MOCKITO

JAVA ACADEMY - XIDERAL

SEPTEMBER 7, 2024

AUTHOR: REGINA RODRIGUEZ CAMPO GARRIDO

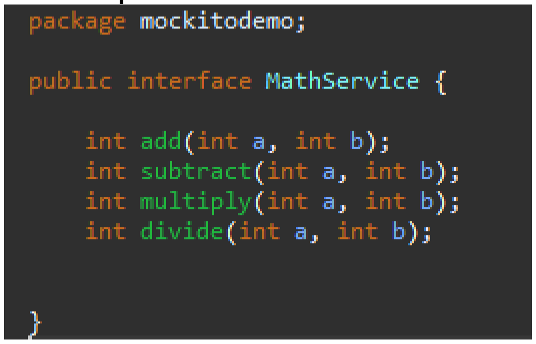
Mockito
September 7, 2024

Introduction

Mockito is a Java testing framework that allows you to create mock objects for unit testing. It helps isolate the code you're testing by simulating the behavior of dependencies, making it easier to verify interactions and test specific scenarios without relying on real implementations. I chose to use Mockito because it simplifies the testing process by providing flexible configuration of mock behavior, ensuring that tests remain clean and focused, and allowing me to verify the exact interactions with dependencies.

MathService Interface

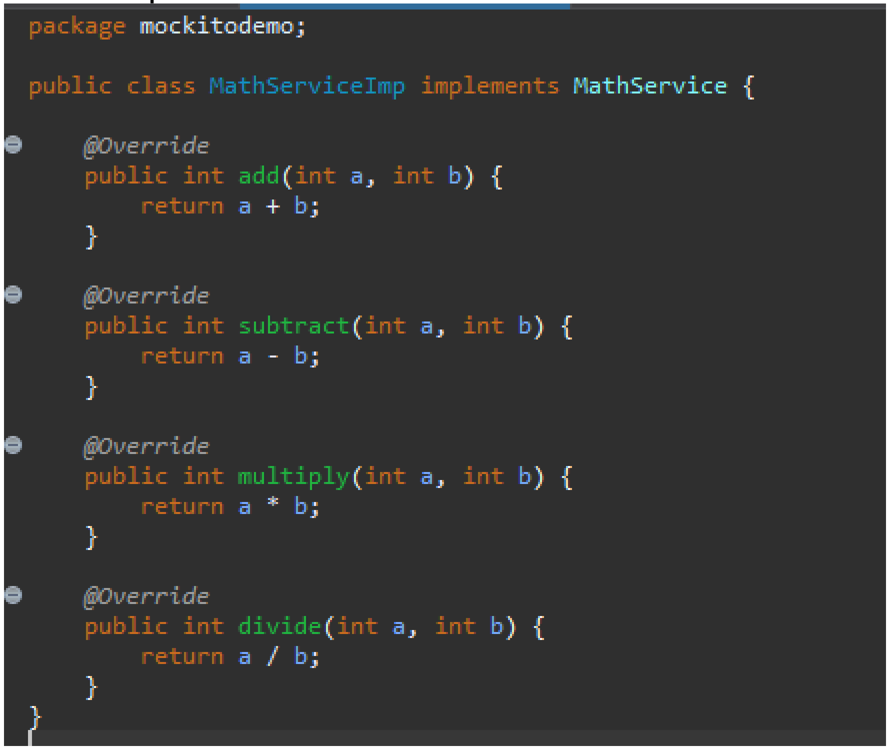
Defines a set of mathematical operations.



```java
package mockitodemo;

public interface MathService {

    int add(int a, int b);
    int subtract(int a, int b);
    int multiply(int a, int b);
    int divide(int a, int b);


}
```

MathServiceImp Implementation

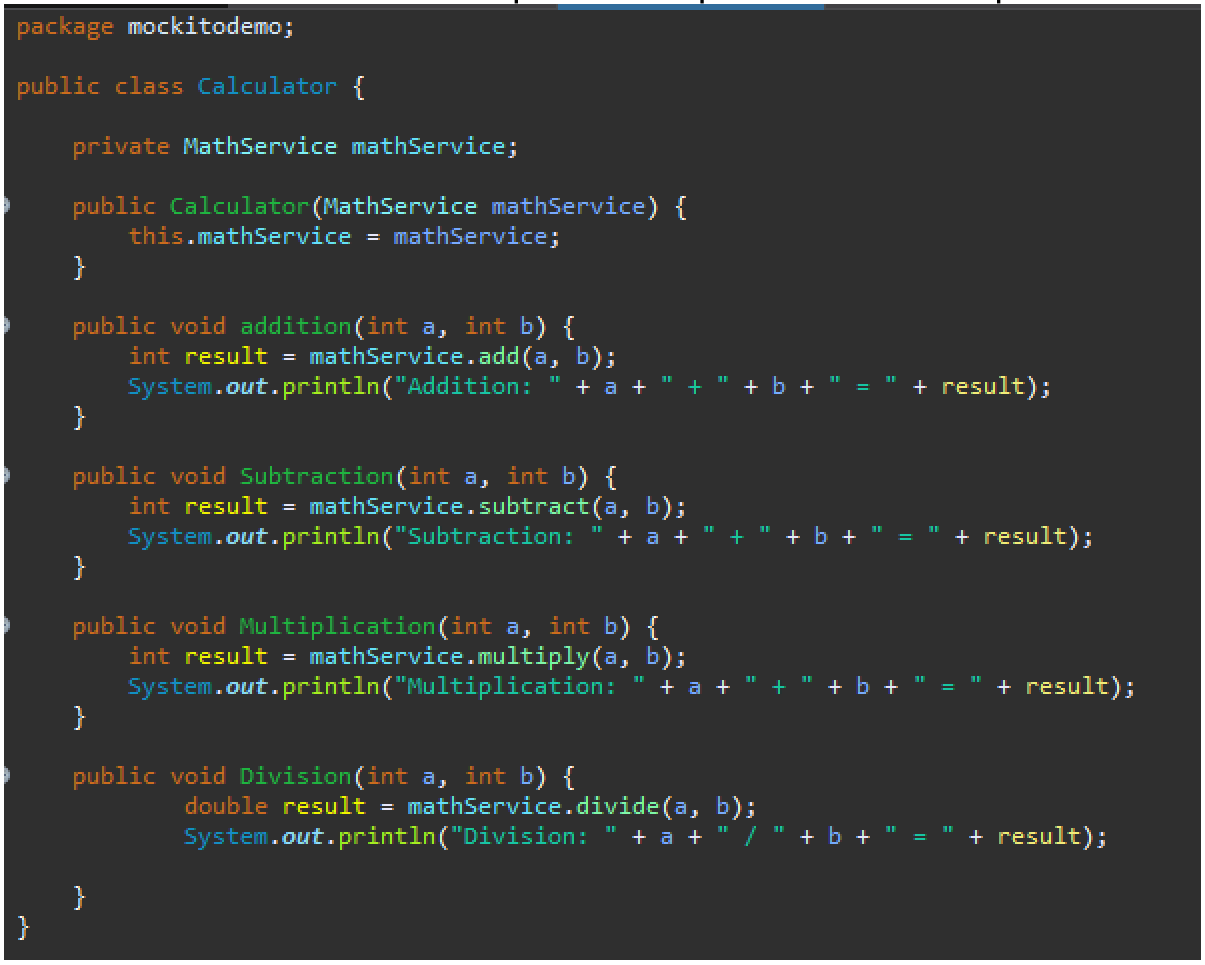
Provides concrete implementations for the MathService interface methods.



```java
package mockitodemo;

public class MathServiceImp implements MathService {

    @Override
    public int add(int a, int b) {
        return a + b;
    }

    @Override
    public int subtract(int a, int b) {
        return a - b;
    }

    @Override
    public int multiply(int a, int b) {
        return a * b;
    }

    @Override
    public int divide(int a, int b) {
        return a / b;
    }
}
```

Calculator Class

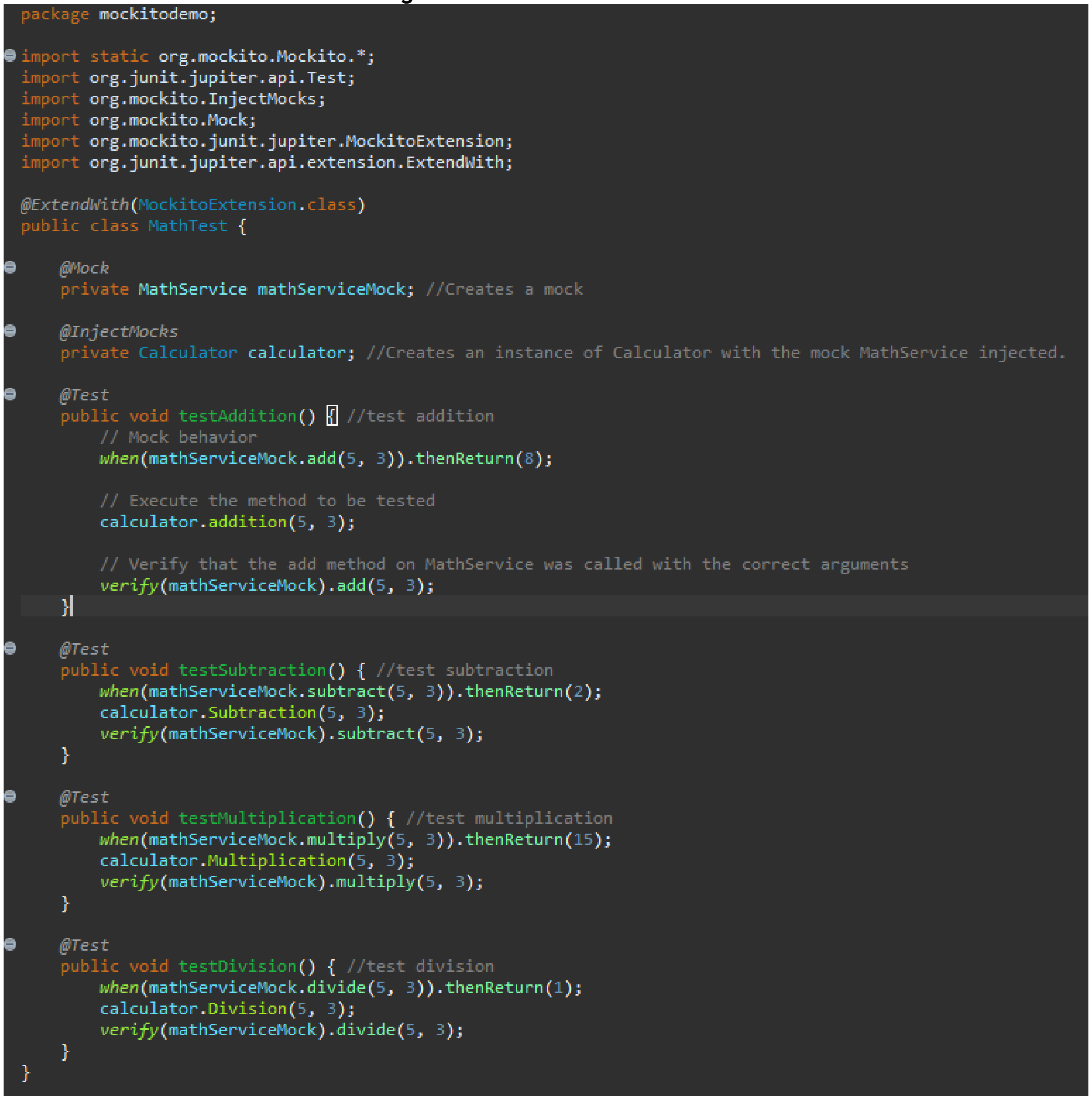
Uses an instance of MathService to perform and print mathematical operations.



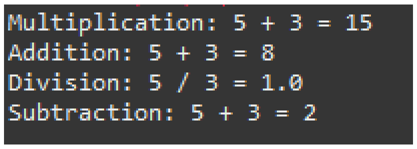
```
package mockitodemo;

public class Calculator {

    private MathService mathService;

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

    public void addition(int a, int b) {
        int result = mathService.add(a, b);
        System.out.println("Addition: " + a + " + " + b + " = " + result);
    }

    public void Subtraction(int a, int b) {
        int result = mathService.subtract(a, b);
        System.out.println("Subtraction: " + a + " + " + b + " = " + result);
    }

    public void Multiplication(int a, int b) {
        int result = mathService.multiply(a, b);
        System.out.println("Multiplication: " + a + " + " + b + " = " + result);
    }

    public void Division(int a, int b) {
            double result = mathService.divide(a, b);
            System.out.println("Division: " + a + " / " + b + " = " + result);

    }
}
```

Test

Tests the Calculator class using Mockito to mock the MathService interface



```java
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 mock MathService injected.

    @Test
    public void testAddition() { //test addition
        // Mock behavior
        when(mathServiceMock.add(5, 3)).thenReturn(8);

        // Execute the method to be tested
        calculator.addition(5, 3);

        // Verify that the add method on MathService was called with the correct arguments
        verify(mathServiceMock).add(5, 3);
    }

    @Test
    public void testSubtraction() { //test subtraction
        when(mathServiceMock.subtract(5, 3)).thenReturn(2);
        calculator.Subtraction(5, 3);
        verify(mathServiceMock).subtract(5, 3);
    }

    @Test
    public void testMultiplication() { //test multiplication
        when(mathServiceMock.multiply(5, 3)).thenReturn(15);
        calculator.Multiplication(5, 3);
        verify(mathServiceMock).multiply(5, 3);
    }

    @Test
    public void testDivision() { //test division
        when(mathServiceMock.divide(5, 3)).thenReturn(1);
        calculator.Division(5, 3);
        verify(mathServiceMock).divide(5, 3);
    }
}
```



```
Multiplication: 5 + 3 = 15
Addition: 5 + 3 = 8
Division: 5 / 3 = 1.0
Subtraction: 5 + 3 = 2
```