**JUnit Testing Exercises**

**Exercise 1: Setting Up JUnit**

package io.javaTesting;

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

import org.junit.jupiter.api.Test;

class MathUtilsTest {

@Test

void test() {

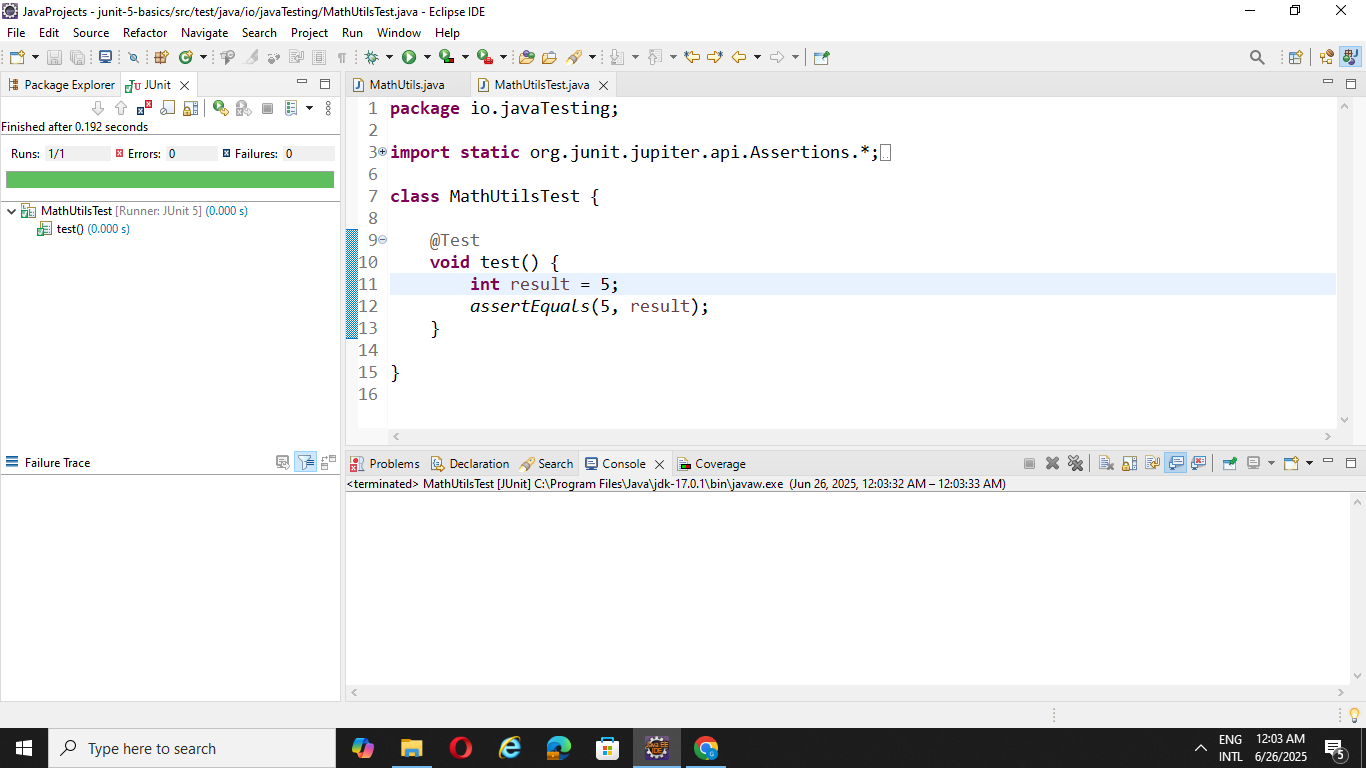
int result = 5;

assertEquals(5, result);

}

}

**OUTPUT:**



**Exercise 3: Assertions in Junit**

package io.javaTesting;

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

import org.junit.jupiter.api.Test;

class AssertionsTest {

@Test

void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 4);

assertFalse(5 < 4);

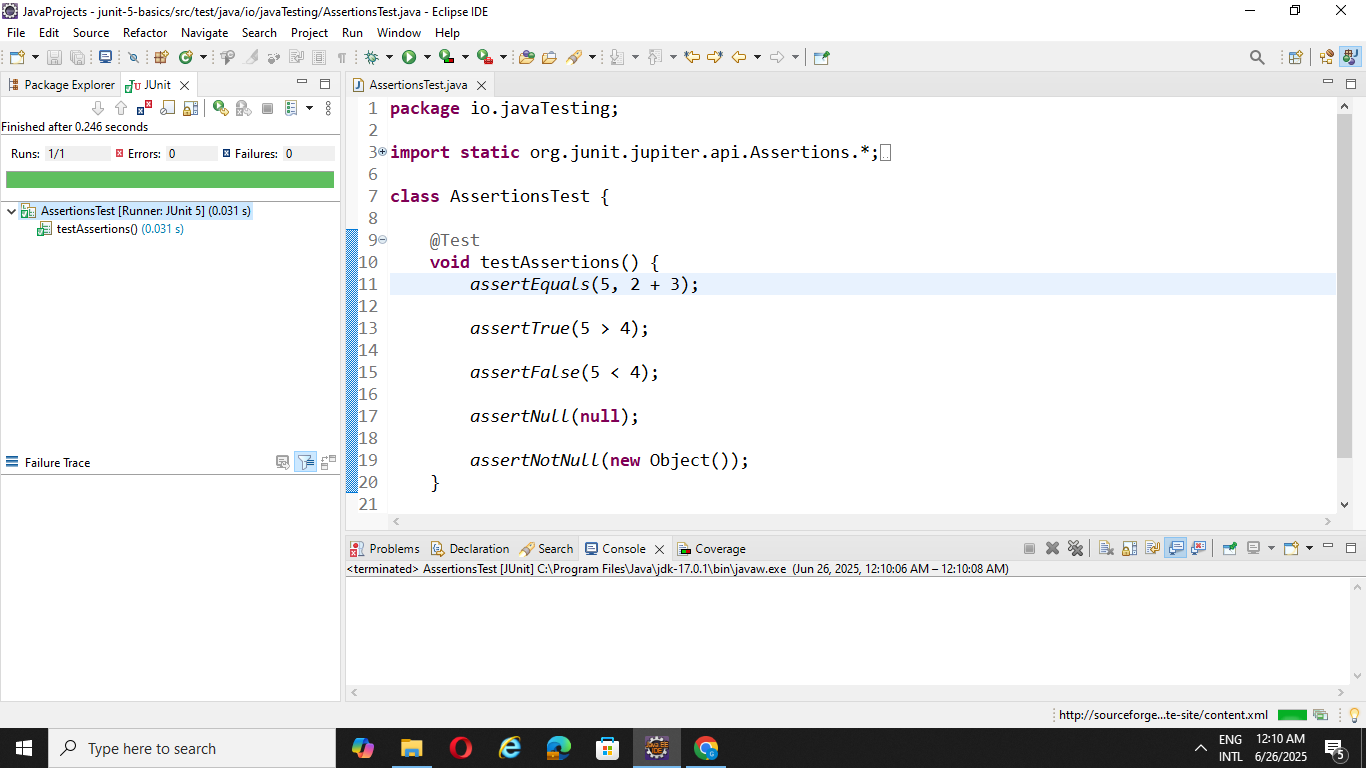
assertNull(null);

assertNotNull(new Object());

}

}

**OUTPUT**



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

package io.javaTesting;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.AfterEach;

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

import org.junit.jupiter.api.Test;

class MathUtilsTest {

private MathUtils math;

@BeforeEach

public void setUp() {

math = new MathUtils();

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

}

@AfterEach

public void tearnDown() {

math = null;

System.out.println("tearnDown - cleaning");

}

@Test

public void testAdd() {

//Arrange

int a = 2, b = 3;

//Act

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

//Assert

assertEquals(5, result);

}

@Test

public void testsquareOfNumber() {

//Arrange

int n = 3;

//Act

int result = math.squareOfNumber(n);

//Assert

assertEquals(9, result);

}

}

package io.javaTesting;

public class MathUtils{

public int add(int a, int b){

return a + b;

}

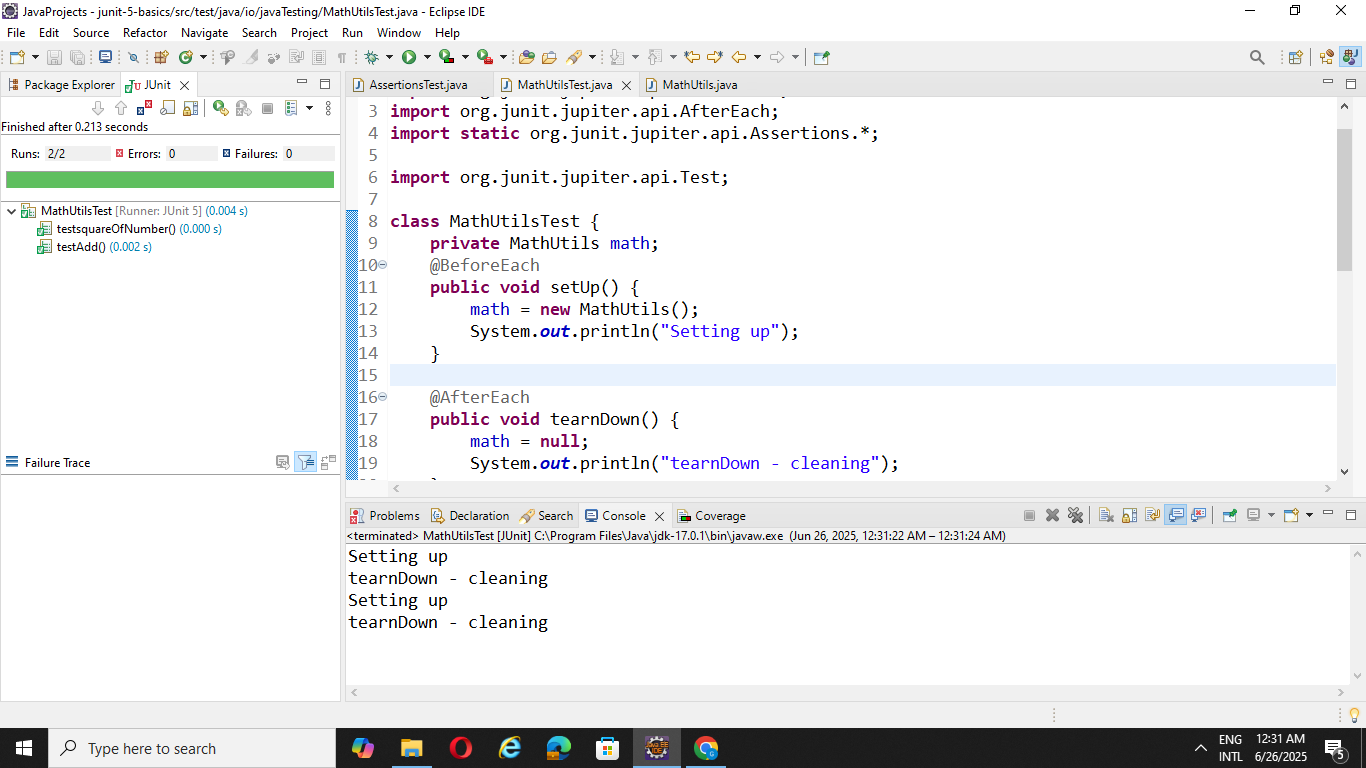
public int squareOfNumber(int n){

return n \* n;

}

}

**OUTPUT:**



**Mockito Hands**

**Exercise 1: Mocking and Stubbing**

package io.javaTesting;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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

public class MyServiceTest {

ExternalApi mockapi;

MyService service;

@BeforeEach

public void setUp() {

mockapi = Mockito.mock(ExternalApi.class);

service = new MyService(mockapi);

}

    @Test

    public void testFetchData() {

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

        String result = service.fetchData();

        assertEquals("Mock Data", result, "Mocking is not working properly");

    }

}

package io.javaTesting;

public class MyService {

ExternalApi extapi;

MyService(ExternalApi extapi) {

this.extapi = extapi;

}

String fetchData() {

return extapi.getData();

}

}

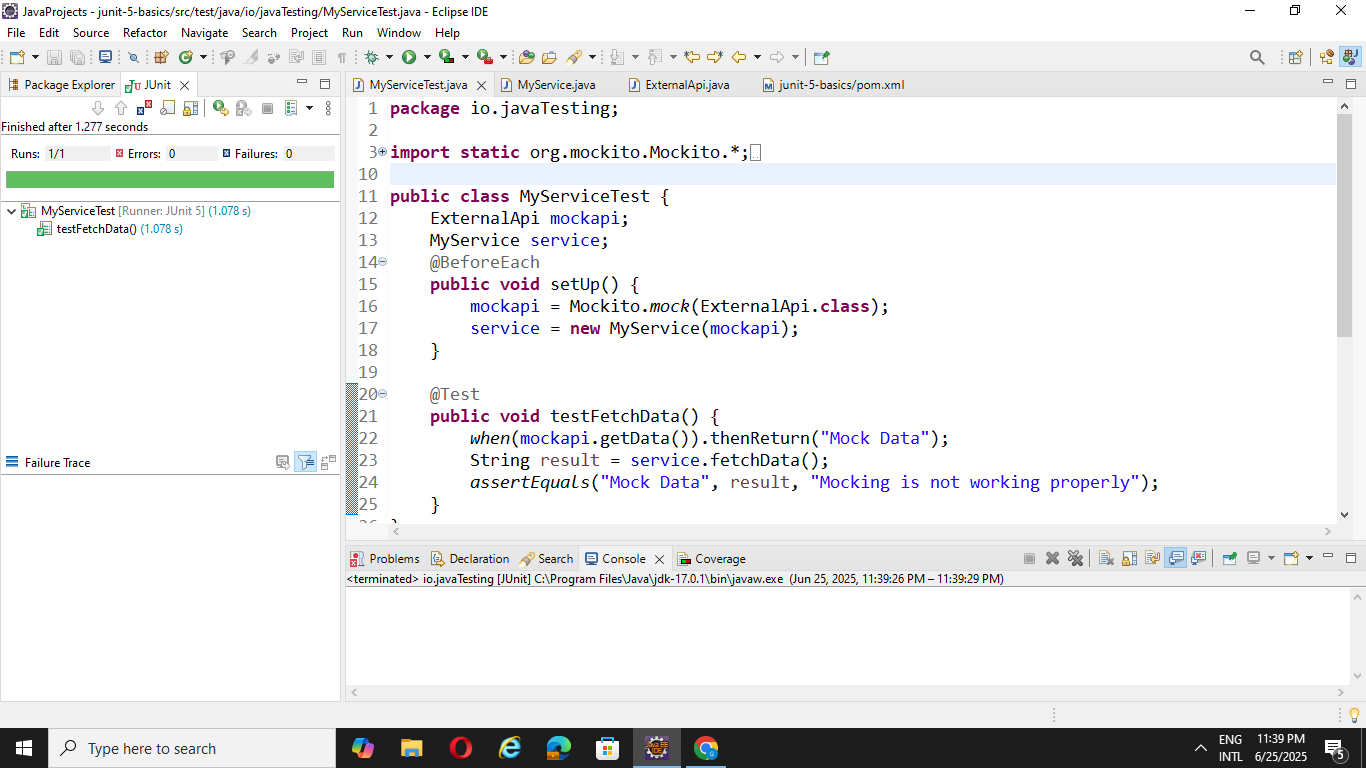
package io.javaTesting;

public interface ExternalApi {

String getData();

}

**OUTPUT:**



**Exercise 2: Verifying Interactions**

package io.javaTesting;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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

public class MyServiceTest {

ExternalApi mockapi;

MyService service;

@BeforeEach

public void setUp() {

mockapi = Mockito.mock(ExternalApi.class);

service = new MyService(mockapi);

}

    @Test

    public void testFetchData() {

     service.fetchData();

     verify(mockapi).getData();

    }

}

package io.javaTesting;

public class MyService {

ExternalApi extapi;

MyService(ExternalApi extapi) {

this.extapi = extapi;

}

String fetchData() {

return extapi.getData();

}

}

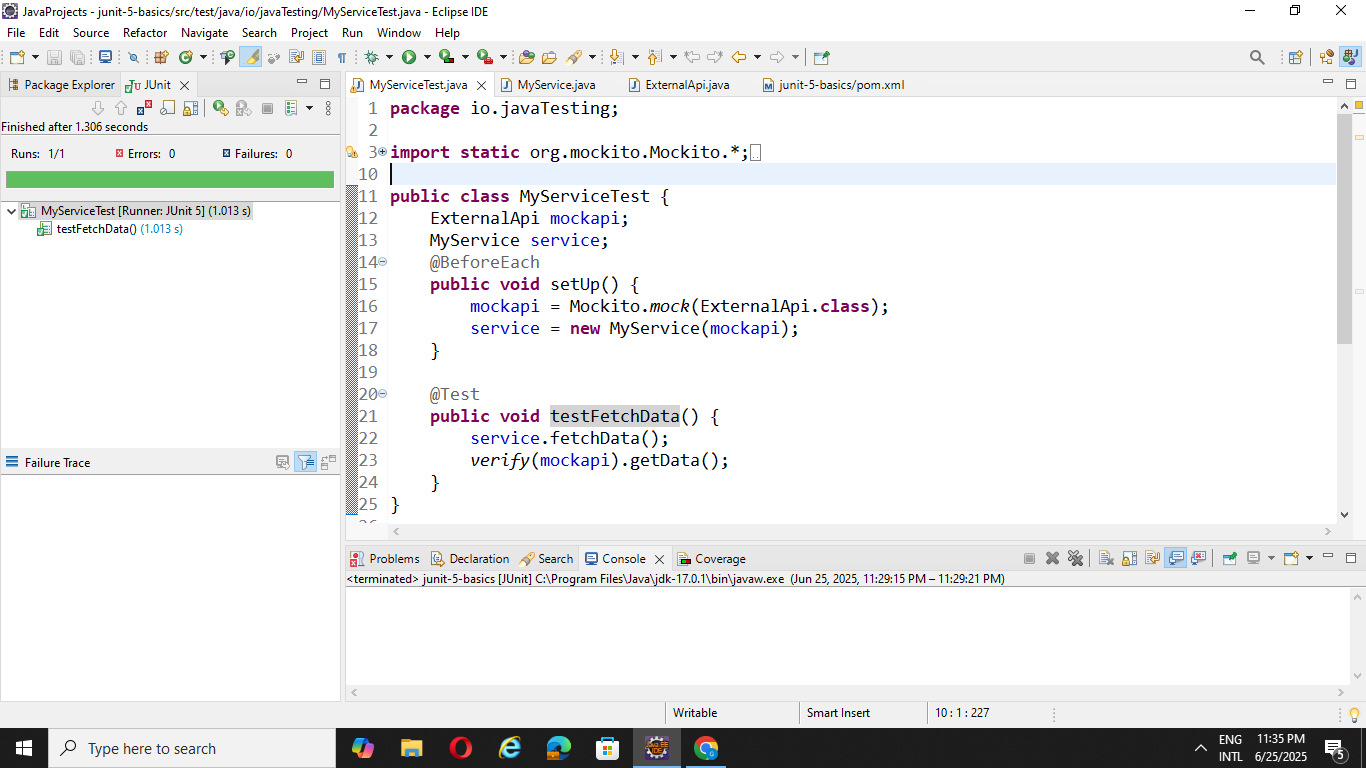
package io.javaTesting;

public interface ExternalApi {

String getData();

}

**OUTPUT:**



**Logging using SLF4J**

**Exercise 1: Logging Error Messages and Warning Levels**

package io.javaTesting;

import org.slf4j.LoggerFactory;

import org.slf4j.Logger;

public class LoggingExample {

private static final Logger log = LoggerFactory.getLogger(LoggingExample.class);

    public static void main(String[] args) {

        log.error("This is an error message");

        log.warn("This is a warning message");

    }

}

**OUTPUT:**

