**JUnit\_Basic Testing Exercises**

Exercise 1: Setting Up JUnit

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}}

package com.example;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

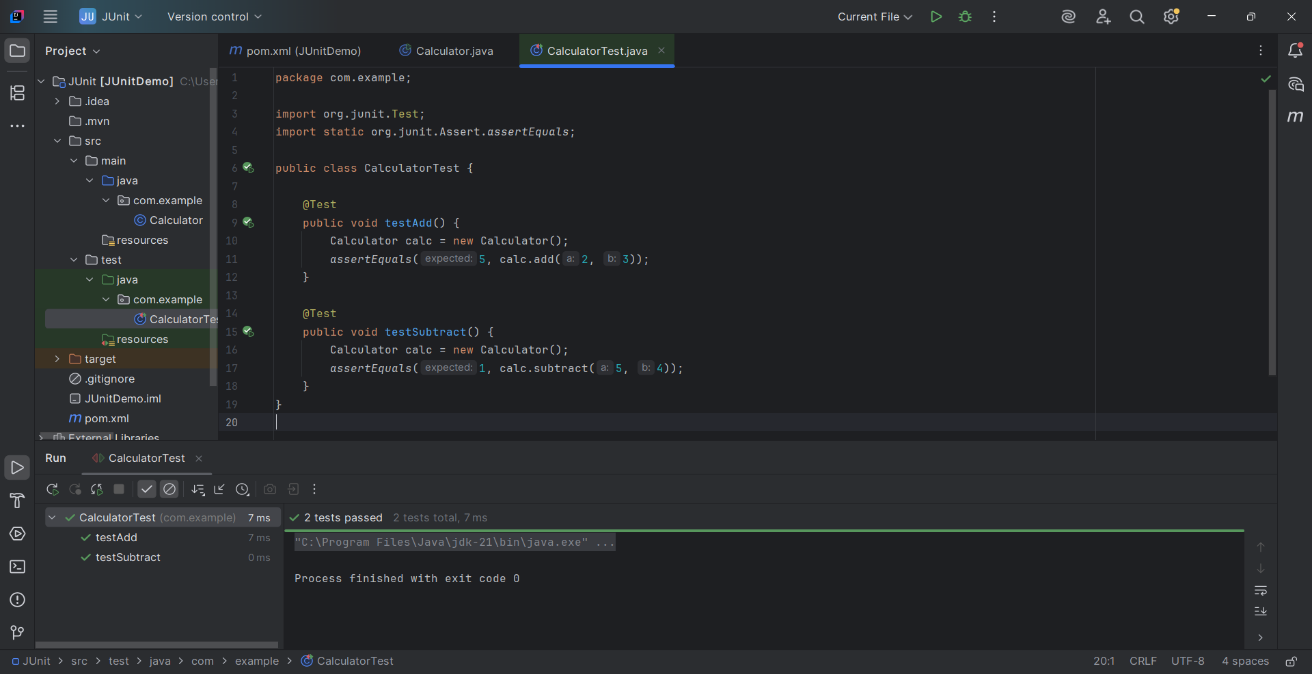
@Test

public void testAdd() {

Calculator calc = new Calculator();

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

assertEquals(5, result);}}



**JUnit\_Basic Testing Exercises**   
Exercise 3: Assertions in JUnit

package com.example;

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);

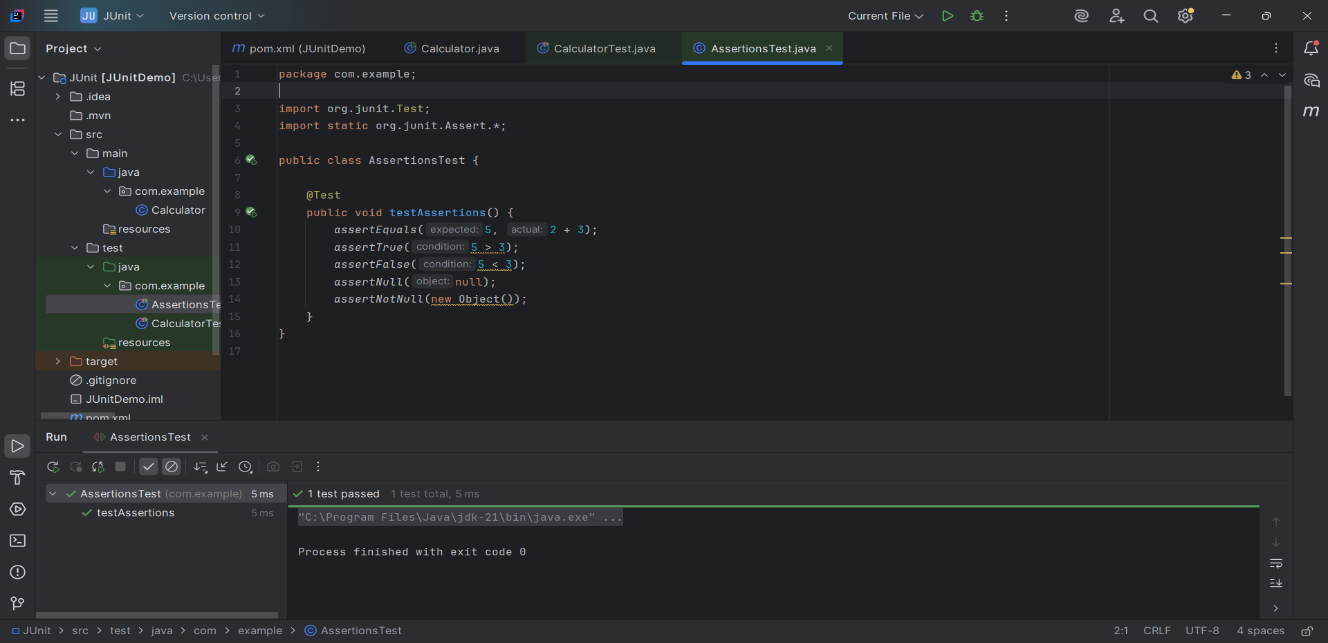
// Assert null

assertNull(null);

// Assert not null

assertNotNull(new Object());

}}



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

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}}

package com.example;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

// Setup – runs before every test

@Before

public void setUp() {

calculator = new Calculator();

System.out.println("🔧 setUp() called");

}

// Teardown – runs after every test

@After

public void tearDown() {

System.out.println("🧹 tearDown() called");

calculator = null;

}

@Test

public void testAddition() {

// Arrange

int a = 2;

int b = 3;

// Act

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

// Assert

assertEquals(5, result);

}

@Test

public void testMultiplication() {

// Arrange

int a = 4;

int b = 5;

// Act

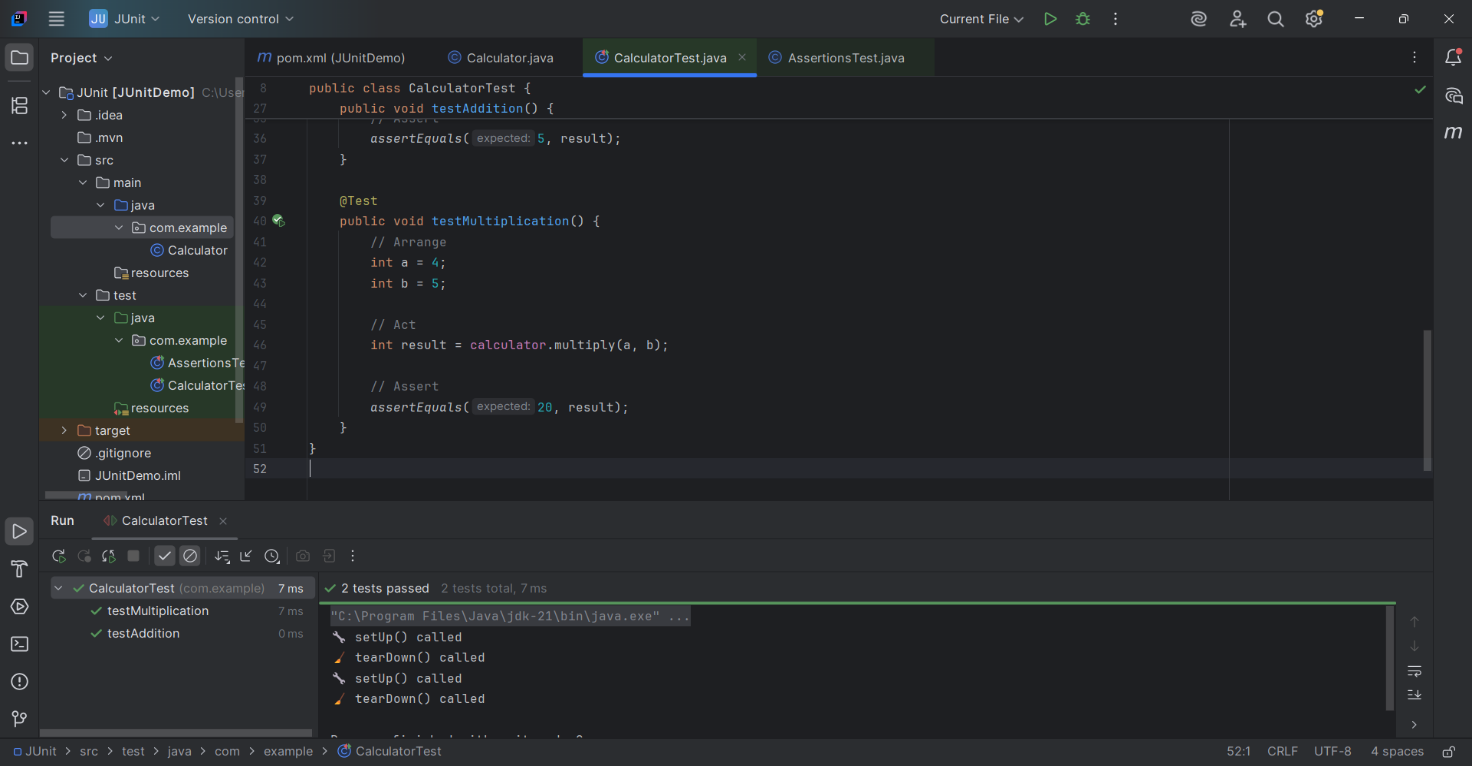
int result = calculator.multiply(a, b);

// Assert

assertEquals(20, result);

}

}



**Mockito exercises**

Exercise 1: Mocking and Stubbing

package com.example;

public interface ExternalApi {

String getData();

}

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); // Uses the mocked method

}

}

package com.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class);

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

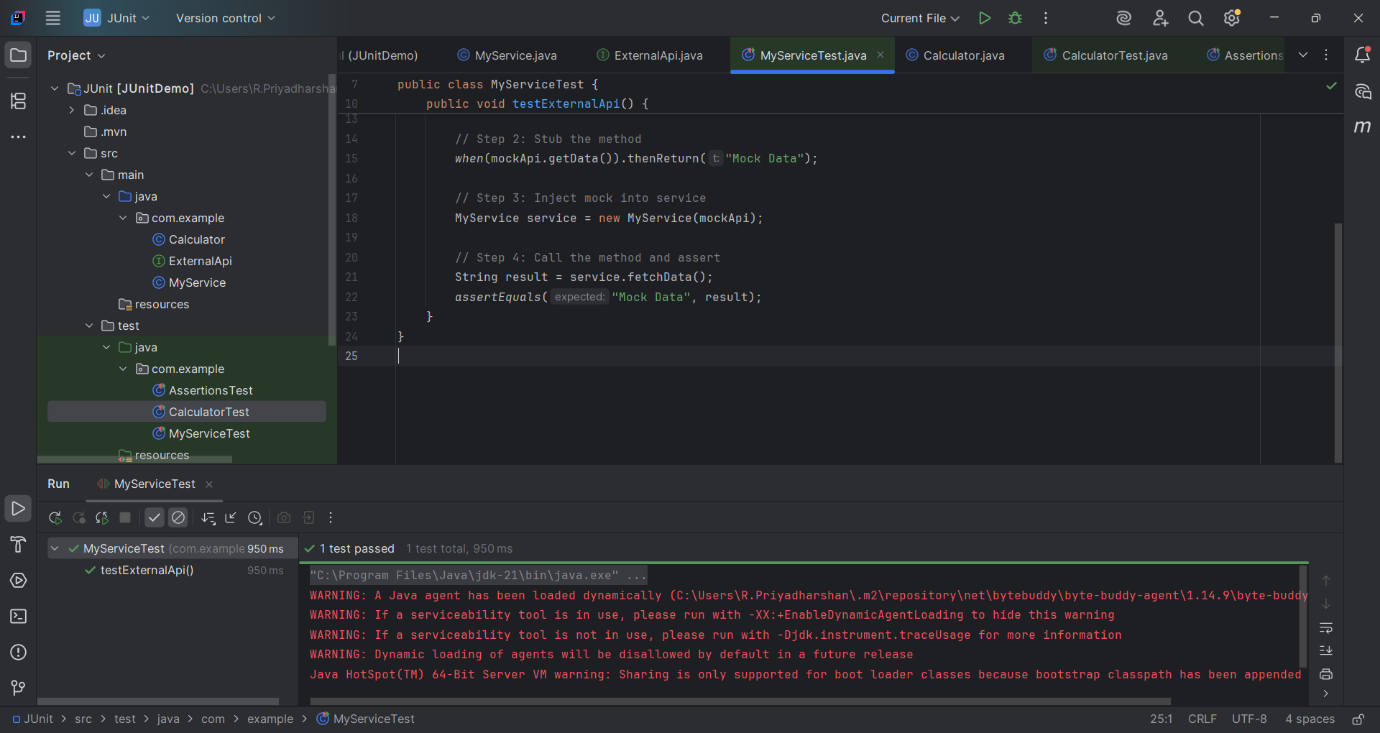
MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}



**Mockito exercises**   
  
Exercise 2: Verifying Interactions

public interface ExternalApi {

String getData();

}

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); // This method must call getData()

}

}

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(); // Step 2: Call the method

verify(mockApi).getData(); // Step 3: Verify interaction

}

}

