JUNIT TESTING , Mockito and SL4J Logging

Exercise 1: Setting Up Junit:

Code:

Square.java:

package com.example;

public class Square {

public double square(double a) {

return a \* a;

}

} SquareTest.java:

package com.example;

import static org.junit.Assert.\*;

import org.junit.Test;

public class SquareTest {

Square s = new Square();

*@Test*

public void test1() {

double res = s.square(3);

*assertEquals*(9.0, res, 0.01);

}

*@Test*

public void test2() {

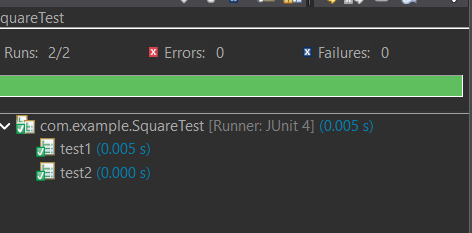
double res = s.square(8);

*assertEquals*(64.0, res, 0.001);

}

}

Output:



Exercise 3: Assertions in Junit:

Code:

package com.example;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionTest {

*@Test*

public void testAssertions() {

// Assert equals

*assertEquals*(5, 2 + 3);

// Assert true

*assertTrue*(5 > 3);

// Assert false

*assertFalse*(5 < 3);

// Assert null

*assertNull*(null);

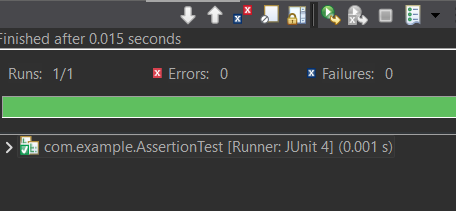
// Assert not null

*assertNotNull*(new Object());

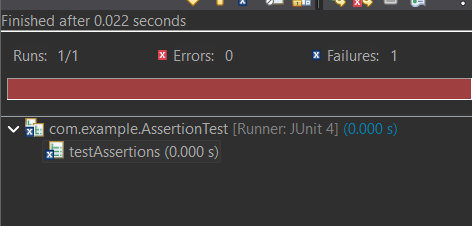
}

}

Output:



*When assertTrue*(5 < 3);



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

Code:

package com.example;

import static org.junit.Assert.\*;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class SquareTest {

private Square s;

*@Before*

public void setUp() {

System.***out***.println("Before");

s = new Square();

}

*@After //we can also use @afterall, @aftereeach,@afterclass*

public void tearDown() {

System.***out***.println("Teardown: Nullifying Calculator object");

s = null; // db.close();

}

*@Test*

public void square(){

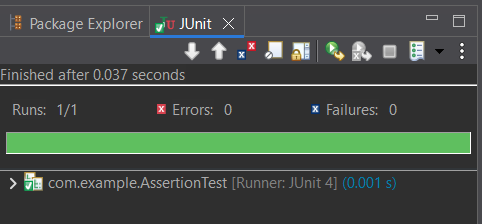
double res = s.square(5.0);

*assertEquals*(25.0, res,0.01);

}

}

Output:



Exercise 1: Mocking and Stubbing:

Code: External.java

package com.example;

public interface External {

String getValue();

}

Service.java:

package com.example;

public class Service {

private External ext;

public Service(External ext) {

this.ext =ext;

}

public String fetchData() {

return ext.getValue();

}

}

ServiceTest.java:

package com.example;

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

import static org.mockito.Mockito;

import org.junit.jupiter.api.Test;

public class ServiceTest {

*@Test*

public void testFetchData\_WithMockApi() {

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

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

Service serv = new Service(mockApi);

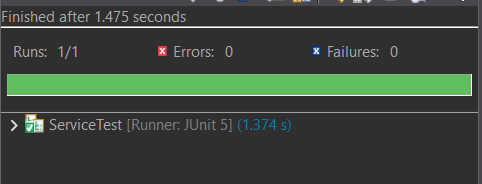
String res = serv.fetchData();

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

}

}

Output:



Exercise 2: Verifying Interactions:

Code:

Service.java, External.java were used from above code formatons and the code that is given as solution is executed. The only need is to change the dependencies in maven.

package com.example;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class ServiceTest {

*@Test*

public void testVerifyInteraction() {

External mockApi = Mockito.*mock*(External.class);

Service service = new Service(mockApi);

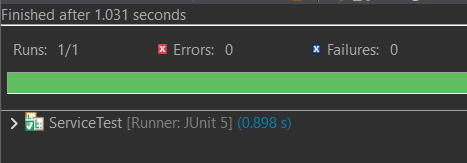
service.fetchData();

*verify*(mockApi).getValue();

}

}

Output:



Exercise 1: Logging Error Messages and Warning Levels:

Code:

// as maven code and Logger example code was given already to run this application we further needed is an xml document

Log.xml:

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{yyyy-dd-mm} -%msg%n</pattern>

</encoder>

</appender>

<root level="debug">

<appender-ref ref="STDOUT" />

</root>

</configuration>

Output:

