**TEST DRIVEN DEVELOPMENT AND LOGGING FRAMEWORK**

**JUNIT BASIC TESTING**

**EXERCISE – 1 : SETTING UP JUNIT:**

**Steps for setting up JUNIT in a Maven project**

1. Create a Maven Project - Go to **File → New → Maven Project**.
2. Add Junit Dependency in pom.xml
   1. Add the following inside the <dependencies> tag:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

2.2 Save the pom.xml.

**AnagramChecker.java:**

**package** com.example;

**import** java.util.Arrays;

**public** **class** anagramChecker {

**public** **static** **boolean** isAnagram(String str1, String str2) {

**if** (str1 == **null** || str2 == **null**) {

**return** **false**;

}

**char**[] arr1 = str1.replaceAll("\\s", "").toLowerCase().toCharArray();

**char**[] arr2 = str2.replaceAll("\\s", "").toLowerCase().toCharArray();

Arrays.*sort*(arr1);

Arrays.*sort*(arr2);

**return** Arrays.*equals*(arr1, arr2);

}

}

**anagramCheckerTest.java:**

**package** com.example;

**import** org.junit.Test;

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

**public** **class** anagramCheckerTest {

@Test

**public** **void** testIsAnagram\_Positive() {

*assertTrue*(anagramChecker.*isAnagram*("listen", "silent"));

*assertTrue*(anagramChecker.*isAnagram*("Triangle", "Integral"));

*assertTrue*(anagramChecker.*isAnagram*("Dormitory", "Dirty room"));

}

@Test

**public** **void** testIsAnagram\_Negative() {

*assertFalse*(anagramChecker.*isAnagram*("hello", "world"));

*assertFalse*(anagramChecker.*isAnagram*("test", "best"));

}

@Test

**public** **void** testIsAnagram\_Null() {

*assertFalse*(anagramChecker.*isAnagram*(**null**, "test"));

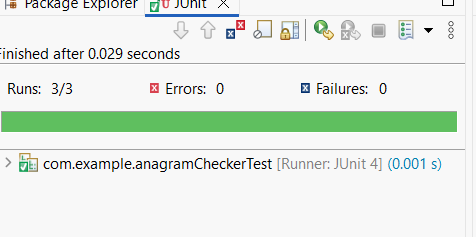
*assertFalse*(anagramChecker.*isAnagram*("test", **null**));

}

}

**OUTPUT:**

1. Right click on the ‘anagramCheckerTest.java’ file.
2. Now select Run As -> JUnit test



**EXERCISE – 3: ASSERTIONS IN JUNIT:**

**Assertions.java:**

**package** com.example;

**import** org.junit.Test;

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

**public** **class** AssertionsTest {

@Test

**public** **void** testAssertions() {

// assertEquals: expected == actual

*assertEquals*(5, 2 + 3);

// assertTrue: condition is true

*assertTrue*(5 > 3);

// assertFalse: condition is false

*assertFalse*(5 < 3);

// assertNull: value is null

String s = **null**;

*assertNull*(s);

// assertNotNull: value is not null

String t = "Hello";

*assertNotNull*(t);

// assertSame: same object reference

Object obj = **new** Object();

Object sameObj = obj;

*assertSame*(obj, sameObj);

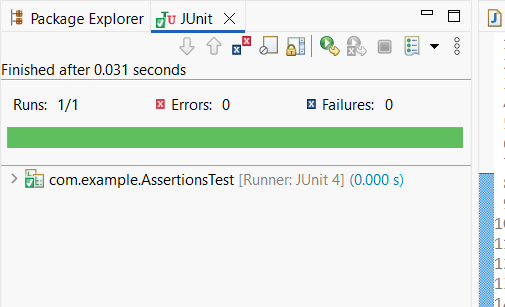
// assertNotSame: different object references

*assertNotSame*(**new** Object(), **new** Object());

}

}

**OUTPUT:**

1. Right click on the ‘AssertionsTest.java’ file.
2. Now select Run As -> JUnit test

**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** add(**int** a, **int** b) {

**return** a + b;

}

**public** **int** subtract(**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;

@Before

**public** **void** setUp() {

// Arrange - Setup

System.***out***.println("Running setup...");

calculator = **new** Calculator();

}

@After

**public** **void** tearDown() {

System.***out***.println("Running teardown...");

calculator = **null**;

}

@Test

**public** **void** testAdd() {

// Act

**int** result = calculator.add(2, 3);

// Assert

*assertEquals*(5, result);

}

@Test

**public** **void** testSubtract() {

// Act

**int** result = calculator.subtract(5, 3);

// Assert

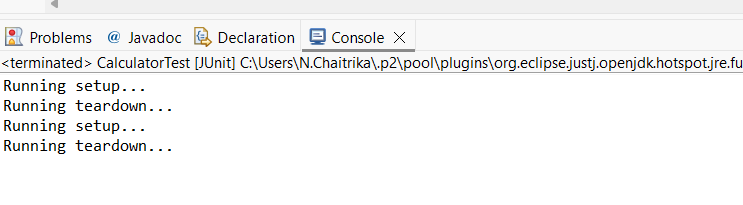
*assertEquals*(2, result);

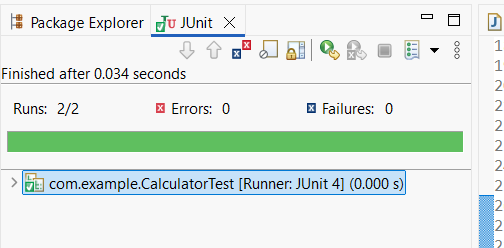
}

}

**OUTPUT:**

1. Right click on the ‘CalculatorTest.java’ file.
2. Now select Run As -> JUnit test





**MOCKITO EXERCISES**

**EXERCISE – 1: MOCKING AND STUBBING**

1.Create a new Maven project in Eclipse.

2.Now add JUnit 5 and Mockito dependencies to pom.xml

**pom.xml**

<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>

3.Create java classes

**ExternalApi.java**

ExternalApi

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.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

private ExternalApi mockApi;

private MyService service;

@Before

public void setUp() {

mockApi = mock(ExternalApi.class);

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

service = new MyService(mockApi);

}

@Test

public void testFetchDataReturnsMockData() {

// Act

String result = service.fetchData();

// Assert

assertEquals("Mock Data", result);

// Optional verification

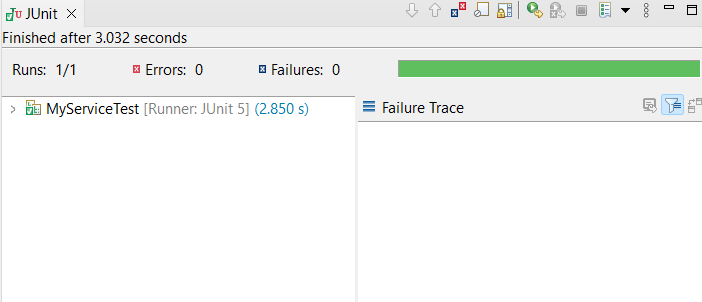
verify(mockApi).getData();

}

}

**OUTPUT:**

1. Right click on the ‘MyServiceTest.java’ file.
2. Now select Run As -> JUnit test



**EXERCISE – 2: VERIFYING INTERACTIONS**

1.Create a new Maven project in Eclipse.

2.Now add JUnit 5 and Mockito dependencies to pom.xml

**pom.xml**

<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>

3.Create java classes

**ExternalApi.java**

**package** com.example;

**public** **interface** ExternalApi {

String getData(String userId);

}

**MyService.java**

**package** com.example;

**public** **class** MyService {

**private** ExternalApi externalApi;

**public** MyService(ExternalApi externalApi) {

**this**.externalApi = externalApi;

}

**public** String fetchData(String userId) {

**return** externalApi.getData(userId);

}

}

**MyServiceTest.java**

**package** com.example;

**import** org.junit.Before;

**import** org.junit.Test;

**import** **static** org.mockito.Mockito.\*;

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

**public** **class** MyServiceTest {

**private** ExternalApi mockApi;

**private** MyService service;

@Before

**public** **void** setUp() {

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

service = **new** MyService(mockApi);

}

@Test

**public** **void** testVerifyInteractionWithSpecificArgument() {

String testUserId = "user123";

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

String result = service.fetchData(testUserId);

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

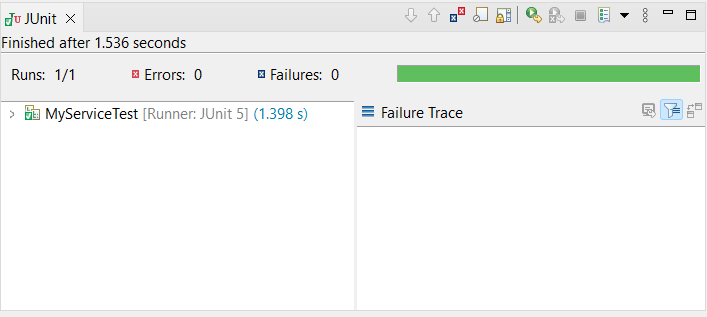
*verify*(mockApi).getData("user123");

}

}

**OUTPUT:**

1. Right click on the ‘MyServiceTest.java’ file.
2. Now select Run As -> JUnit test



**LOGGING USING SLF4J**

**EXERCISE – 1 : LOGGING ERROR MESSAGES AND WARNING MESSAGES**

1.Create a new Maven project in Eclipse.

2.Now add SLF4J dependencies to pom.xml

**pom.xml**

<dependencies>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

3.Create java classes

**LoggingExample.java**

**package** com.example;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** LoggingExample {

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(LoggingExample.**class**);

**public** **static** **void** main(String[] args) {

***logger***.error("This is an error message");

***logger***.warn("This is a warning message");

***logger***.info("This is an info message");

***logger***.debug("This is a debug message");

***logger***.trace("This is a trace message");

}

}

**OUTPUT:**

1. Right click on the ‘LoggingExample.java’ file.
2. Now select Run As -> JUnit test

