**NAME OF THE QUESTION**

**Exercise 1:** Setting Up JUnit

**CODE**

**Calculator.java**

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculaterTest.java**

package com.example;

import org.junit.jupiter.api.Test;

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

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

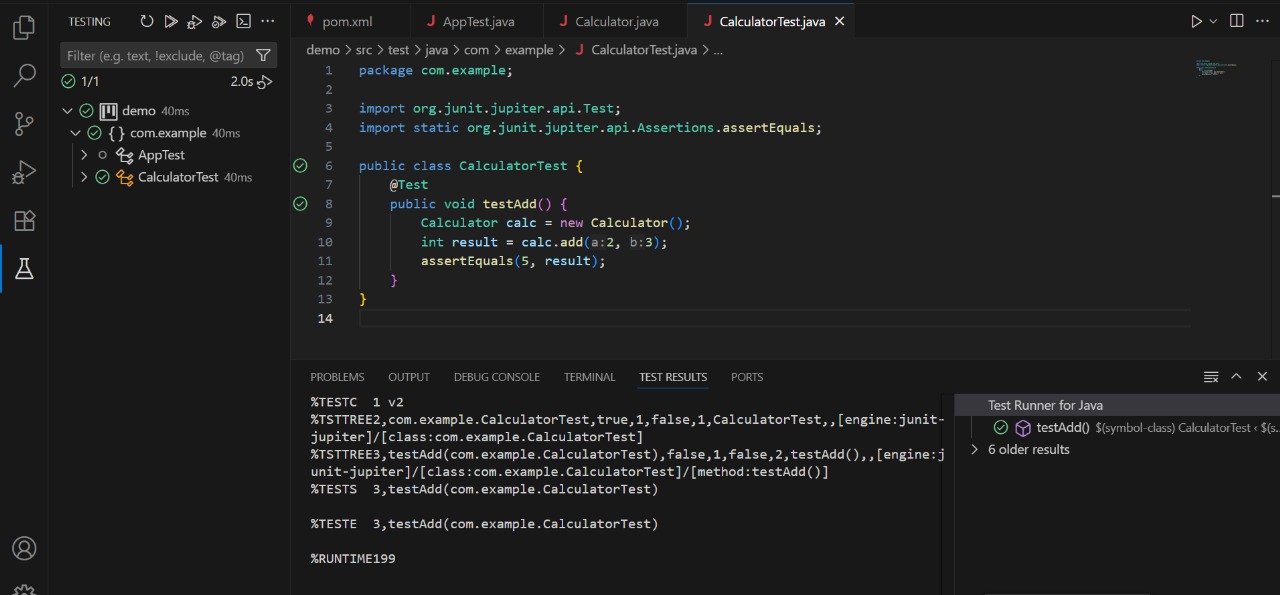
int result = calc.add(2, 3);

assertEquals(5, result);

}

}

**OUTPUT**

****

**NAME OF THE QUESTION**

**Exercise 3:** Assertions in JUnit

**CODE**

package com.example;

import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

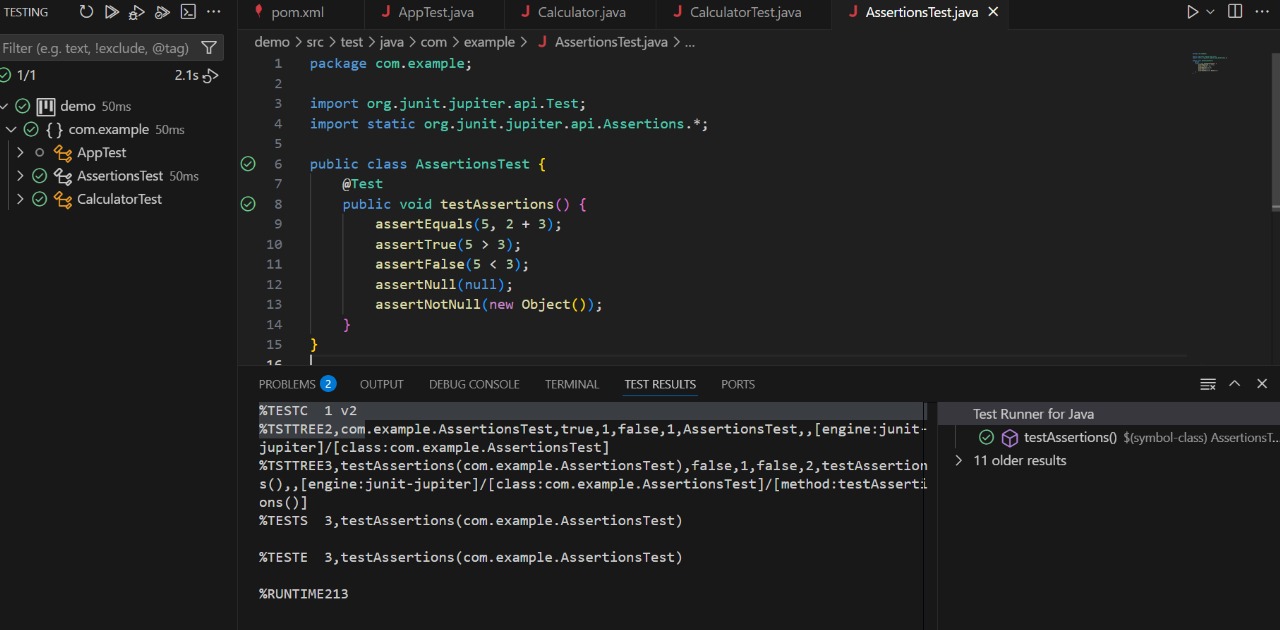
assertNull(null);

assertNotNull(new Object());

}

}

**OUTPUT**



**NAME OF THE QUESTION:**

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

**CODE:**

**paymentProcessor:**

package com.example;

public interface PaymentProcessor {

boolean processPayment(double amount);

}

**PayPalAdapter:**

package com.example;

public class PayPalAdapter implements PaymentProcessor {

private PayPalGateway payPalGateway;

public PayPalAdapter(PayPalGateway payPalGateway) {

this.payPalGateway = payPalGateway;

}

@Override

public boolean processPayment(double amount) {

return payPalGateway.sendPayment(amount);

}

}

**PayPalGateway:**

package com.example;

public class PayPalGateway {

public boolean sendPayment(double amount) {

System.out.println("PayPalGateway: Sending payment of ₹" + amount);

return true;

}

}

**StripeAdapter:**

package com.example;

public class StripeAdapter implements PaymentProcessor {

private StripeGateway stripeGateway;

public StripeAdapter(StripeGateway stripeGateway) {

this.stripeGateway = stripeGateway;

} @Override

public boolean processPayment(double amount) {

return stripeGateway.makePayment(amount);

}

}

**StripeGateway:**

package com.example;

public class StripeGateway {

public boolean makePayment(double amount) {

System.out.println("StripeGateway: Processing payment of ₹" + amount);

return true;

}

}

class PaymentTest:

package com.example;

public class PaymentTest {

public static void main(String[] args) {

PaymentProcessor stripe = new StripeAdapter(new StripeGateway());

stripe.processPayment(1000.0);

PaymentProcessor paypal = new PayPalAdapter(new PayPalGateway());

paypal.processPayment(2000.0);

}

}

**PaymentTest:**

package com.example;

public class PaymentTest {

public static void main(String[] args) {

PaymentProcessor stripeProcessor = new StripeAdapter(new StripeGateway());

PaymentProcessor paypalProcessor = new PayPalAdapter(new PayPalGateway());

stripeProcessor.processPayment(200.0);

paypalProcessor.processPayment(89.50);

}

}

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class PaymentAdapterTest {

@Test

public void testStripeAdapter() {

PaymentProcessor processor = new StripeAdapter(new StripeGateway());

assertTrue(processor.processPayment(500.0));

}

@Test

public void testPayPalAdapter() {

PaymentProcessor processor = new PayPalAdapter(new PayPalGateway());

assertTrue(processor.processPayment(1500.0));

}

}

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

**NAME OF THE QUESTION:  
Exercise 1:** Mocking and Stubbing

**CODE:**

**MyService.java:**

package com.example;

public class MyService {

private ExternalApi api; public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**ExternalApi.java:**

package com.example;

public interface ExternalApi {

String getData();

}

Test:

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class);

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

MyService service = new MyService(mockApi);

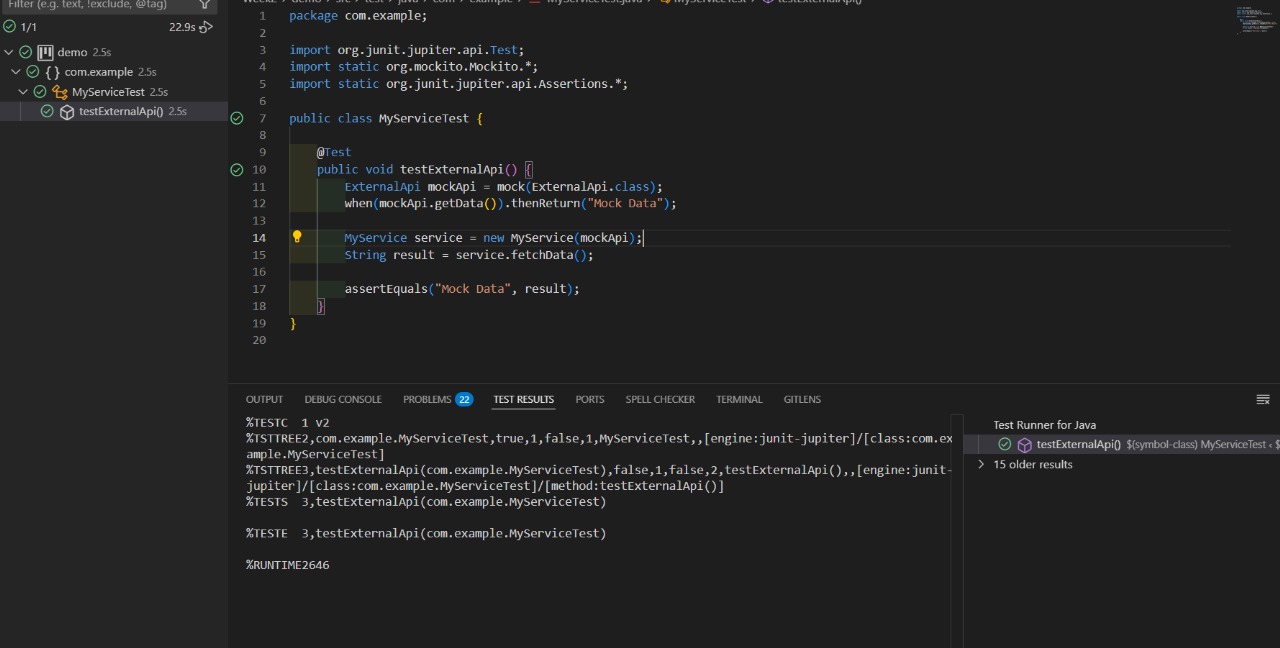
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**OUTPUT**



**NAME OF THE QUESTION**

**Exercise 2:** Verifying Interactions

**CODE:**

public interface ExternalApi {

String getData();

}

**MyService.java:**

public class MyService {

private final ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public void fetchData() {

String data = externalApi.getData();

System.out.println("Fetched: " + data);

}

}

TEST:

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

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

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**OUTPUT**

A screenshot of a computer screen

AI-generated content may be incorrect.