JavaAcademy

Calculator Program and Unit Testing using Mockito

Author

Pedro Jahir Hinojosa García

Instructor

Miguel Rugerio

Delivery date: September 6, 2024

Introduction

This document describes a Java program that implements a simple calculator with three functions: calculating the factorial of a number, finding the square of a number, and calculating the square root of a number. Additionally, the program uses Mockito for unit testing to verify that the 'Calculator' class interacts correctly with its 'MathService' dependency.

The program consists of the following classes:

- **Calculator**: The class that performs the main mathematical operations.
- **MathService**: An interface that defines the operations of the calculator.
- **MathServiceImp**: The implementation of the 'MathService' interface.
- **Main**: A driver class that executes the calculator functions.
- **MathTest**: A JUnit test class that uses Mockito to test the 'Calculator' class.

Calculator Class

The 'Calculator' class uses the 'MathService' interface to perform mathematical operations such as factorial, square, and square root.

```
package mockitodemo;

public class Calculator {

    private MathService mathService;

    public Calculator(MathService mathService) {
        this.mathService = mathService;
    }

    public void factorial(int a) {
        int result = mathService.factorial(a);
        System.out.println("Factorial of " + a + " = " + result);
    }

    public void square(int a) {
        int result = mathService.square(a);
        System.out.println(a + " squared = " + result);
    }

    public void squareRoot(int a) {
        double result = mathService.squareRoot(a);
        System.out.println("Square root of " + a + " = " + result);
```

```
    }
}
```

Explanation:

- The ‘Calculator‘ class depends on an instance of ‘MathService‘ for its operations. The ‘MathService‘ is injected into the constructor.
- The ‘factorial()‘, ‘square()‘, and ‘squareRoot()‘ methods call the corresponding methods in the ‘MathService‘ interface and print the results.

MathService Interface

The ‘MathService‘ interface defines the mathematical operations that the calculator will perform.

```
package mockitodemo;

public interface MathService {
    int factorial(int a);
    int square(int a);
    double squareRoot(int a);
}
```

Explanation:

- The interface declares three methods: ‘factorial()‘, ‘square()‘, and ‘squareRoot()‘.
- These methods are implemented in the ‘MathServiceImp‘ class.

MathServiceImp Class

The ‘MathServiceImp‘ class implements the ‘MathService‘ interface and provides the actual logic for the operations.

```
package mockitodemo;

public class MathServiceImp implements MathService {

    @Override
    public int factorial(int a) {
        if (a == 0 || a == 1) {
            return 1;
        }
        int fact = 1;
        for (int i = 2; i <= a; i++) {
            fact *= i;
```

```
        }
        return fact;
    }

    @Override
    public int square(int a) {
        return a * a;
    }

    @Override
    public double squareRoot(int a) {
        return Math.sqrt(a);
    }
}
```

Explanation:

- The ‘factorial()‘ method calculates the factorial of a number by multiplying all numbers from 1 to the input number.
- The ‘square()‘ method returns the square of the input number.
- The ‘squareRoot()‘ method calculates the square root of the input number using the ‘Math.sqrt()‘ method.

Main Class

The ‘Main‘ class runs the program and performs the three operations: factorial, square, and square root.

```
package mockitodemo;

public class Main {

    public static void main(String[] args) {
        MathService mathService = new MathServiceImp();
        Calculator calculator = new Calculator(mathService);

        calculator.factorial(5);
        calculator.square(4);
        calculator.squareRoot(16);
    }
}
```

Explanation:

- The ‘Main‘ class creates an instance of ‘MathServiceImp‘ and passes it to the ‘Calculator‘ constructor.

- The methods ‘factorial(5)‘, ‘square(4)‘, and ‘squareRoot(16)‘ are called, which print the results of these calculations.

MathTest Class

The ‘MathTest‘ class uses Mockito to mock the ‘MathService‘ and verify that the ‘Calculator‘ class interacts with the mock correctly.

```
package mockitodemo;

import static org.mockito.Mockito.*;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;
import org.junit.jupiter.api.extension.ExtendWith;

@ExtendWith(MockitoExtension.class)
public class MathTest {

    @Mock
    private MathService mathServiceMock; // Creates a mock

    @InjectMocks
    private Calculator calculator; // Creates an instance of Calculator with the

    @Test
    public void testFactorial() {
        when(mathServiceMock.factorial(5)).thenReturn(120);
        calculator.factorial(5);
        verify(mathServiceMock).factorial(5);
    }

    @Test
    public void testSquare() {
        when(mathServiceMock.square(4)).thenReturn(16);
        calculator.square(4);
        verify(mathServiceMock).square(4);
    }

    @Test
    public void testSquareRoot() {
        when(mathServiceMock.squareRoot(16)).thenReturn(4.0);
        calculator.squareRoot(16);
        verify(mathServiceMock).squareRoot(16);
    }
```

}

Explanation:

- The ‘@Mock‘ annotation creates a mock object of the ‘MathService‘ class, which simulates the real behavior.
- The ‘@InjectMocks‘ annotation creates an instance of ‘Calculator‘ and injects the mock ‘MathService‘.
- In the ‘testFactorial()‘ method, the mock is set to return ‘120‘ when the factorial of ‘5‘ is requested. It verifies that the ‘factorial()‘ method was called on the mock.
- Similarly, ‘testSquare()‘ and ‘testSquareRoot()‘ test the square and square root operations, verifying that the mock methods are called with the correct arguments.

Program Output

Here is the output generated by the ‘Main‘ class:

```
Factorial of 5 = 120
Square root of 16 = 4.0
4 squared = 16
```

Explanation of Output:

- **Factorial of 5 = 120**: The program calculates 5! as $5 \times 4 \times 3 \times 2 \times 1 = 120$.
- **Square root of 16 = 4.0**: The square root of 16 is calculated as $\sqrt{16} = 4.0$.
- **4 squared = 16**: The square of 4 is calculated as $4 \times 4 = 16$.

Conclusion

This program demonstrates basic mathematical operations such as calculating the factorial, square, and square root of numbers. The unit tests ensure that the ‘Calculator‘ class interacts with the mocked ‘MathService‘ as expected. Using Mockito simplifies testing by allowing us to mock dependencies, making it easier to focus on testing the class logic.