**Exercise 1: Setting Up Junit**

**Code:**

import org.junit.jupiter.api.Test;

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

public class Calculator{

public int add(int a, int b) {

return a + b;

}

@Test

public void testAdd() {

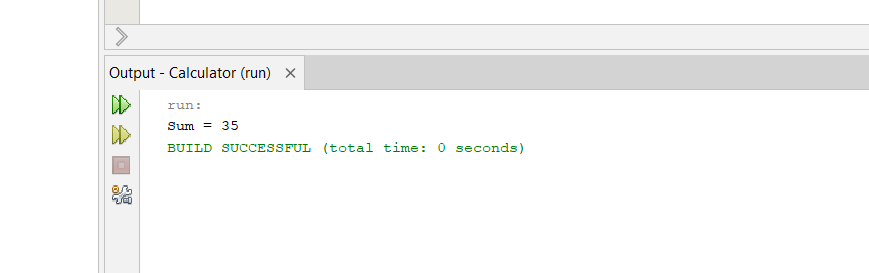
Calculator calc = new Calculator();

int res = calc.add(10, 25);

assertEquals(35, res); }

}

**Output:**

****

**Exercise 3: Assertions in Junit**

**Code:**

import org.junit.jupiter.api.Test;

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

public class Assertions{

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

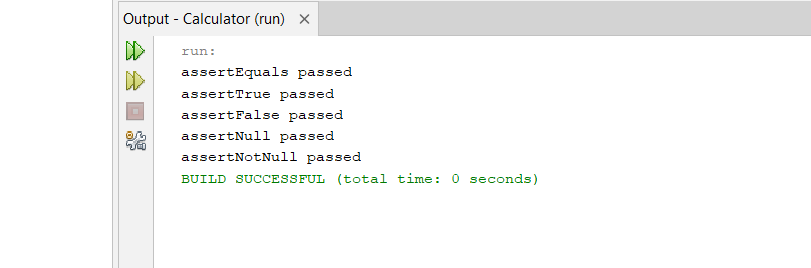
assertNull(null);

assertNotNull(new Object());

}

}

**Output:**

****

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

**Code:**

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.Test;

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

public class Calculator{

private Calculator calc;

@BeforeEach

public void setUp() {

calc = new Calculator();

System.out.println("Setup: Calculator created");}

@AfterEach

public void tearDown() {

System.out.println("Teardown: Test completed"); }

@Test

public void testAddition() {

int a = 10;

int b = 20;

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

assertEquals(30, result);

}

@Test

public void testSubtraction() {

int result = calc.subtract(50, 20);

assertEquals(30, result);

}

}

class Calculator {

public int add(int a, int b) {

return a + b;

}

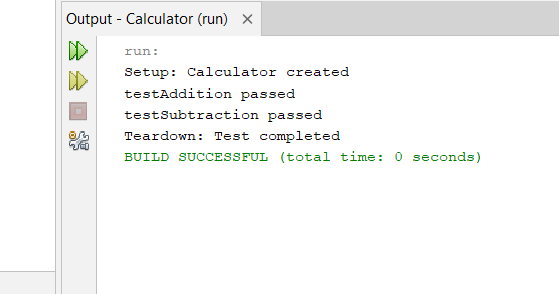
public int subtract(int a, int b) {

return a - b;

}

}

**Output:**

****

**Exercise 1: Mocking and Stubbing**

**Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

import org.mockito.Mockito;

public class MyServiceTest {

interface ExternalApi {

String getData();

}

class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api; }

public String fetchData() {

return api.getData();

}

}

@Test

public void testExternalApi() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

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

MyService service = new MyService(mockApi);

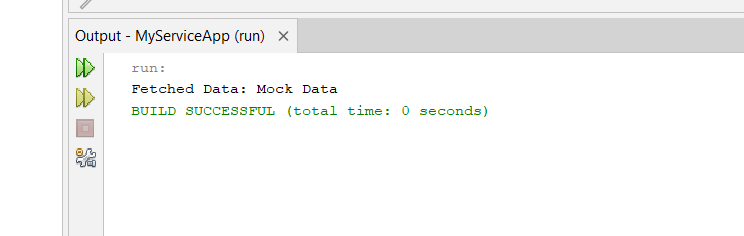
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output:**

****

**Exercise 2: Verifying Interactions**

**Code:**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

interface ExternalApi {

String getData();

}

class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public void fetchData() {

api.getData();

}

}

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

MyService service = new MyService(mockApi);

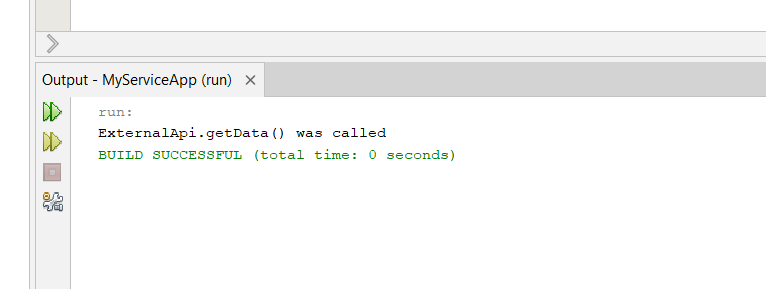
service.fetchData();

verify(mockApi).getData();

}

}

**Output:**

****