**SCD Lab: Software Testing - Junit Tests**



**Session: 2021**

**Submitted by:**

**Wali Muhammad 2021-SE-39**

**Submitted to:**

**Ma’am Alina Munit**

Department of Computer Science,

**University of Engineering and Technology, New Campus Lahore**

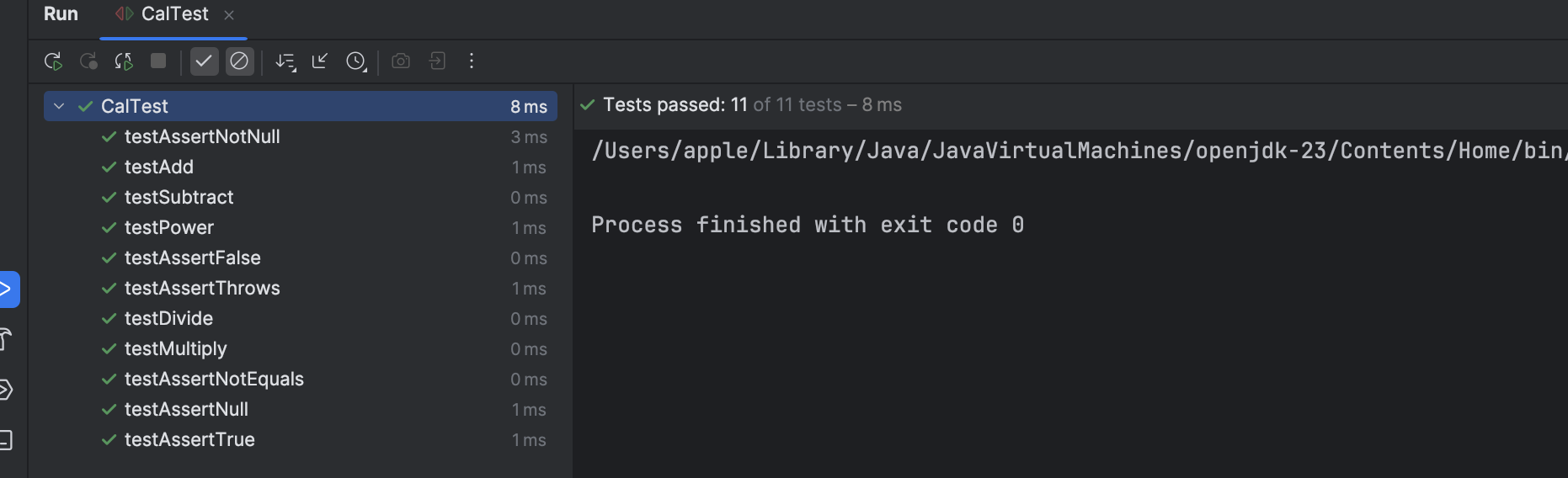
# Task 1

Write a test for each of the following assertion.

Answer

import static org.junit.Assert.\*;  
import org.junit.Test;  
  
public class CalTest {  
 private final Calculator calculator = new Calculator();  
  
 @Test  
 public void testAdd() {  
 *assertEquals*(5.0, calculator.add(2.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testSubtract() {  
 *assertEquals*(1.0, calculator.subtract(3.0, 2.0), 0.0001);  
 }  
  
 @Test  
 public void testMultiply() {  
 *assertEquals*(6.0, calculator.multiply(2.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testDivide() {  
 *assertEquals*(2.0, calculator.divide(6.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testPower() {  
 *assertEquals*(8.0, calculator.power(2.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testAssertNotEquals() {  
 *assertNotEquals*(7.0, calculator.add(2.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testAssertTrue() {  
 *assertTrue*(calculator.add(2.0, 3.0) == 5.0);  
 }  
  
 @Test  
 public void testAssertFalse() {  
 *assertFalse*(calculator.add(2.0, 3.0) == 6.0);  
 }  
  
 @Test  
 public void testAssertNull() {  
 // assertNull(calculator);  
 }  
  
 @Test  
 public void testAssertNotNull() {  
 *assertNotNull*(calculator);  
 }  
  
 @Test(expected = ArithmeticException.class)  
 public void testAssertThrows() {  
 calculator.divide(1.0, 0.0);  
 }  
}

Results



# Task 2

What is Mockito? When to use it?

Answer

Mockito is a Java-based framework used for unit testing of Java applications. This mocking framework helps in the development of testable applications. You use Java Reflection API internally to generate mock objects.

You use Java Reflection API internally to generate mock objects. Mockito is used to simplify test development by mocking external dependencies and using them in the code.[1]

Two important prerequisites are to be kept in mind before starting with learning Mockito are:

1. Java: Hands-on experience in Java programming.
2. JUnit: Basic understanding of the JUnit Framework.

When to Use Mockito?

Mockito is used in the following scenarios:

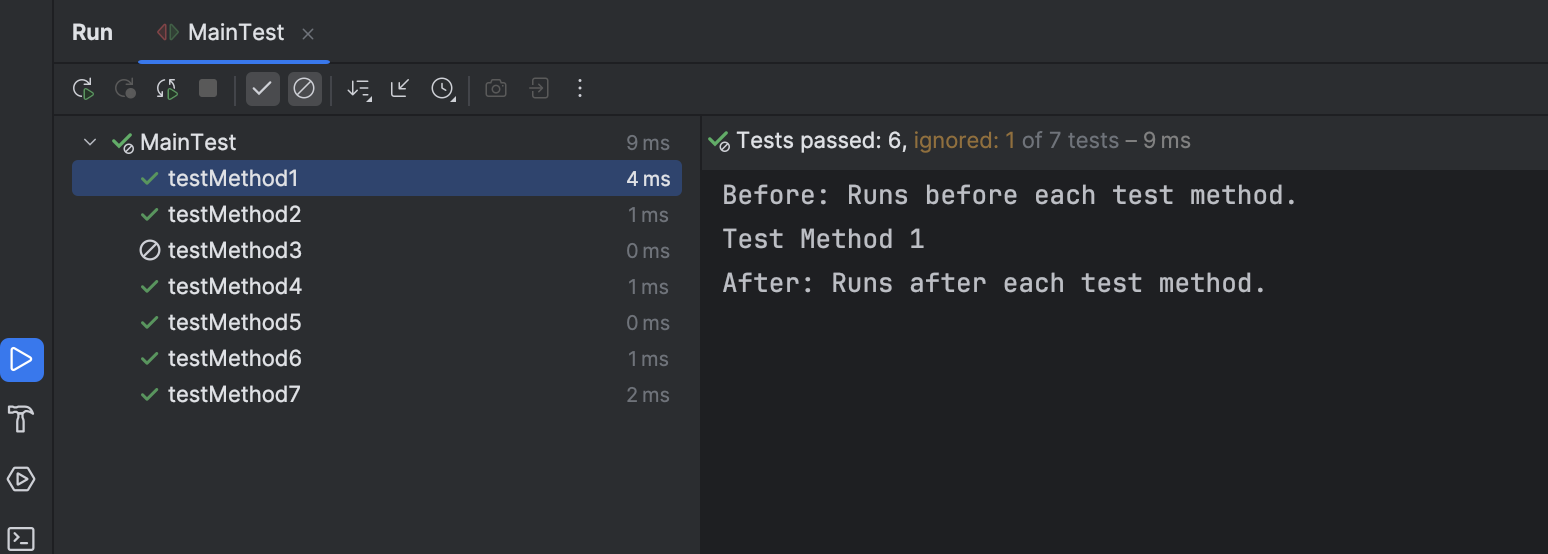
1. Isolating Dependencies: When you want to test a class in isolation, you can use Mockito to mock its dependencies. This ensures that the test focuses only on the behavior of the class under test.
2. Stubbing Behavior: You can use Mockito to define the behavior of mock objects. For example, you can specify what a method should return when it is called.
3. Verifying Interactions: Mockito allows you to verify that certain methods were called on mock objects with specific arguments.

# Task 3

Answer

import static org.junit.Assert.\*;  
import org.junit.Test;  
  
public class CalTest {  
 private final Calculator calculator = new Calculator();  
  
 @Test  
 public void testAdd() {  
 *assertEquals*(5.0, calculator.add(2.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testSubtract() {  
 *assertEquals*(1.0, calculator.subtract(3.0, 2.0), 0.0001);  
 }  
  
 @Test  
 public void testMultiply() {  
 *assertEquals*(6.0, calculator.multiply(2.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testDivide() {  
 *assertEquals*(2.0, calculator.divide(6.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testPower() {  
 *assertEquals*(8.0, calculator.power(2.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testAssertNotEquals() {  
 *assertNotEquals*(7.0, calculator.add(2.0, 3.0), 0.0001);  
 }  
  
 @Test  
 public void testAssertTrue() {  
 *assertTrue*(calculator.add(2.0, 3.0) == 5.0);  
 }  
  
 @Test  
 public void testAssertFalse() {  
 *assertFalse*(calculator.add(2.0, 3.0) == 6.0);  
 }  
  
 @Test  
 public void testAssertNull() {  
 // assertNull(calculator);  
 }  
  
 @Test  
 public void testAssertNotNull() {  
 *assertNotNull*(calculator);  
 }  
  
 @Test(expected = ArithmeticException.class)  
 public void testAssertThrows() {  
 calculator.divide(1.0, 0.0);  
 }  
}

Result



# References

[1] - https://www.simplilearn.com/tutorials/devops-tutorial/mockito-junit#:~:text=Mockito%20is%20a%20Java%2Dbased,internally%20to%20generate%20mock%20objects.