**Exercise 1: Setting Up JUnit**

Scenario:

You need to set up JUnit in your Java project to start writing unit tests.

Steps:

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your

pom.xml:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

3. Create a new test class in your project.

**Code:  
package** com.teja.demo.Junit;

**import** org.junit.jupiter.api.Test;

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

**public** **class** CalculatorTest {

// Example method to test

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

**return** a + b;

}

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

**return** a \* b;

}

@Test

**void** testAddition() {

*assertEquals*(5, add(2, 3), "2 + 3 should be 5");

}

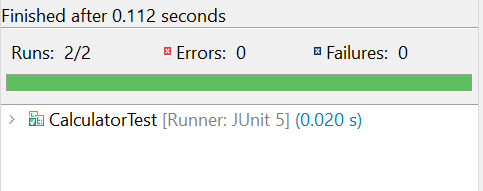
@Test

**void** testMultiplication() {

*assertEquals*(6, multiply(2, 3), "2 \* 3 should be 6");

}

}

**Output:** **Exercise 3: Assertions in JUnit**

Scenario:

You need to use different assertions in JUnit to validate your test results.

Steps: 1. Write tests using various JUnit assertions.

Solution Code:

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());

}

}

Code:  
**package** com.teja.demo.Junit;

**import** org.junit.jupiter.api.Test;

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

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

@Test

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

// Assert equals

*assertEquals*(5, 2 + 3, "2 + 3 should equal 5");

// Assert true

*assertTrue*(5 > 3, "5 is greater than 3");

// Assert false

*assertFalse*(5 < 3, "5 is not less than 3");

// Assert null

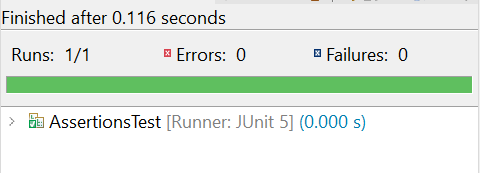
*assertNull*(**null**, "Should be null");

// Assert not null

*assertNotNull*(**new** Object(), "Should not be null");

}

}

OutPut:  
  


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

**Teardown Methods in JUnit**

Scenario:

You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup

and teardown methods.

Steps:

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods

**Code:**

**package** com.teja.demo.Junit;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.AfterEach;

**import** org.junit.jupiter.api.Test;

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

**public** **class** CalculatorTest {

**private** Calculator calculator;

@BeforeEach

**void** setUp() {

calculator = **new** Calculator();

System.***out***.println("Setup: New calculator created");

}

@AfterEach

**void** tearDown() {

calculator = **null**;

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

}

@Test

**void** testAddition() {

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

*assertEquals*(5, result, "2 + 3 should equal 5");

}

@Test

**void** testMultiplication() {

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

*assertEquals*(20, result, "4 \* 5 should equal 20");

}

}

**package** com.teja.demo.Junit;

**public** **class** Calculator {

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

**return** a + b;

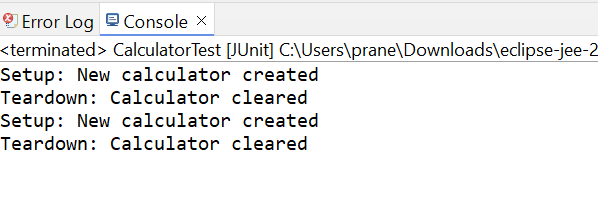
}

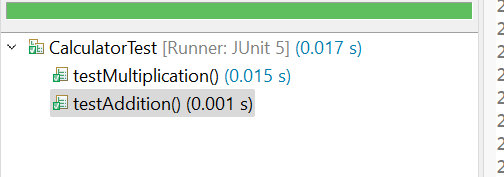
**public** **int** multiply(**int** a, **int** b) {

**return** a \* b;

}

}

**Output:**



**xercise 1: Mocking and Stubbing**

Scenario:

You need to test a service that depends on an external API. Use Mockito to mock the

external API and stub its methods.

Steps:

1. Create a mock object for the external API.

2. Stub the methods to return predefined values.

3. Write a test case that uses the mock object.

**Solution Code**

**package** com.teja.demo.Junit;

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

String getData();

}

**package** com.teja.demo.Junit;

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

**private** **final** ExternalApi api;

**public** MyService(ExternalApi api) {

**this**.api = api;

}

**public** String fetchData() {

**return** api.getData();

}

}

**package** com.teja.demo.Junit;

**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);

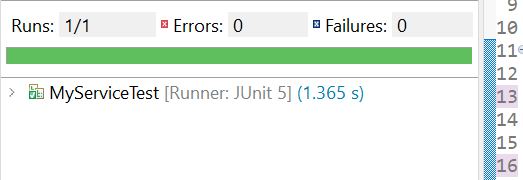
String result = service.fetchData();

*assertEquals*("Mock Data", result, "Service should return mocked data");

}

}

**Output:**



**Exercise 2: Verifying Interactions**

Scenario:

You need to ensure that a method is called with specific arguments.

Steps:

1. Create a mock object.

2. Call the method with specific arguments.

3. Verify the interaction.

**Solution Code:**

**package** com.teja.demo.Junit;

**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);

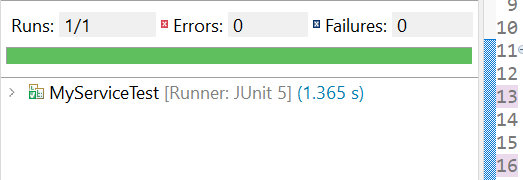
String result = service.fetchData();

*assertEquals*("Mock Data", result, "Service should return mocked data");

}

}

**Output:**



**Exercise 1: Logging Error Messages and Warning Levels**

Task: Write a Java application that demonstrates logging error messages and warning levels

using SLF4

**Code:  
package** com.teja.demo.Junit;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**public** **class** SJ4 {

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(SJ4.**class**);

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

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

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

}

}

**Output:**