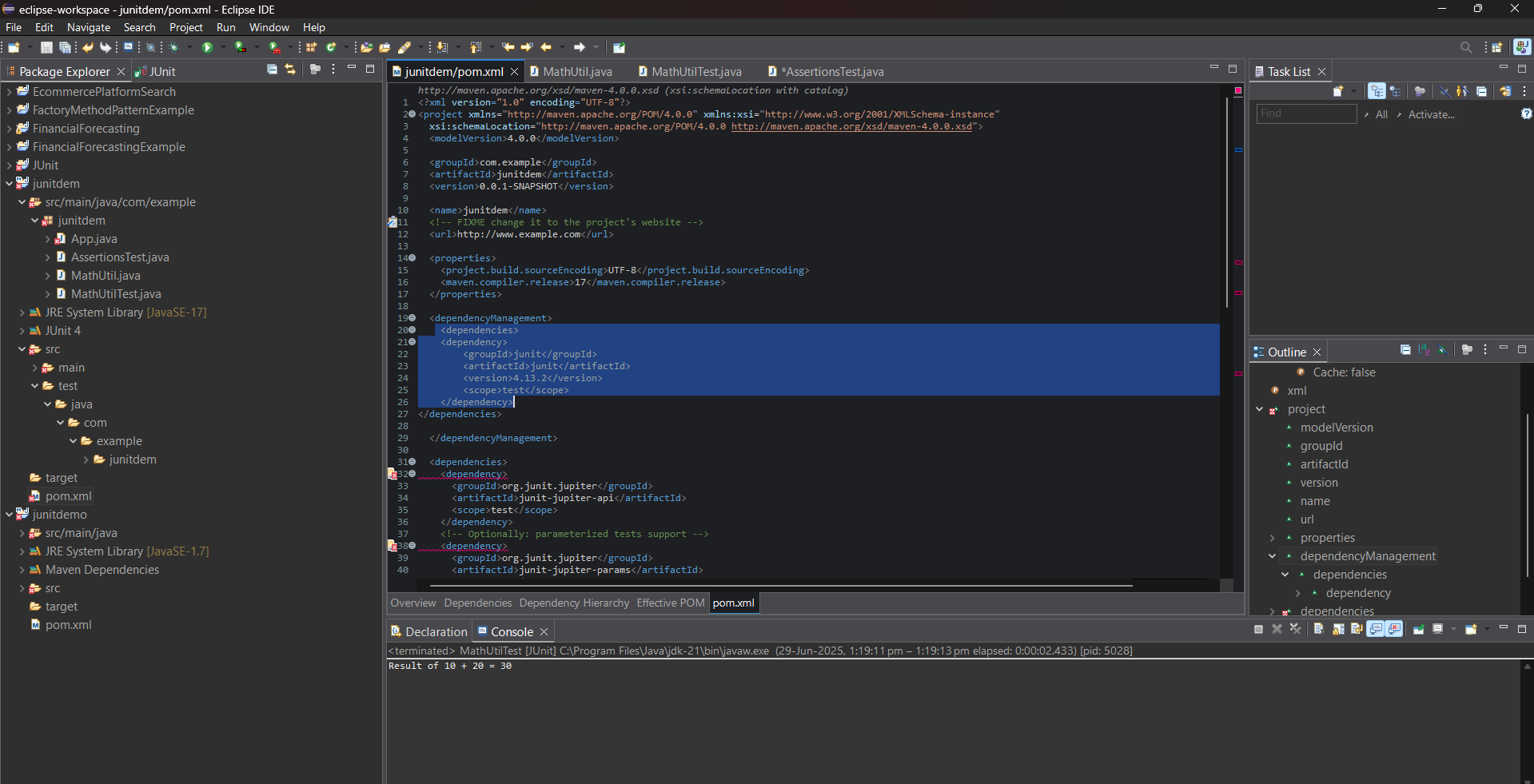
1. JUnit\_Basic Testing Exercises

# Exercise 1: Setting Up JUnit

**Adding Dependencies in Pom.xml :**

****

**MathUtil.java :**

package junitdem;

public class MathUtil {

public int add(int a, int b) {

return a + b;

}

}

**MathUtilTest.java :**

package junitdem;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class MathUtilTest {

@Test

public void testAdd() {

MathUtil math = new MathUtil();

int result = math.add(10, 20);

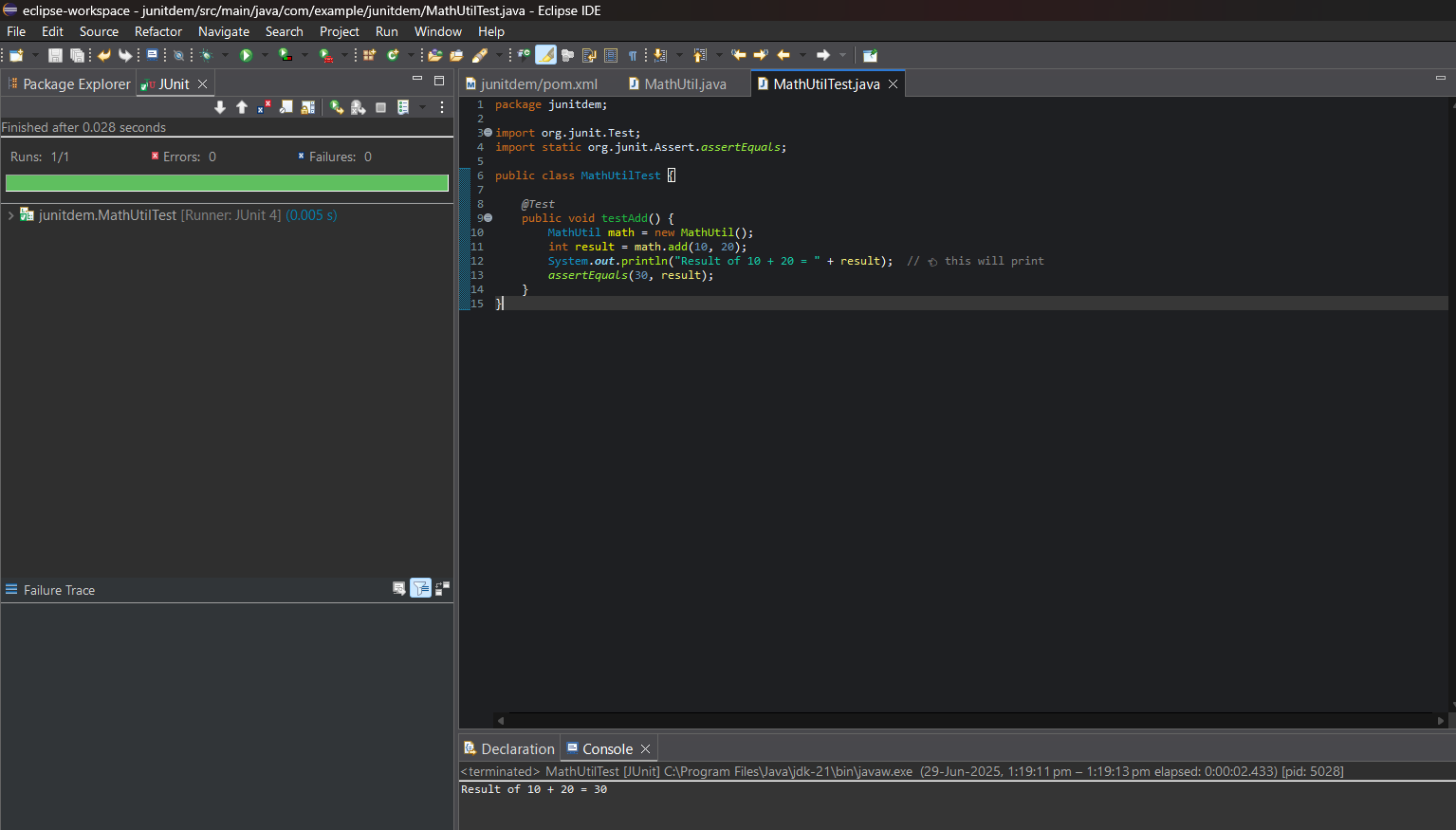
System.out.println("Result of 10 + 20 = " + result);

assertEquals(30, result);

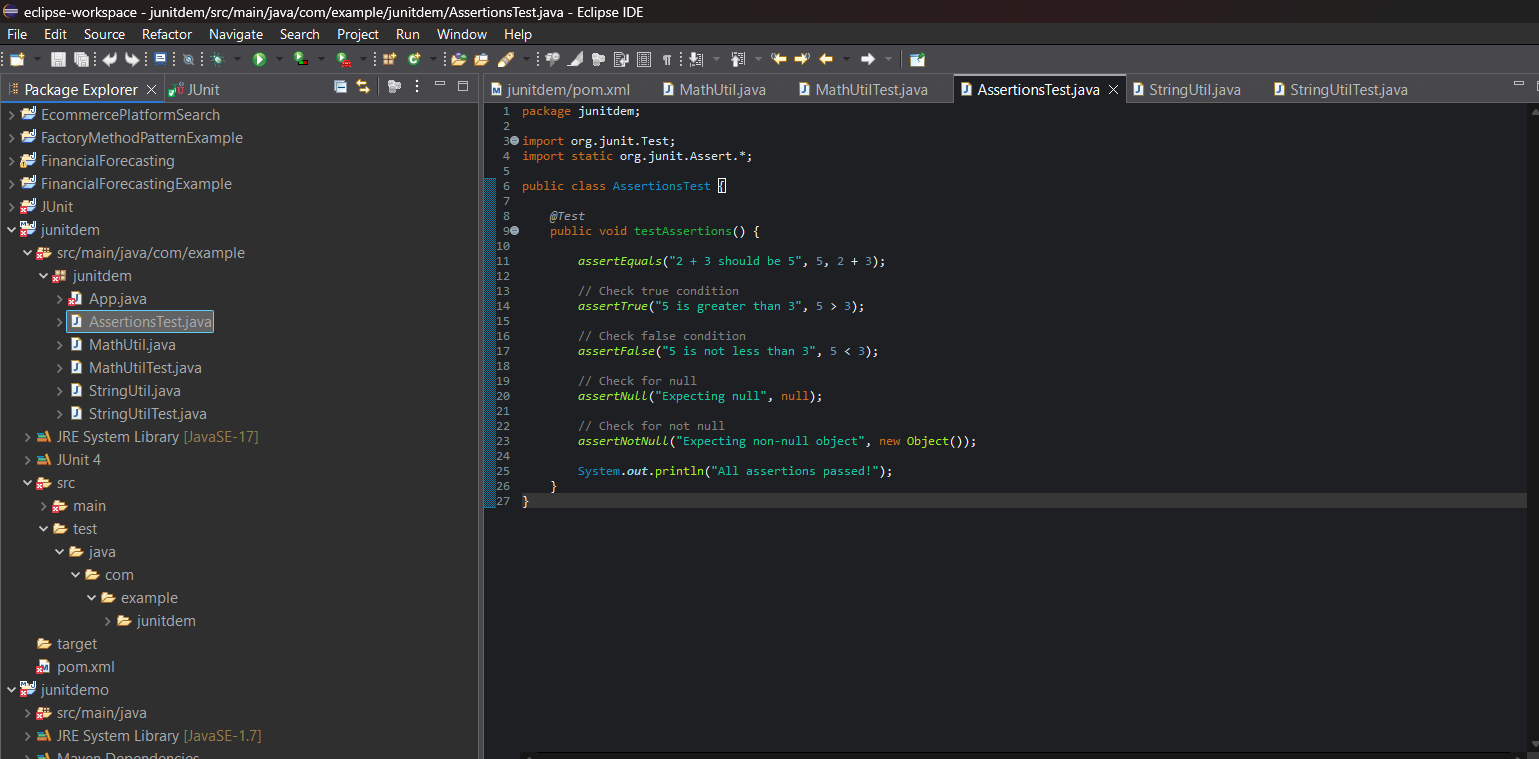
}

}

OUTPUT:



# Exercise 3: Assertions in JUnit

****

**AssertionsTest.java :**

package junitdem;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

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

// Check true condition

assertTrue("5 is greater than 3", 5 > 3);

// Check false condition

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

// Check for null

assertNull("Expecting null", null);

// Check for not null

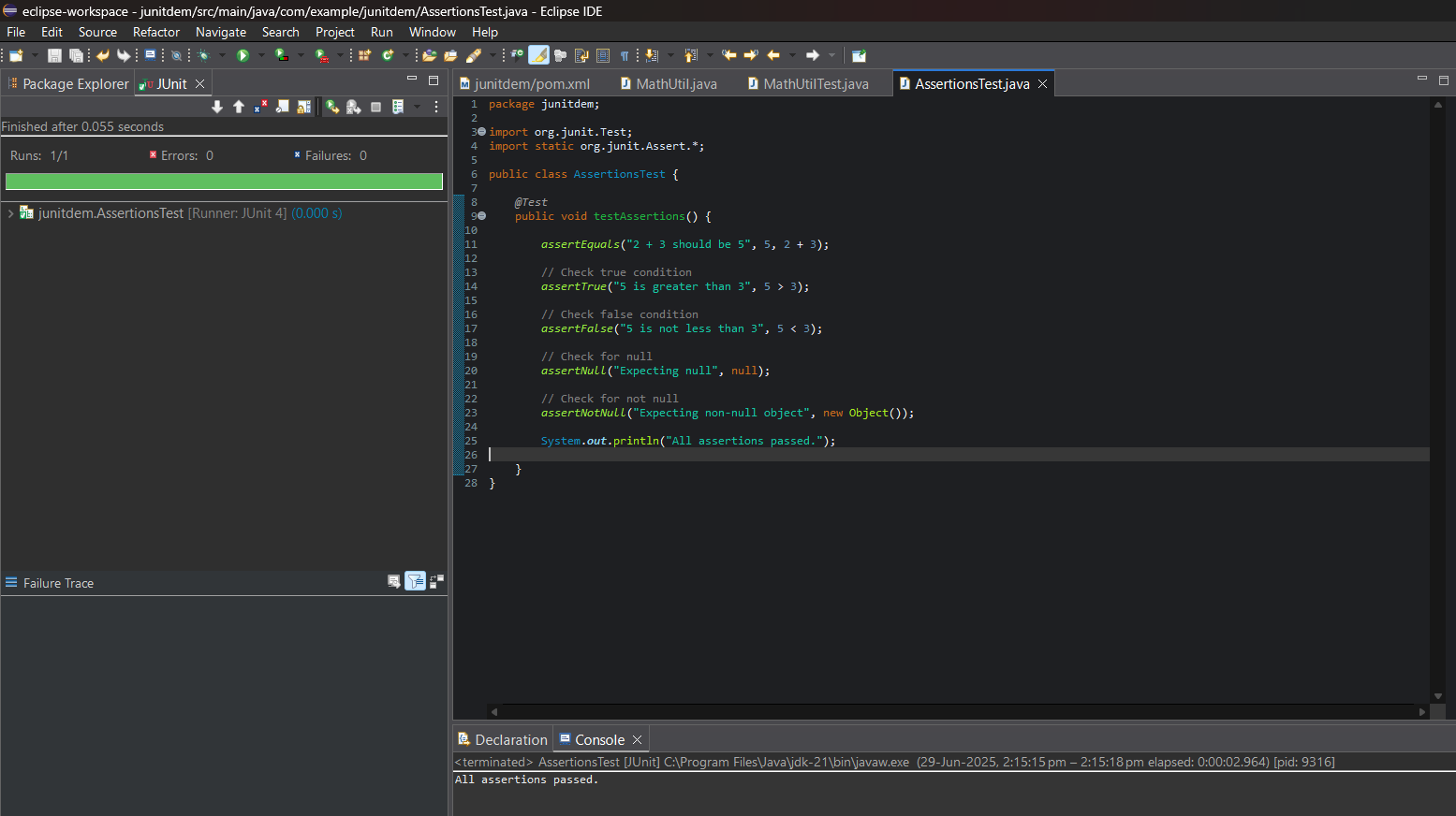
assertNotNull("Expecting non-null object", new Object());

System.out.println("All assertions passed!");

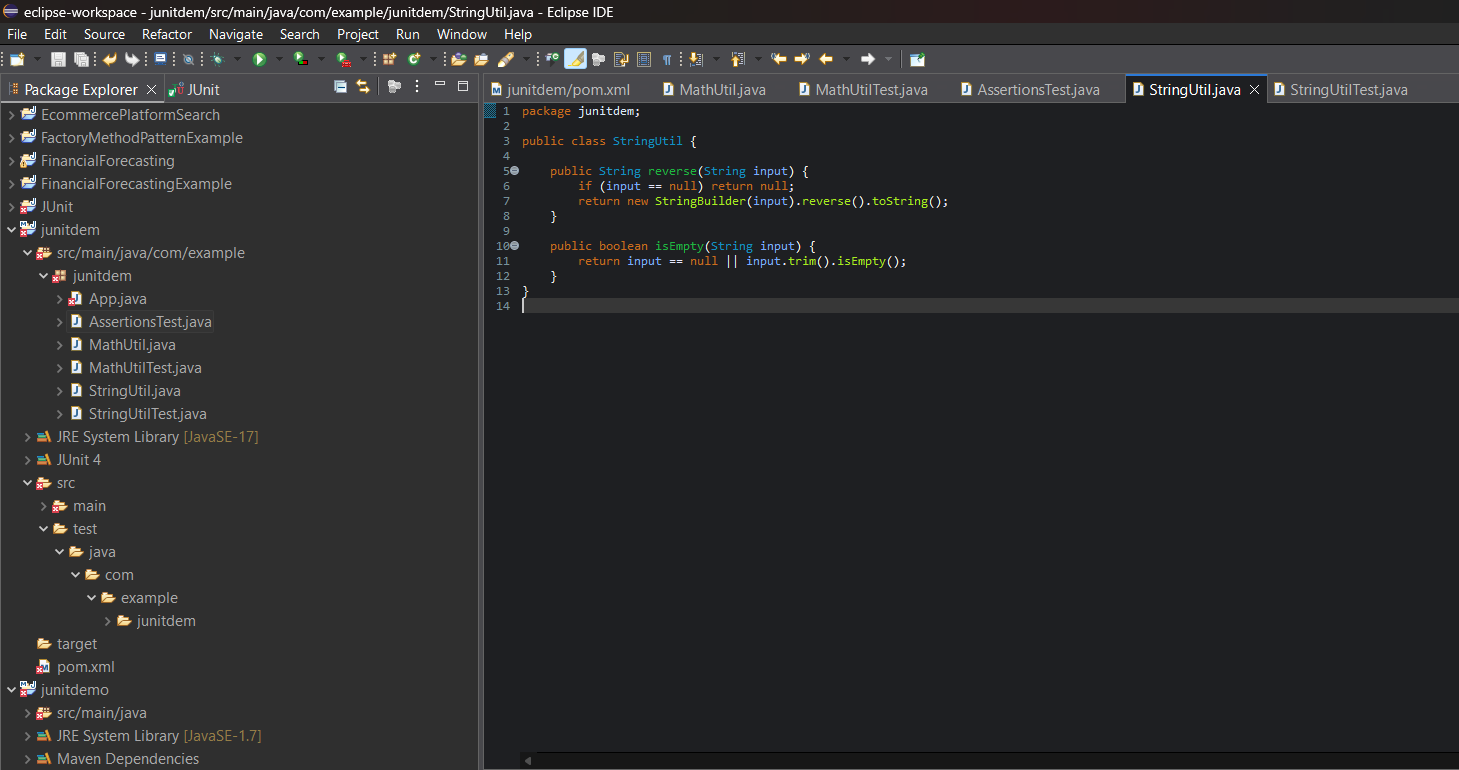
}

}

**OUTPUT:**



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

****

**StringUtil.java :**

package junitdem;

public class StringUtil {

public String reverse(String input) {

if (input == null) return null;

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

}

public boolean isEmpty(String input) {

return input == null || input.trim().isEmpty();

}

}

**StringUtilTest.java :**

package junitdem;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class StringUtilTest {

private StringUtil stringUtil;

@Before

public void setUp() {

System.out.println("Setting up...");

stringUtil = new StringUtil();

}

@After

public void tearDown() {

System.out.println("Cleaning up...");

stringUtil = null;

}

@Test

public void testReverse() {

String result = stringUtil.reverse("hello");

assertEquals("olleh", result);

}

@Test

public void testReverseNull() {

String result = stringUtil.reverse(null);

assertNull(result);

}

@Test

public void testIsEmpty\_True() {

assertTrue(stringUtil.isEmpty(" "));

}

@Test

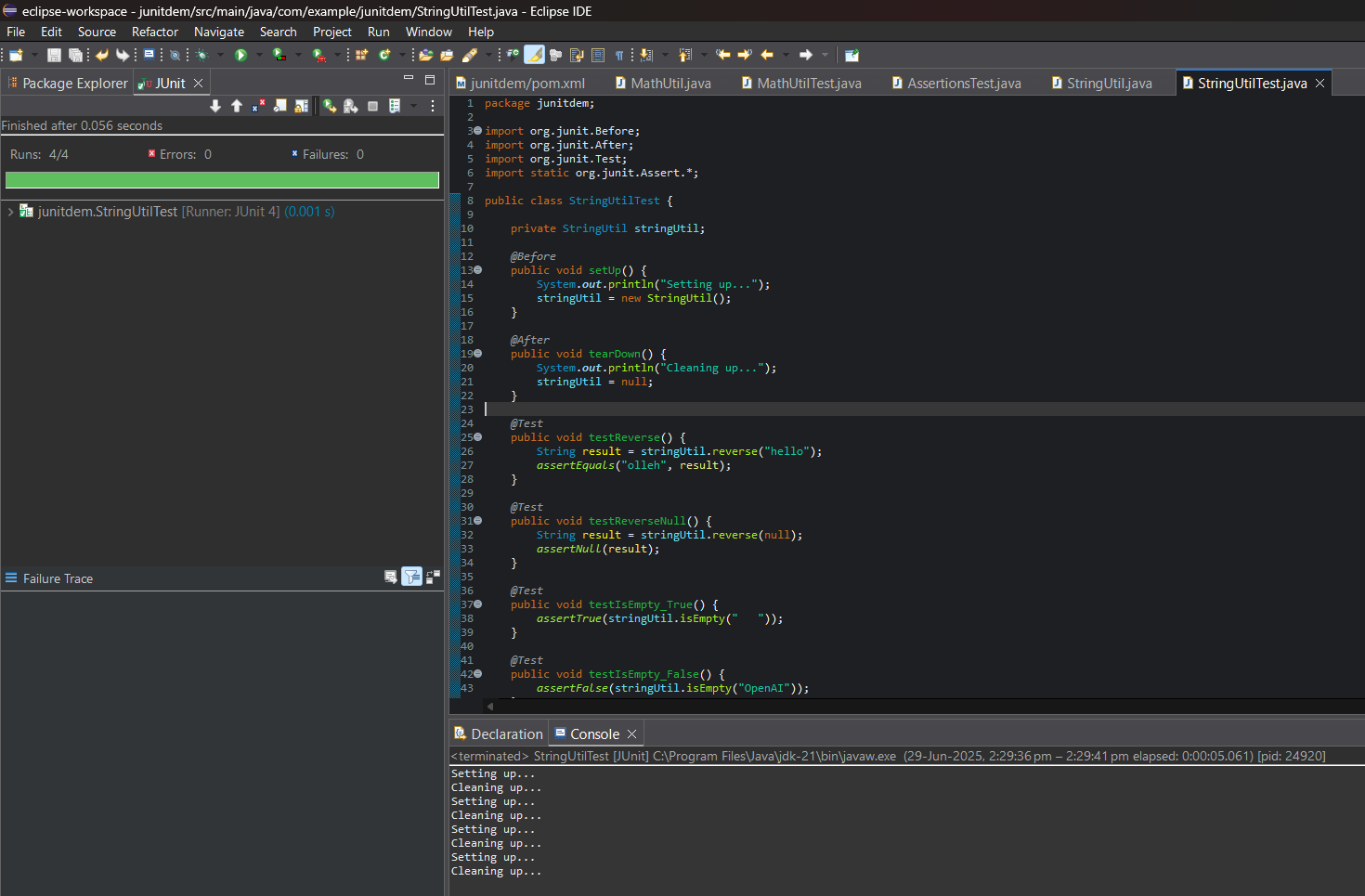
public void testIsEmpty\_False() {

assertFalse(stringUtil.isEmpty("OpenAI"));

}

}

**OUTPUT:**



3. Mockito exercises

# Exercise 1: Mocking and Stubbing from Scratch

**Adding Dependencies to Pom.xml :**

<dependencies>

<!-- JUnit 5 API -->

<dependency>

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

<artifactId>junit-jupiter-api</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

<!-- JUnit 5 Engine (required to run tests) -->

<dependency>

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

<artifactId>junit-jupiter-engine</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

<!-- Mockito Core -->

<dependency>

<groupId>org.mockito</groupId>

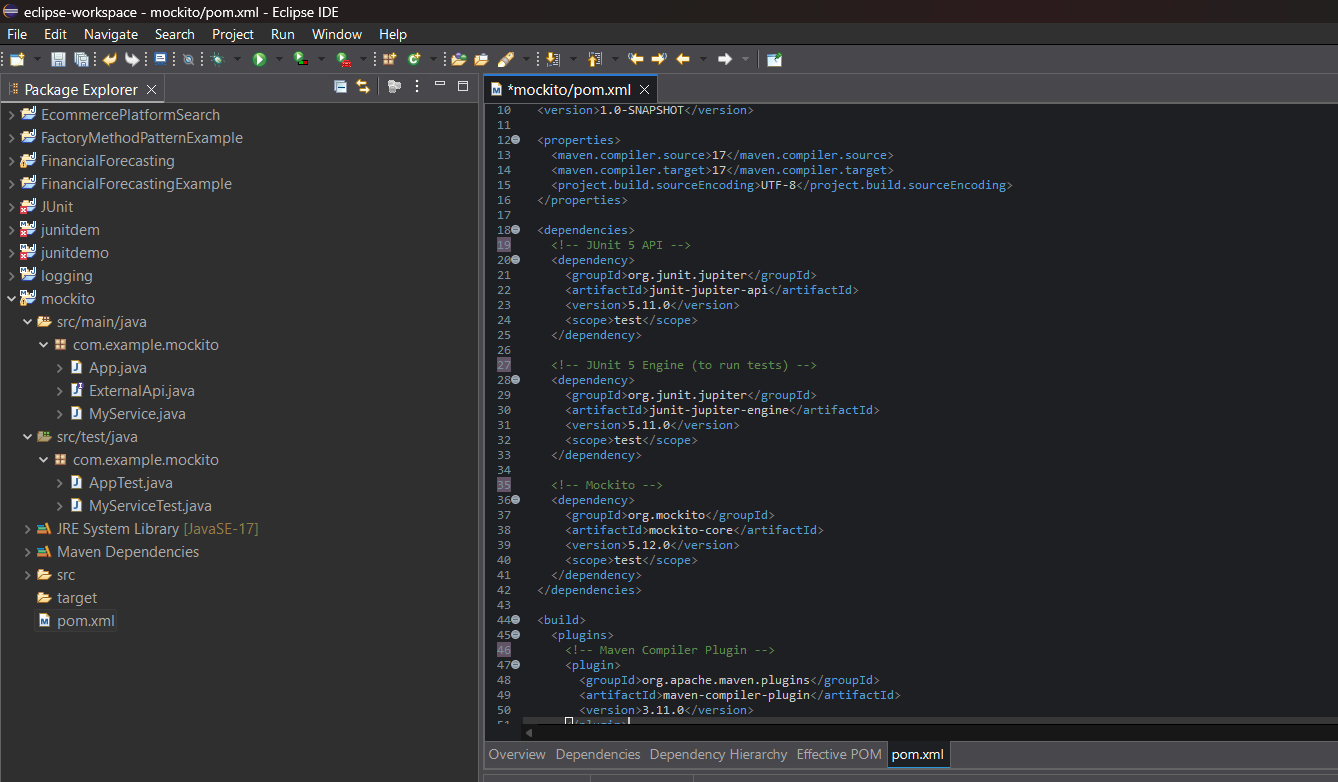
<artifactId>mockito-core</artifactId>

<version>5.12.0</version>

<scope>test</scope>

</dependency>

</dependencies>



**ExternalApi.java :**

package com.example.mockito;

public interface ExternalApi {

String getData();

}

**MyService.java :**

package com.example.mockito;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java :**

package com.example.mockito;

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

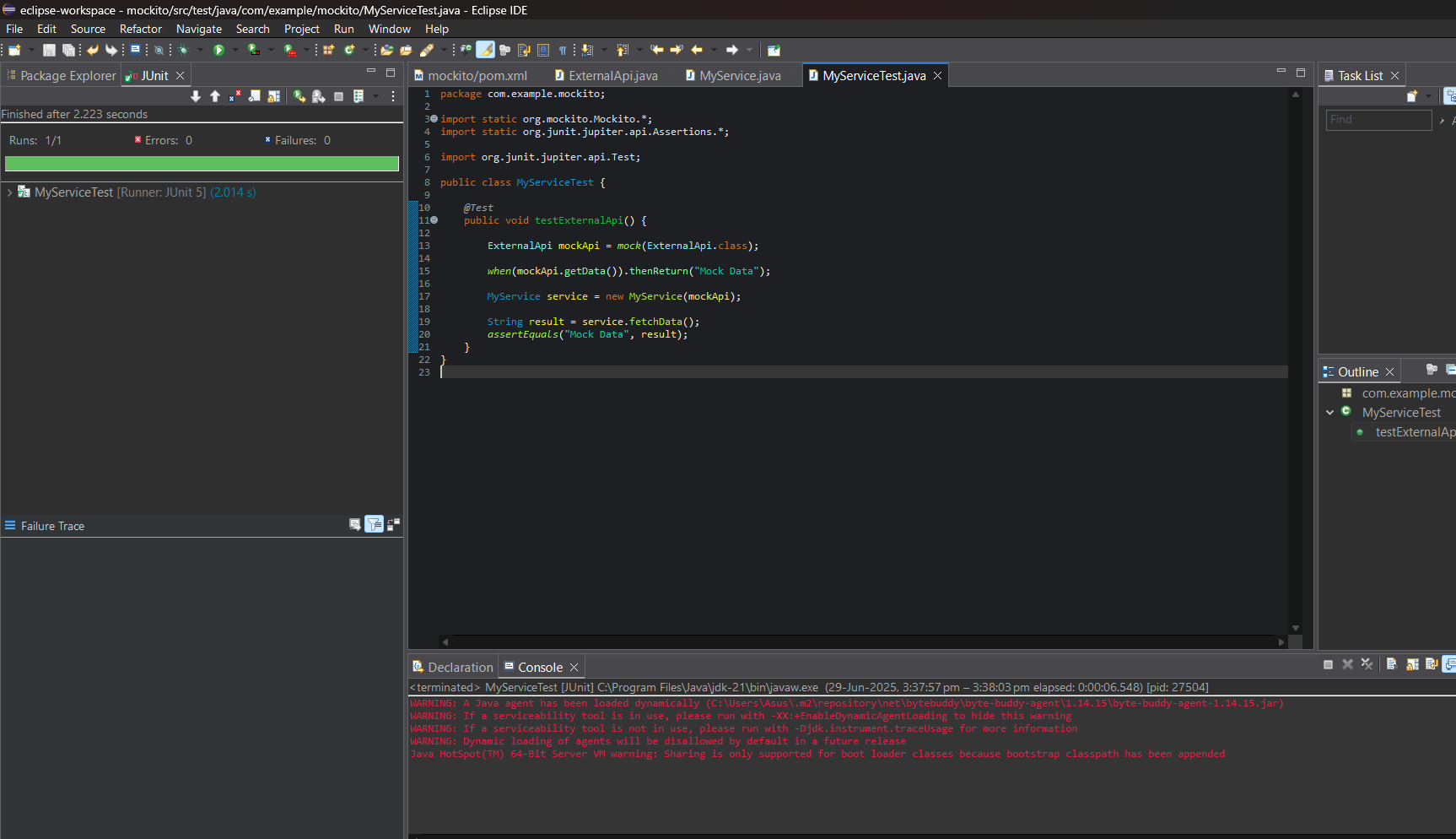
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**OUTPUT:**



# Exercise 2: Verifying Interactions

**ExternalApi.java :**

package com.example.mockito;

public interface ExternalApi {

String getData();

}

**MyService.java :**

package com.example.mockito;

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java :**

package com.example.mockito;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**OUTPUT:**

