JUnit\_Basic Testing Exercises (Mandatory Hands On)

Exercise 1: Setting Up Junit

Code:

package com.example.junit;

import static org.junit.Assert.*assertEquals*;

import org.junit.Test;

public class CalculatorTest {

public int add(int a, int b) {

return a + b;

}

*@Test*

public void testAddition() {

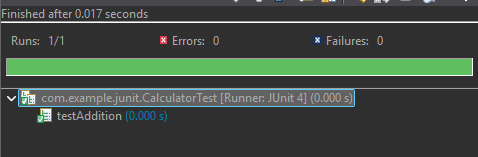
int result = add(2, 3);

*assertEquals*(5, result); // this test will pass

}

}

Output:



Exercise 3: Assertions in Junit

Code:

package com.example.junit;

import static org.junit.Assert.\*; // Required for assertions

import org.junit.Test;

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);

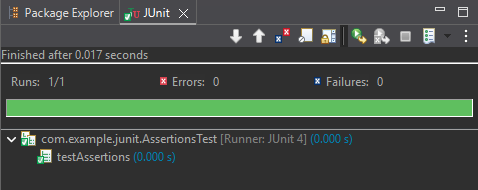
// Assert not null

*assertNotNull*(new Object());

}

}

Output:



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

Teardown Methods in Junit

Code:

package com.example.junit;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class CalculatorTest {

private Calculator calculator;

// Setup method (executed before each test)

*@Before*

public void setUp() {

calculator = new Calculator();

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

}

// Teardown method (executed after each test)

*@After*

public void tearDown() {

calculator = null;

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

}

*@Test*

public void testAddition() {

// Arrange is handled by @Before

// Act

int result = calculator.add(5, 3);

// Assert

*assertEquals*(8, result);

}

*@Test*

public void testSubtraction() {

// Act

int result = calculator.subtract(10, 4);

// Assert

*assertEquals*(6, result);

}

}

package com.example.junit;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

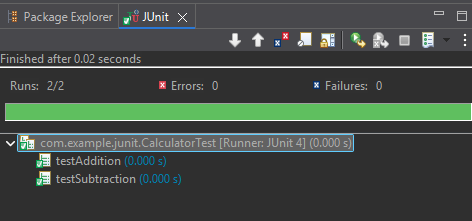
public int subtract(int a, int b) {

return a - b;

}

}

Output:



Mockito exercises (Mandatory Hands On)

Exercise 1: Mocking and Stubbing

Code:

package com.mockdemo.service;

public interface ExternalApi {

String getData();

}

package com.mockdemo.service;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

package com.mockdemo.test;

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import com.mockdemo.service.ExternalApi;

import com.mockdemo.service.MyService;

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);

}

}

<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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.mockdemo</groupId>

<artifactId>MockingExample</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.0.0-M9</version>

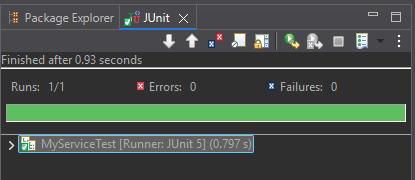
</plugin>

</plugins>

</build>

</project>

Output:



Exercise 2: Verifying Interactions

Code:

package com.mockdemo.service;

public interface ExternalApi {

String getData();

}

package com.mockdemo.service;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

package com.mockdemo.test;

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import com.mockdemo.service.ExternalApi;

import com.mockdemo.service.MyService;

public class MyServiceTest {

*@Test*

public void testVerifyInteraction() {

// Step 1: Create mock

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

// Step 2: Use service with mock

MyService service = new MyService(mockApi);

service.fetchData(); // this should call mockApi.getData()

// Step 3: Verify interaction

*verify*(mockApi).getData();

}

}

<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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.mockdemo</groupId>

<artifactId>MockingExample</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.0.0-M9</version>

</plugin>

</plugins>

</build>

</project>

Output:

