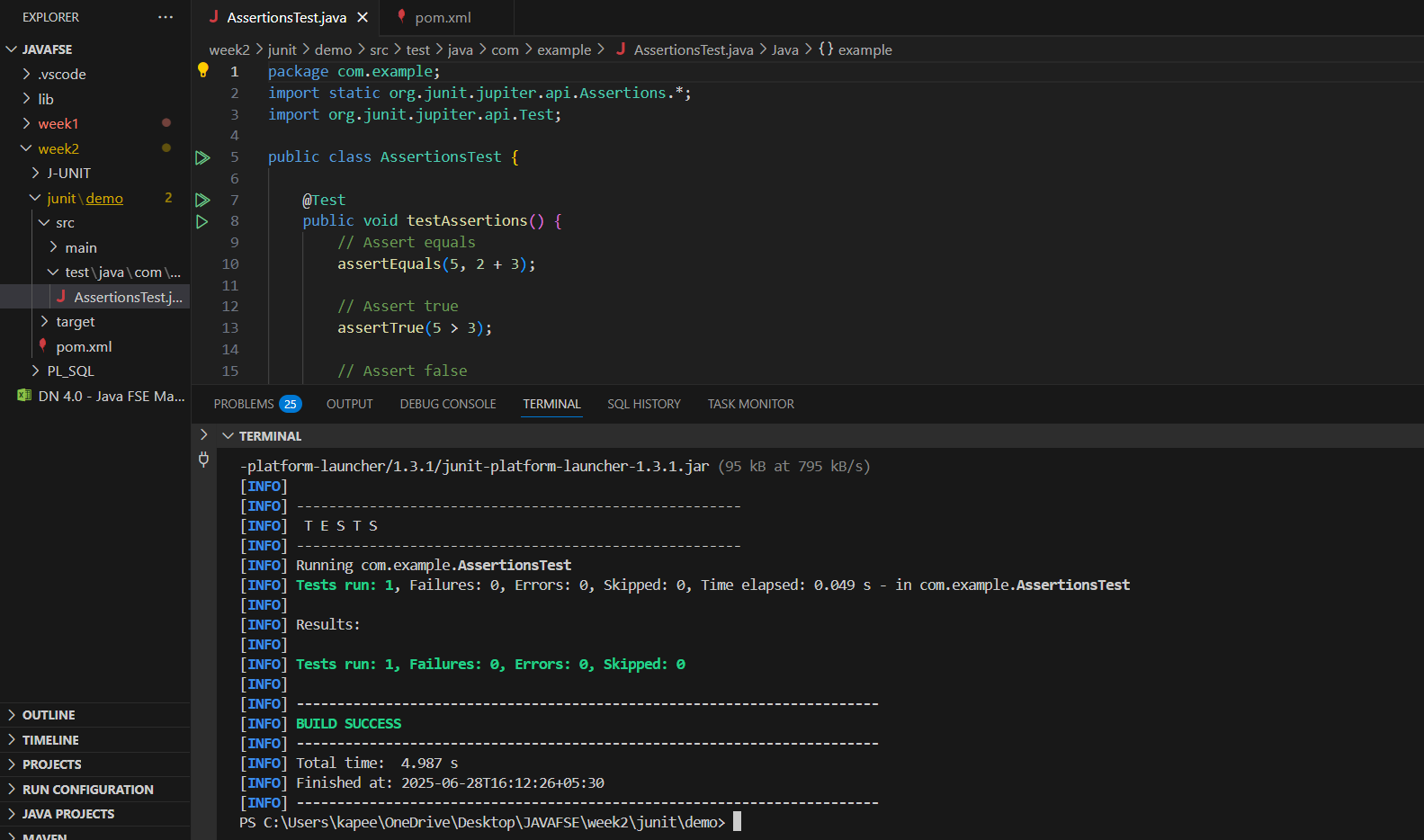
<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

Output:



Exercise-4:Arrange-Act-Assert Pattern (AAA) ,Test fixtures and Tear down methods in junit

**StringUtilsTest.java**

package com.example;

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

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

class StringUtils {

    String reverse(String input) {

        return new StringBuilder(input).reverse().toString();

    }

}

public class StringUtilsTest {

    private StringUtils stringUtils;

    @BeforeEach

    void setUp() {

        stringUtils = new StringUtils();

    }

    @AfterEach

    void tearDown() {

        stringUtils = null;

    }

    @Test

    void testReverse() {

        String original = "hello";

        String reversed = stringUtils.reverse(original);

        assertEquals("olleh", reversed, "The reversed string should be 'olleh'");

    }

}

//compares the string after reversed is it equal or not

//@beforeeach is used to intiliaze new object

//@aftereach is used to assign to null after the used

Output:

