**SUPERSET ID:** 6376594

**1.JUnit\_Basic Testing Exercises(Hands-On)**

**Exercise 1: Setting Up JUnit**

Project Name: MyJUnitProject

Calculator.java Code:

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

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

**return** a + b;

}

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

**return** a \* b;

}

}

CalculatorTest.java Code:

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

**import** org.junit.Before;

**import** org.junit.After;

**import** org.junit.Test;

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

**private** Calculator calculator;

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

calculator = **new** Calculator();

}

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

calculator = **null**;

}

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

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

*assertEquals*(5, result);

}

**public** **void** testMultiply() {

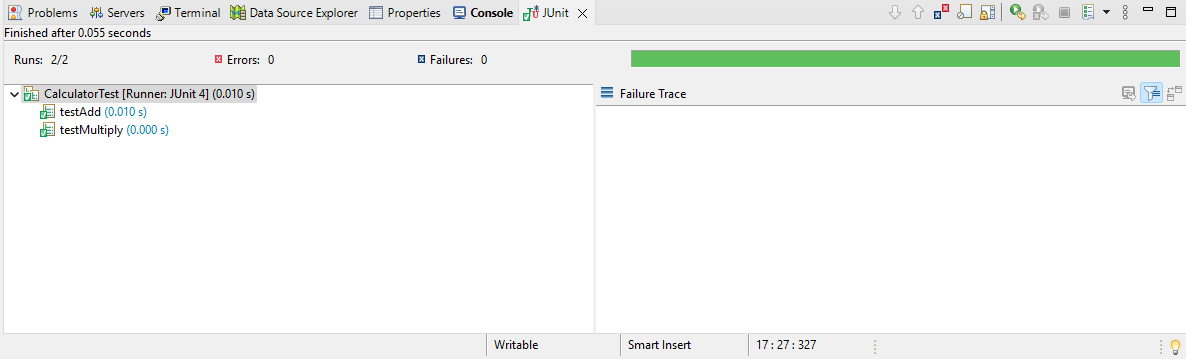
**int** result = calculator.multiply(2, 3);

*assertEquals*(6, result);

}

}

Output:



**Exercise 2: Writing Basic JUnit Tests**

Project Name: MyJUnitProject

AdvancedCalculator.java Code:

**public** **class** AdvancedCalculator {

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

**return** a + b;

}

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

**return** a - b;

}

}

AdvancedCalculatorTest.java Code:

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

**import** org.junit.Before;

**import** org.junit.After;

**import** org.junit.Test;

**public** **class** AdvancedCalculatorTest {

**private** AdvancedCalculator calculator;

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

calculator = **new** AdvancedCalculator();

}

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

calculator = **null**;

}

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

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

*assertEquals*(5, result);

}

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

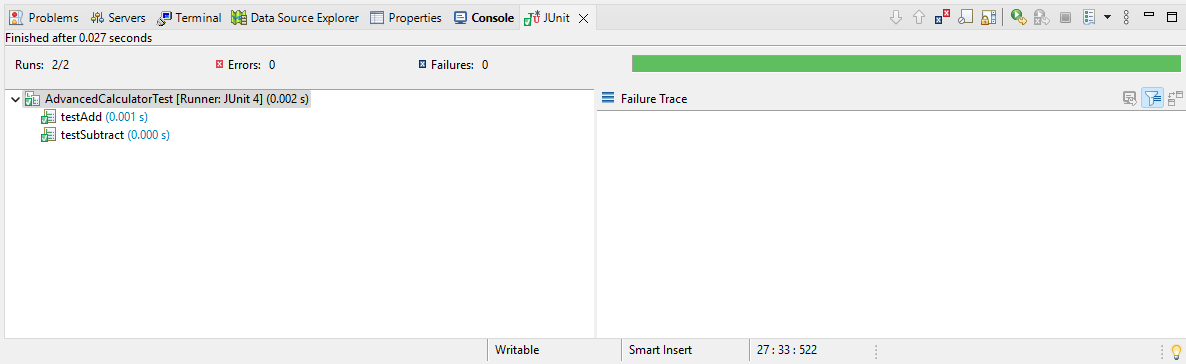
**int** result = calculator.subtract(5, 3);

*assertEquals*(2, result);

}

}

Output:



**Exercise 3: Assertions in JUnit**

Project Name: MyJUnitProject

AssertionsTest.java Code:

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

**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

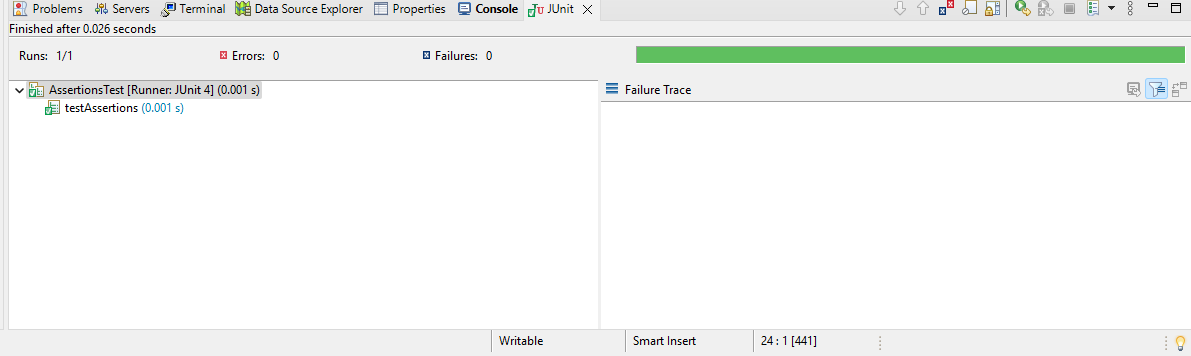
*assertNull*(**null**);

// Assert not null

*assertNotNull*(**new** Object());

}

}  
  
Output:



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

Project Name: MyJUnitProject

CalculatorAAATest.java Code:

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

**import** org.junit.Before;

**import** org.junit.After;

**import** org.junit.Test;

**public** **class** CalculatorAAATest {

**private** Calculator calculator;

// Will run BEFORE every test

@Before

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

// Arrange for every test

calculator = **new** Calculator();

}

// Will run AFTER every test

@After

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

// Cleanup after every test

calculator = **null**;

}

// TEST METHOD using the AAA Pattern

@Test

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

// Arrange

**int** a = 2;

**int** b = 3;

// Act

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

// Assert

*assertEquals*(5, result);

}

@Test

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

// Arrange

**int** a = 5;

**int** b = 3;

// Act

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

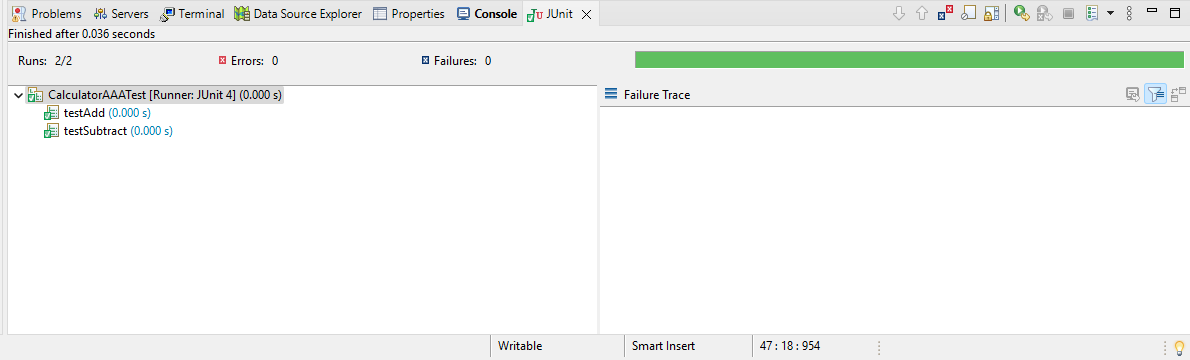
// Assert

*assertEquals*(2, result);

}

}

Output:



**2.Advanced JUnit Testing Exercises(Hands-On)**

**Exercise 1: Parameterized Tests**

Project Name: EvenTestProject

**EvenChecker.java** Code:

**public** **class** EvenChecker {

// Method to check if a number is even

**public** **boolean** isEven(**int** number) {

**return** number % 2 == 0;

}

}

EvenCheckerTest.java Code:

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

**import** org.junit.jupiter.params.ParameterizedTest;

**import** org.junit.jupiter.params.provider.ValueSource;

**public** **class** EvenCheckerTest {

**private** EvenChecker evenChecker = **new** EvenChecker();

@ParameterizedTest

@ValueSource(ints = {2, 4, 8, 10})

**void** testEvenNumbers(**int** number) {

// Arrange: We have the input number

// Act: Call the isEven method

**boolean** result = evenChecker.isEven(number);

// Assert: The result must be true

*assertTrue*(result);

}

@ParameterizedTest

@ValueSource(ints = {1, 3, 5, 9})

**void** testOddNumbers(**int** number) {

// Arrange

// Act

**boolean** result = evenChecker.isEven(number);

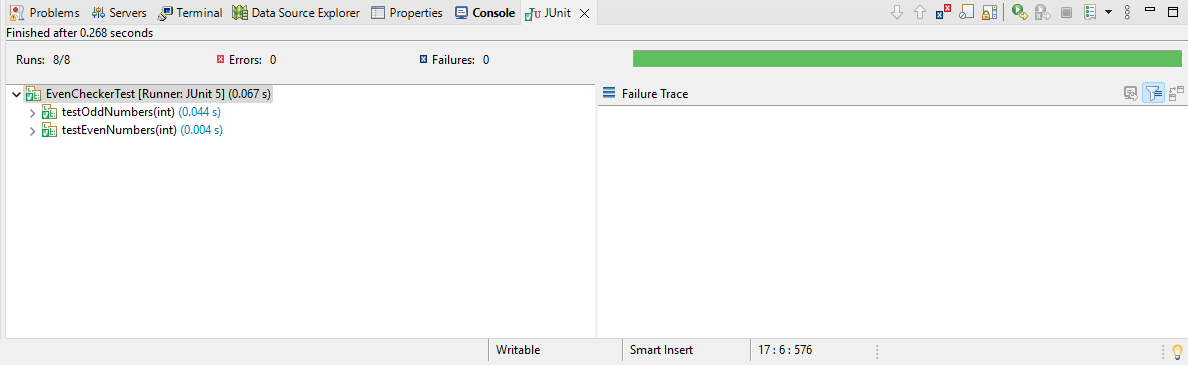
// Assert: The result must be false

*assertFalse*(result);

}

}

Output:



**Exercise 2: Test Suites and Categories**

Project Name: EvenTestProject

**EvenChecker.java** Code:

**public** **class** EvenChecker {

// Method to check if a number is even

**public** **boolean** isEven(**int** number) {

**return** number % 2 == 0;

}

}

EvenCheckerTest.java Code:

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

**import** org.junit.jupiter.params.ParameterizedTest;

**import** org.junit.jupiter.params.provider.ValueSource;

**public** **class** EvenCheckerTest {

**private** EvenChecker evenChecker = **new** EvenChecker();

@ParameterizedTest

@ValueSource(ints = {2, 4, 8, 10})

**void** testEvenNumbers(**int** number) {

// Arrange: We have the input number

// Act: Call the isEven method

**boolean** result = evenChecker.isEven(number);

// Assert: The result must be true

*assertTrue*(result);

}

@ParameterizedTest

@ValueSource(ints = {1, 3, 5, 9})

**void** testOddNumbers(**int** number) {

// Arrange

// Act

**boolean** result = evenChecker.isEven(number);

// Assert: The result must be false

*assertFalse*(result);

}

}

DummyTest.java Code:

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

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

**public** **class** DummyTest {

@Test

**void** sampleTest() {

*assertEquals*(2, 1 + 1);

}

}

AllTest.java Code:

**import** org.junit.platform.suite.api.SelectClasses;

**import** org.junit.platform.suite.api.Suite;

@Suite

@SelectClasses({

EvenCheckerTest.**class**,

DummyTest.**class**

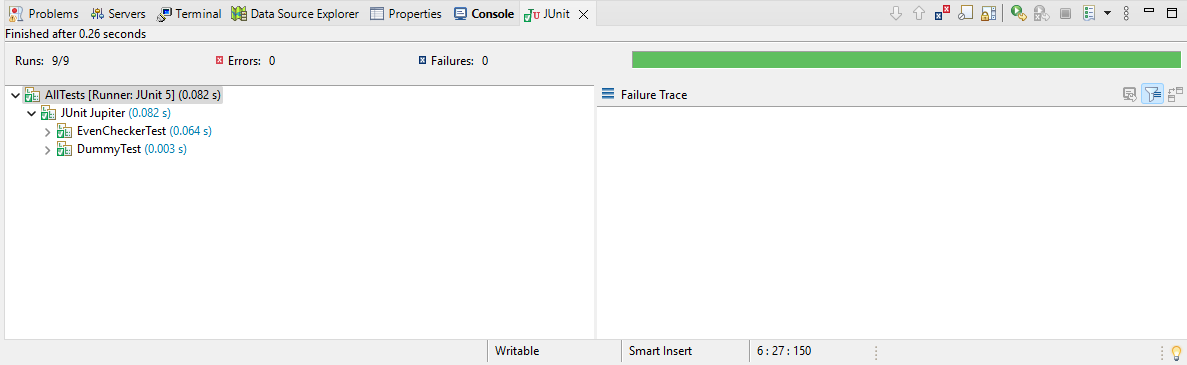
})

**public** **class** AllTests {

// This class is just a test suite

}

Output:



**Exercise 3: Test Execution Order**

Project Name: EvenTestProject

OrderedTests.java Code:

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

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

**import** org.junit.jupiter.api.MethodOrderer.OrderAnnotation;

@TestMethodOrder(OrderAnnotation.**class**) // Enables ordering

**public** **class** OrderedTests {

@Test

@Order(1) // Will run first

**void** testFirst() {

System.***out***.println("First Test");

*assertTrue*(2 + 2 == 4);

}

@Test

@Order(2) // Will run second

**void** testSecond() {

System.***out***.println("Second Test");

*assertEquals*(5, 2 + 3);

}

@Test

@Order(3) // Will run third

**void** testThird() {

System.***out***.println("Third Test");

*assertNotNull*(**new** Object());

}

}

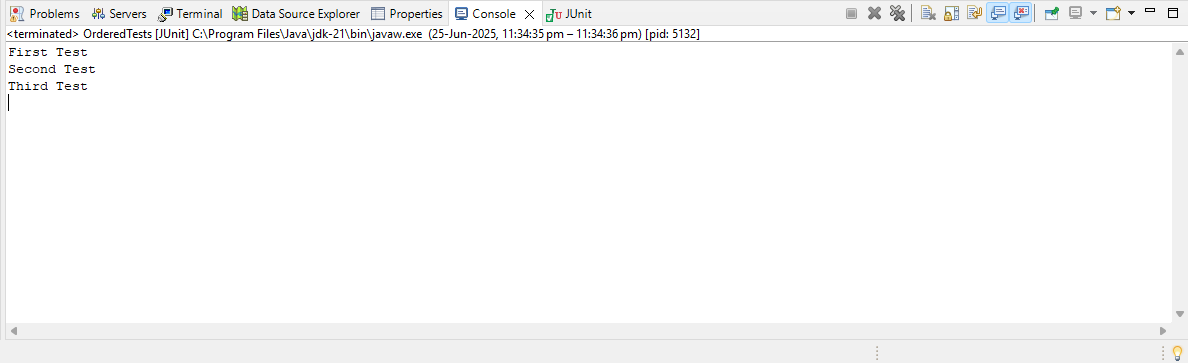
Expected Output:

First Test

Second Test

Third Test

Output:



**Exercise 4: Exception Testing**

Project Name: EvenTestProject

ExceptionThrower.java Code:

**public** **class** ExceptionThrower {

// This method intentionally throws an IllegalArgumentException

**public** **void** throwException() {

**throw** **new** IllegalArgumentException("This is an intentional error");

}

}

ExceptionThrowerTest.java Code:

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

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

**public** **class** ExceptionThrowerTest {

@Test

**void** testExceptionThrower() {

// Arrange

ExceptionThrower exceptionThrower = **new** ExceptionThrower();

// Act & Assert

*assertThrows*(IllegalArgumentException.**class**, () -> {

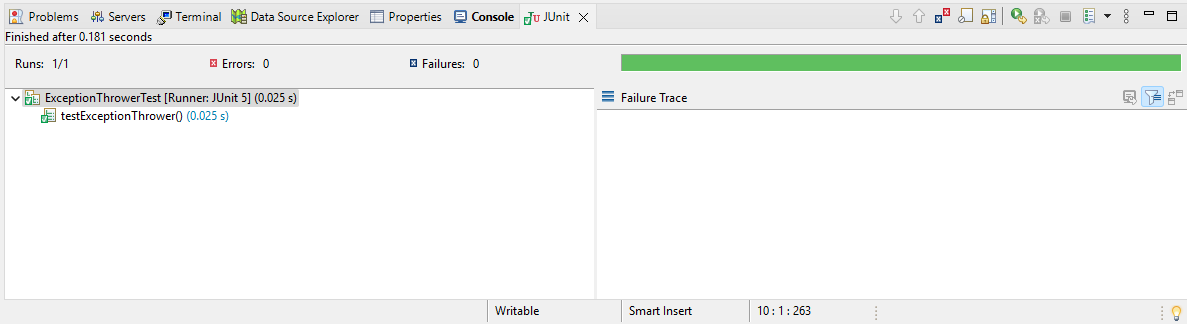
exceptionThrower.throwException();

});

}

}

Output:



**Exercise 5: Timeout and Performance Testing**

Project Name: EvenTestProject

PerformanceTest.java Code:

**public** **class** PerformanceTester {

// Simulates a long-running task

**public** **void** performTask() {

**try** {

// Simulate some work that takes roughly 500 ms

Thread.*sleep*(500);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

PerformanceTesterTest.java Code:

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

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

**import** java.time.Duration;

**public** **class** PerformanceTesterTest {

@Test

**void** testPerformTaskWithTimeout() {

PerformanceTester tester = **new** PerformanceTester();

// The test will fail if the method takes longer than 1 second

*assertTimeout*(Duration.*ofSeconds*(1), () -> {

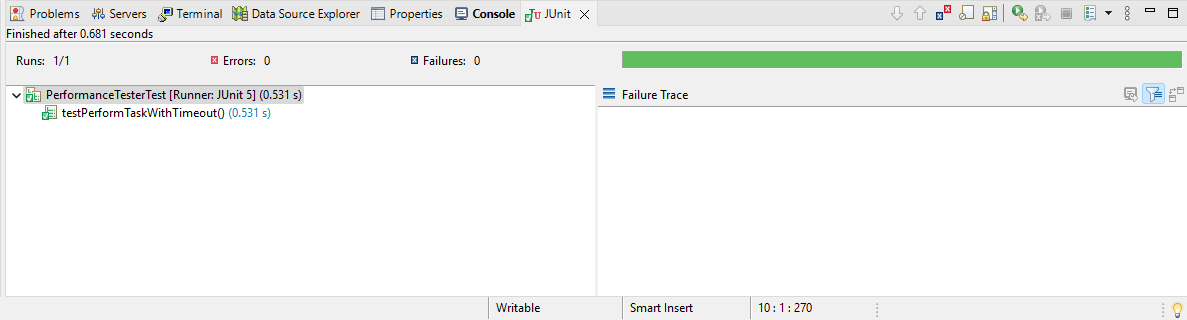
tester.performTask();

});

}

}

Output:



**3.Mockito Exercises(Hands-On)**

**Exercise 1: Mocking and Stubbing**

Project Name:Mockito TestProject

ExternalApi.java Code:

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

String getData();

}

MyService.java Code:

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

**private** ExternalApi api;

**public** MyService(ExternalApi api) {

**this**.api = api;

}

**public** String fetchData() {

**return** api.getData();

}

}

MyServiceTest.java Code:

**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);

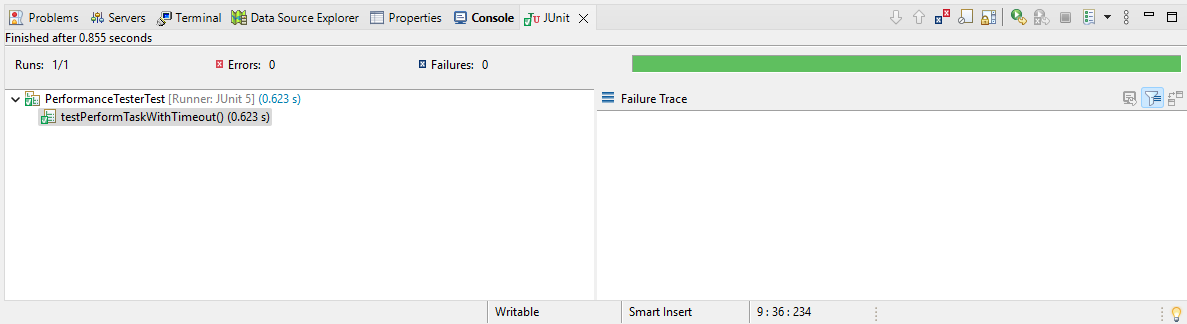
String result = service.fetchData();

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

}

}

Output:



**Exercise 2: Verifying Interactions**

Project Name:Mockito TestProject

ExternalApi.java Code:

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

String getData();

}

MyService.java Code:

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

**private** ExternalApi api;

**public** MyService(ExternalApi api) {

**this**.api = api;

}

**public** String fetchData() {

**return** api.getData();

}

}

MyServiceInteractionTest.java Code:

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

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

**import** org.mockito.Mockito;

**public** **class** MyServiceInteractionTest {

@Test

**public** **void** testVerifyInteraction() {

// 1. Create a mock of ExternalApi

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

// 2. Pass the mock into MyService

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

// 3. Call the method

service.fetchData();

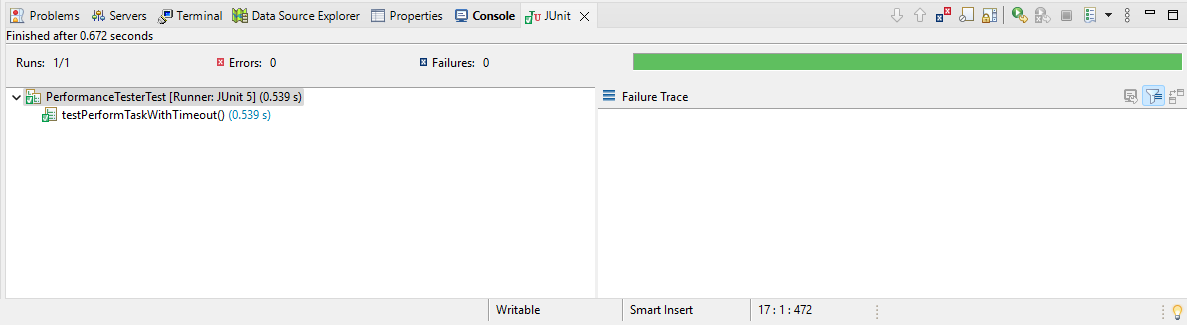
// 4. Verify the interaction

*verify*(mockApi).getData();

}

}

Output:



**6.Logging using SLF4J**

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

Project Name: LoggingExampleProject

LoggingExample.java Code:

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

// Create a logger instance

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

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

// Log messages at different levels

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

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

}

}

Excepted Output:

23:14:41.973 [main] ERROR LoggingExample - This is an error message

23:14:41.976 [main] WARN LoggingExample - This is a warning message

Output:

