**TDD USING JUNIT5 AND MOCKITO**

**Exercise 1: Setting Up Junit**

* Created a Java project named JUnitExample in Eclipse and added JUnit 5 via Build Path → Add Libraries → JUnit 5.
* Created two packages: JUnitSetup for the main class and JUnitTest for the test class.
* Write unit tests in the test class using @Test from JUnit 5.
* Run the test by right-clicking inside the test class → Run As → JUnit Test, and verify the results using the green bar in the JUnit panel.

**MathUtils.java:**

package JUnitSetup;

public class MathUtils {

public int multiply(int a, int b) {

return a \* b;

}

}

**MathUtilsTest.java:**

package JunitTest;

import org.junit.jupiter.api.Test;

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

import JUnitSetup.MathUtils;

public class MathUtilsTest {

@Test

void testMultiply() {

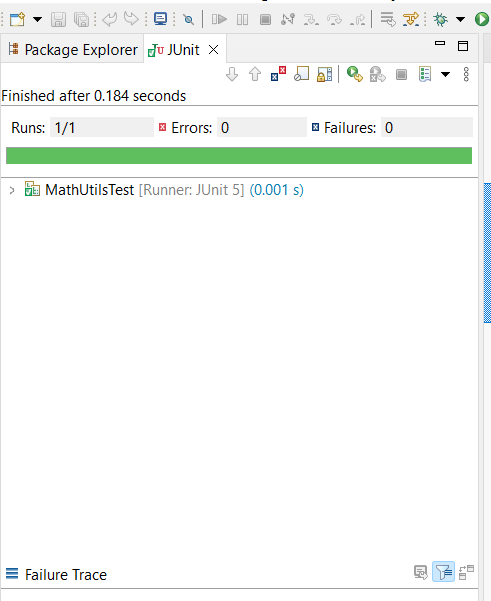
MathUtils math = new MathUtils();

assertEquals(20, math.multiply(4, 5));

}

}

**Output:**

****

**Exercise 3: Assertions in JUnit**

* Creating a class AssertionsTest in JUnitTest package.
* Right-click inside the file → Choose Run As → JUnit Test.
* Eclipse runs the test and shows the JUnit panel with a green bar if all tests pass.

**AssertionsTest.java:**

package JunitTest;

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

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.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);

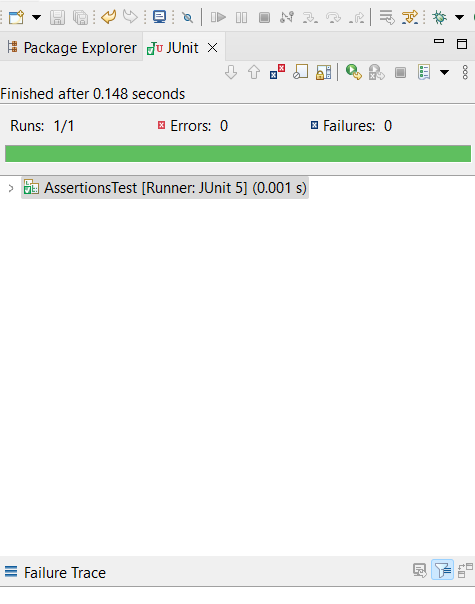
// Assert not null

*assertNotNull*(new Object());

}

}

**Output:**

****

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

* Create a new package AAA and add a simple class Calculator.java.
* Create a separate test package AAATest and add CalculatorTest.java.
* Right-click on CalculatorTest.java.
* Select Run As > JUnit Test.
* Check the JUnit result tab and console for output.

**Calculator.java:**

package AAA;

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 AAATest;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

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

import AAA.Calculator;

public class CalculatorTest {

private Calculator calculator;

// Setup method - runs before each test

@BeforeEach

public void setUp() {

calculator = new Calculator();

System.*out*.println("Setup complete.");

}

// Teardown method - runs after each test

@AfterEach

public void tearDown() {

calculator = null;

System.*out*.println("Teardown complete.");

}

@Test

public void testAddition() {

// Arrange

int a = 5;

int b = 3;

// Act

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

// Assert

*assertEquals*(8, result);

}

@Test

public void testSubtraction() {

// Arrange

int a = 10;

int b = 4;

// Act

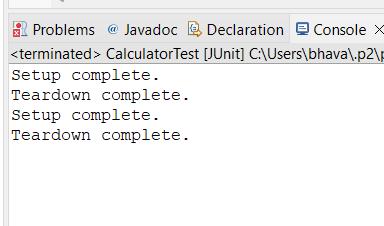
int result = calculator.subtract(a, b);

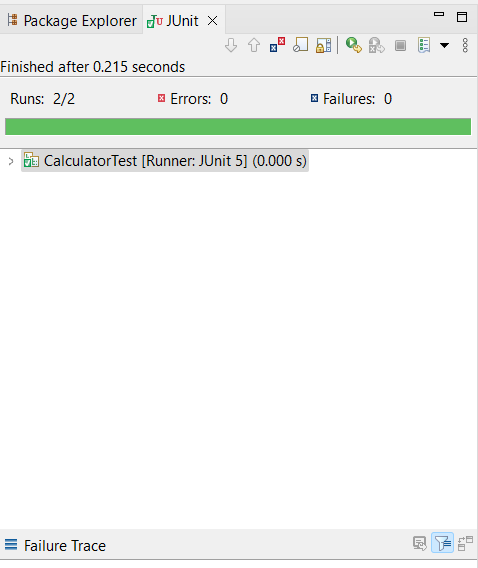
// Assert

*assertEquals*(6, result);

}

}

**Output:**

****

**Exercise 1: Mocking and Stubbing**

* Create Maven Project.
* In Eclipse click File > New > Maven Project.Check "Create a simple project (skip archetype)".
* Give Group ID as com.example and Artifact ID as MockitoExample and Click Finish.
* Add JUnit 5 and Mockito dependencies in pom.xml.
* Create an interface which need to be mocked and a class(MyService) in src/main/java.
* Write the test unit class(MyServiceTest)in which we mock the interface and stub the methods in src/test/java.
* Run the test by right clicking on test class>Run As>Junit test case. You should see a green bar for successful test result.

**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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>MockitoExample</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Enable JUnit 5 -->

<plugin>

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

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

<version>3.0.0-M9</version>

<configuration>

<useModulePath>false</useModulePath>

</configuration>

</plugin>

</plugins>

</build>

</project>

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

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

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

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

MyService service = new MyService(mockApi);

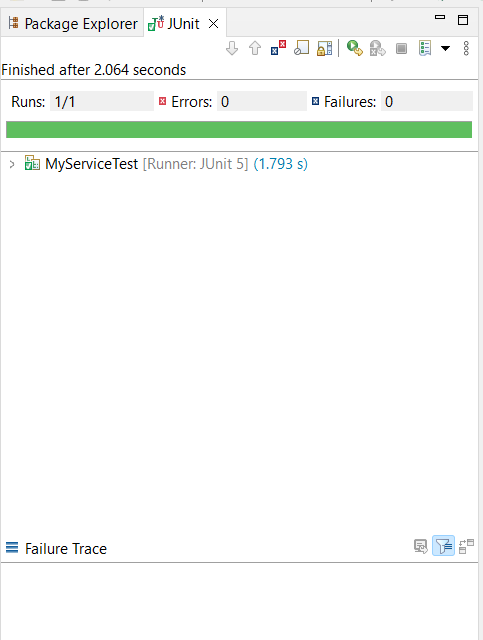
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Output:**

****

**Exercise 2: Verifying Interactions**

* Create Maven Project.
* In Eclipse click File > New > Maven Project.Check "Create a simple project (skip archetype)".
* Give Group ID as com.example and Artifact ID as MockitoVerifyExample and Click Finish.
* Add JUnit 5 and Mockito dependencies in pom.xml.
* Create an interface which need to be mocked and a class(MyService) in src/main/java.
* Write the test unit class(MyServiceTest)in which we mock the interface and stub the methods in src/test/java.
* Run the test by right clicking on test class>Run As>Junit test case. You should see a green bar for successful test result.

**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

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>MockitoExample</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

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

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Enable JUnit 5 -->

<plugin>

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

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

<version>3.0.0-M9</version>

<configuration>

<useModulePath>false</useModulePath>

</configuration>

</plugin>

</plugins>

</build>

</project>

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

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

MyService service = new MyService(mockApi);

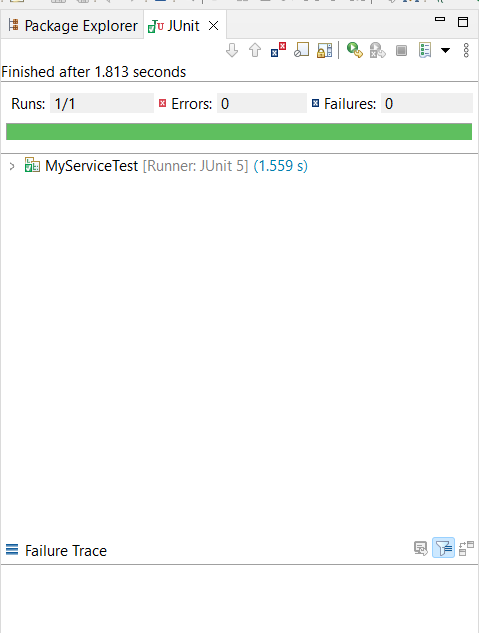
service.fetchData();

verify(mockApi).getData();

}

}

**Output:**

****