Exercise 1: Setting Up JUnit

pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

App.java

package com.example;

public class App {

public int add(int a, int b) {

return a + b;

}

}

AppTest.java

package com.example;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AppTest {

*@Test*

public void testAdd() {

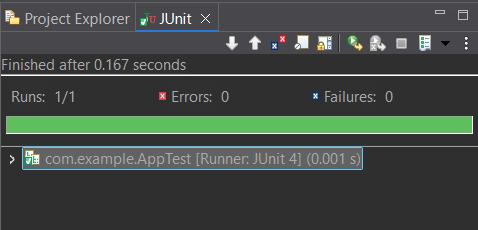
App app = new App();

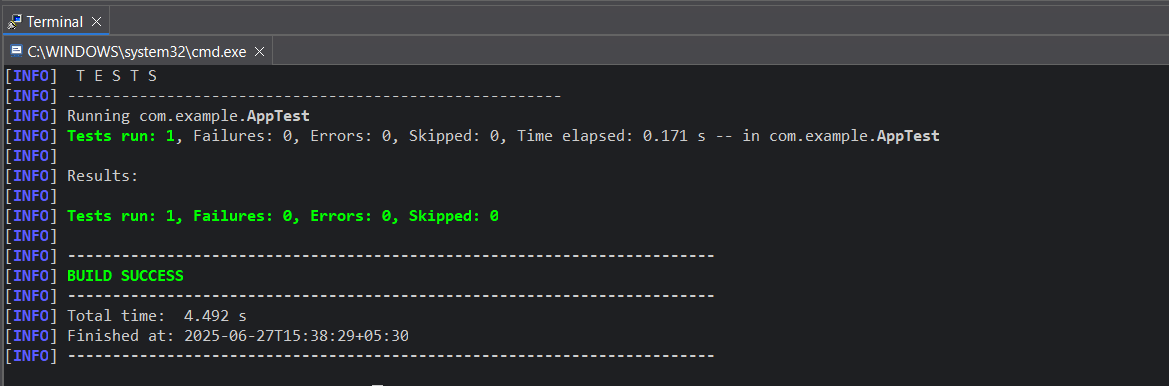
*assertEquals*(5, app.add(2, 3));

}

}

Output :





Exercise 3: Assertions in JUnit

AssertionsTest.java

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);

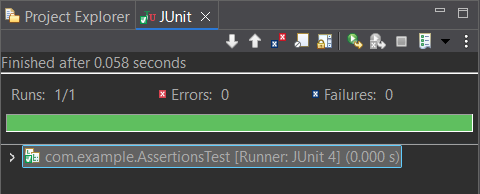
// Assert not null

*assertNotNull*(new Object());

}

}

Output :



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

Calculator.java

package com.example;

public class Calculator {

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

return a / b;

}

}

CalculatorTest.java

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 each test

*@Before*

public void setUp() {

calculator = new Calculator();

System.***out***.println("Setup: Calculator initialized");

}

// Teardown - runs after each test

*@After*

public void tearDown() {

calculator = null;

System.***out***.println("Teardown: Calculator cleaned up");

}

*@Test*

public void testMultiply() {

// Arrange is already done in setUp()

// Act

int result = calculator.multiply(4, 5);

// Assert

*assertEquals*(20, result);

}

*@Test*

public void testDivide() {

// Act

int result = calculator.divide(10, 2);

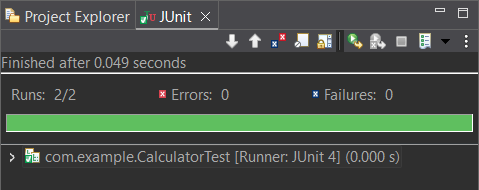
// Assert

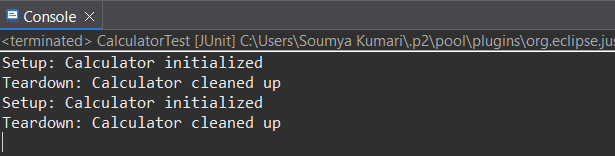
*assertEquals*(5, result);

}

}

Output :





Mockito exercises

Exercise 1: Mocking and Stubbing

pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitDemon</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 (Jupiter API and Engine) -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<!-- Mockito Core for mocking/stubbing -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Surefire Plugin to run JUnit 5 tests -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.1.2</version>

<configuration>

<useModulePath>false</useModulePath>

</configuration>

</plugin>

</plugins>

</build>

</project>

ExternalApi.java

package com.example;

public interface ExternalApi {

String getData();

}

MyService.java

package com.example;

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

MyServiceTest.java

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() {

// Step 1: Create mock

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

// Step 2: Stub method

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

// Step 3: Inject mock into service

MyService service = new MyService(mockApi);

// Step 4: Act and Assert

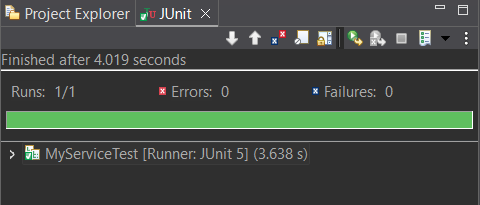
String result = service.fetchData();

*assertEquals*("Mock Data", result);

}

}

Output :



Exercise 2: Verifying Interactions

MyServiceTest.java

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

// Step 1: Create a mock

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

// Step 2: Inject into service

MyService service = new MyService(mockApi);

// Step 3: Call the method

service.fetchData();

// Step 4: Verify interaction with the mock

*verify*(mockApi).getData(); // Verifies it was called

}

}

Output :

