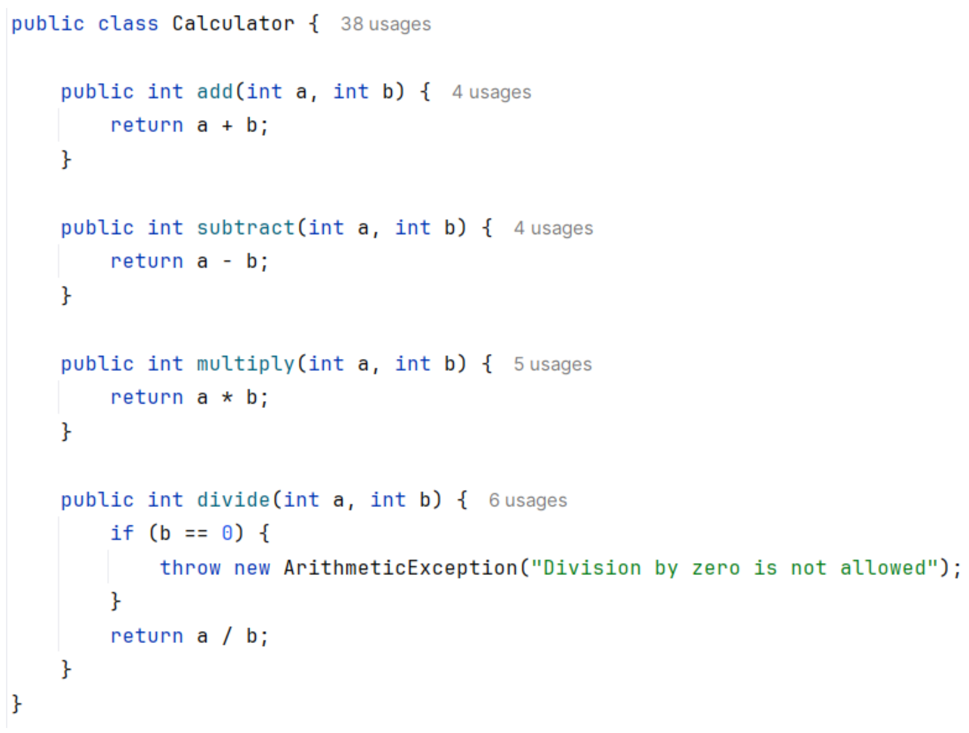
Software Testing TP Report

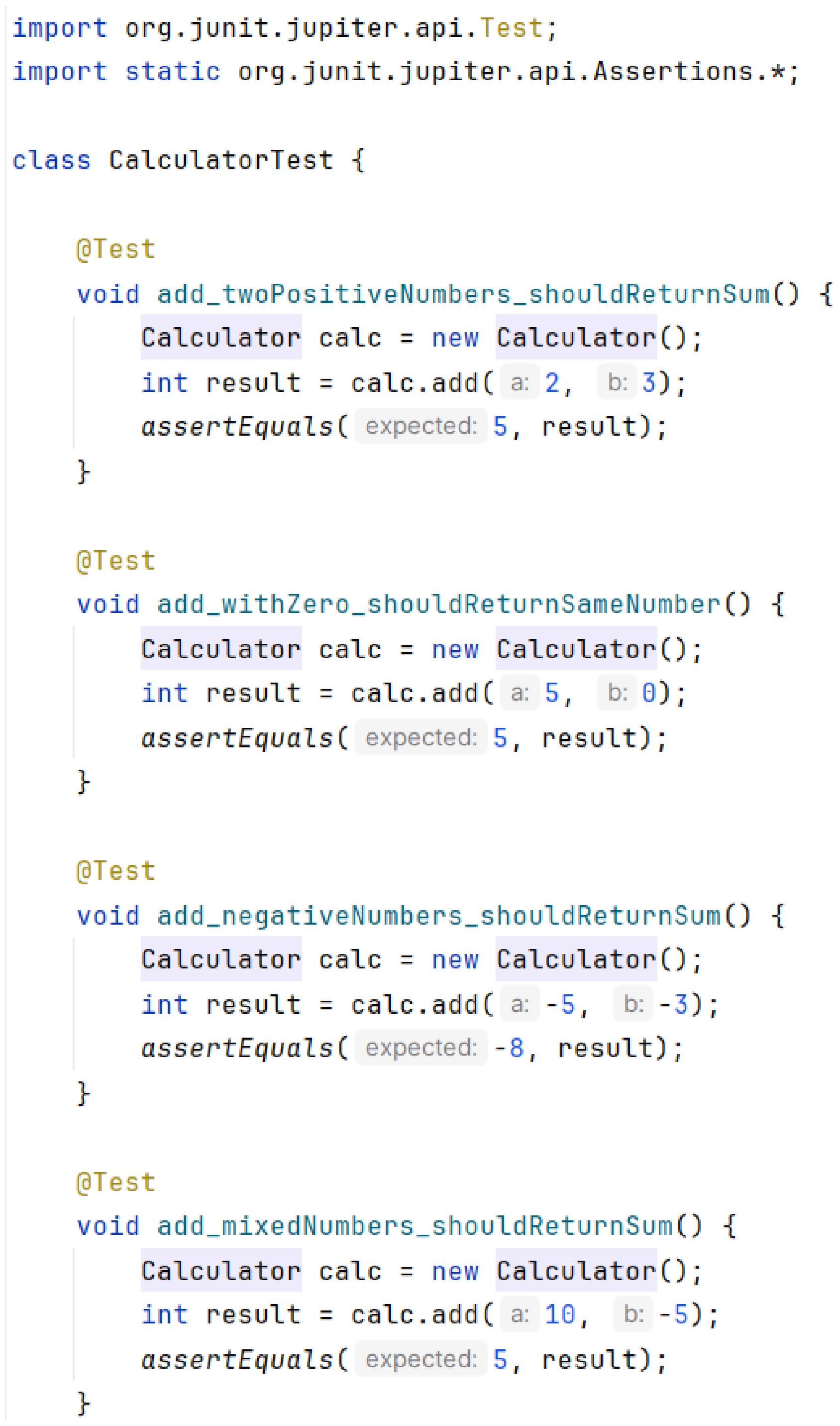
Ilyass Skiriba

Exercise 1: Calculator Unit Testing



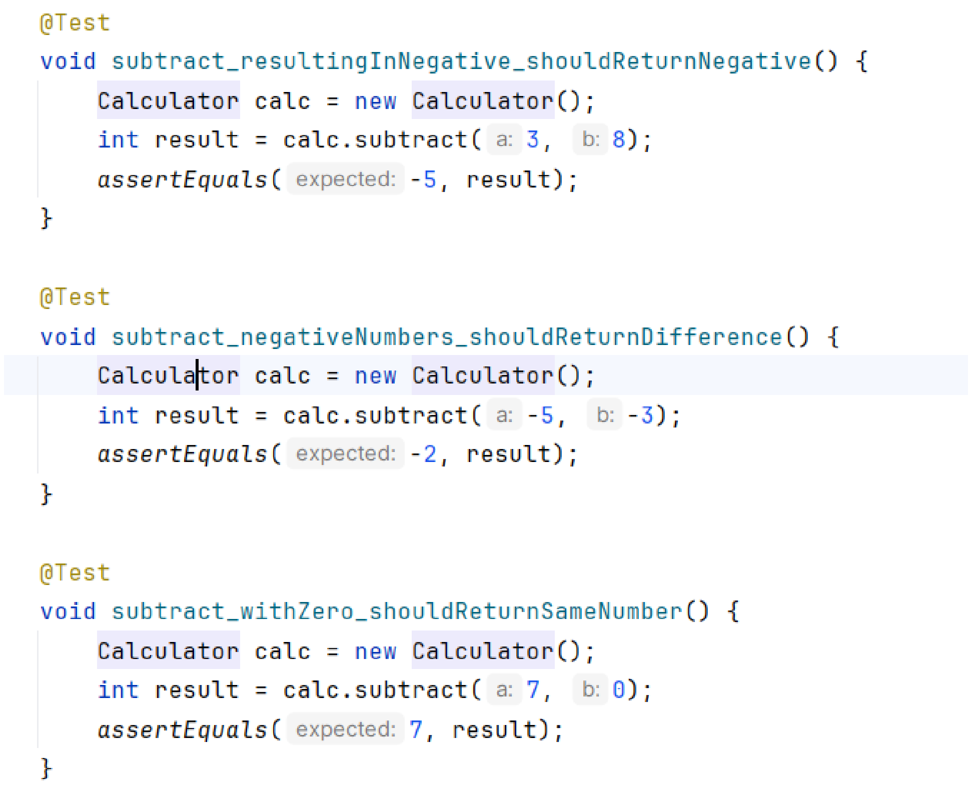
```java
public class Calculator {  38 usages

    public int add(int a, int b) {  4 usages
        return a + b;
    }

    public int subtract(int a, int b) {  4 usages
        return a - b;
    }

    public int multiply(int a, int b) {  5 usages
        return a * b;
    }

    public int divide(int a, int b) {  6 usages
        if (b == 0) {
            throw new ArithmeticException("Division by zero is not allowed");
        }
        return a / b;
    }
}
```

Figure 1: Calculator Class Implementation



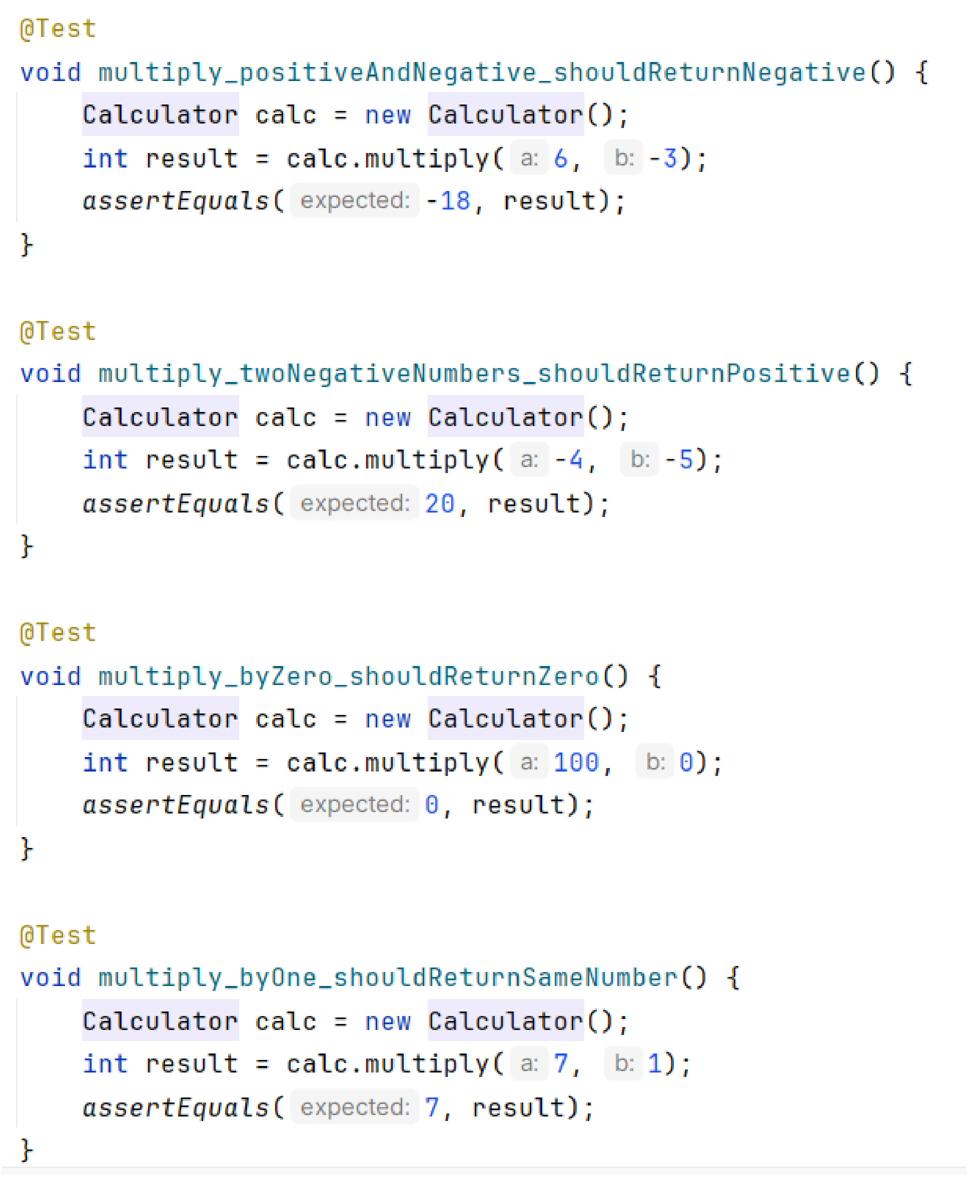
```java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class CalculatorTest {

    @Test
    void add_twoPositiveNumbers_shouldReturnSum() {
        Calculator calc = new Calculator();
        int result = calc.add( a: 2,  b: 3);
        assertEquals( expected: 5, result);
    }

    @Test
    void add_withZero_shouldReturnSameNumber() {
        Calculator calc = new Calculator();
        int result = calc.add( a: 5,  b: 0);
        assertEquals( expected: 5, result);
    }

    @Test
    void add_negativeNumbers_shouldReturnSum() {
        Calculator calc = new Calculator();
        int result = calc.add( a: -5,  b: -3);
        assertEquals( expected: -8, result);
    }

    @Test
    void add_mixedNumbers_shouldReturnSum() {
        Calculator calc = new Calculator();
        int result = calc.add( a: 10,  b: -5);
        assertEquals( expected: 5, result);
    }
```

Figure 2: Addition Test Cases



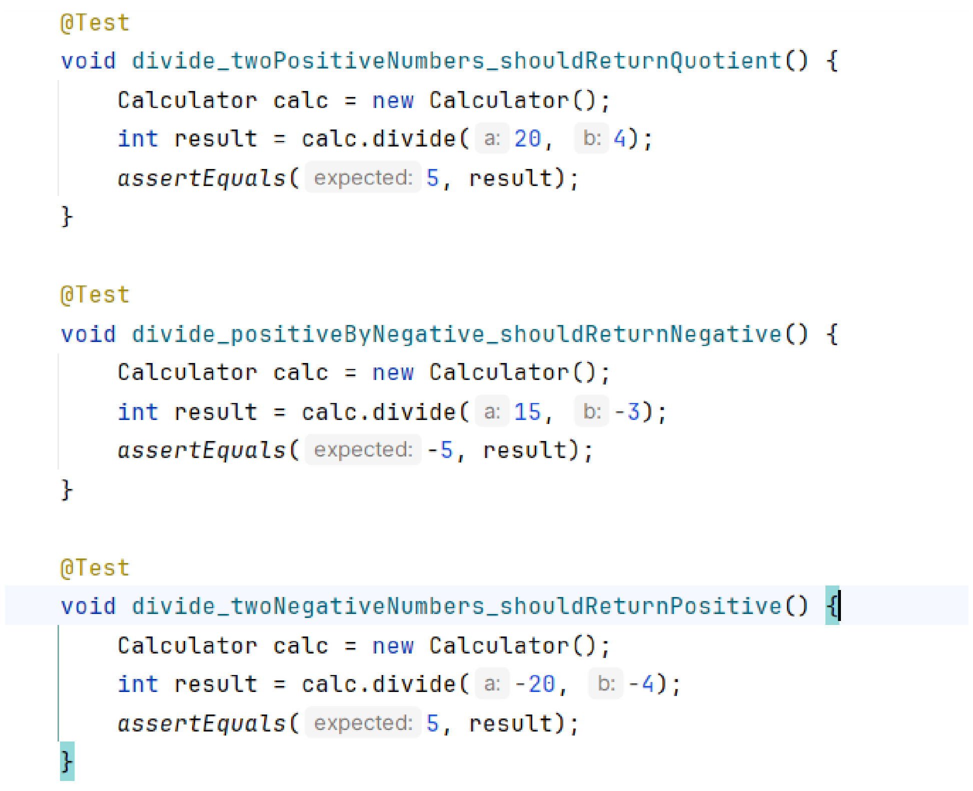
```java
@Test
void subtract_resultingInNegative_shouldReturnNegative() {
    Calculator calc = new Calculator();
    int result = calc.subtract( a: 3,  b: 8);
    assertEquals( expected: -5, result);
}

@Test
void subtract_negativeNumbers_shouldReturnDifference() {
    Calculator calc = new Calculator();
    int result = calc.subtract( a: -5,  b: -3);
    assertEquals( expected: -2, result);
}

@Test
void subtract_withZero_shouldReturnSameNumber() {
    Calculator calc = new Calculator();
    int result = calc.subtract( a: 7,  b: 0);
    assertEquals( expected: 7, result);
}
```

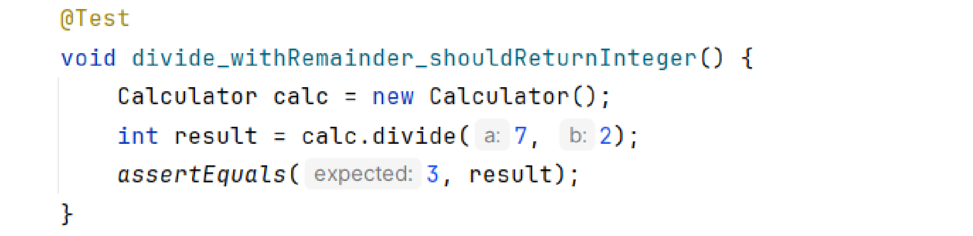
Figure 3: Subtraction Test Cases



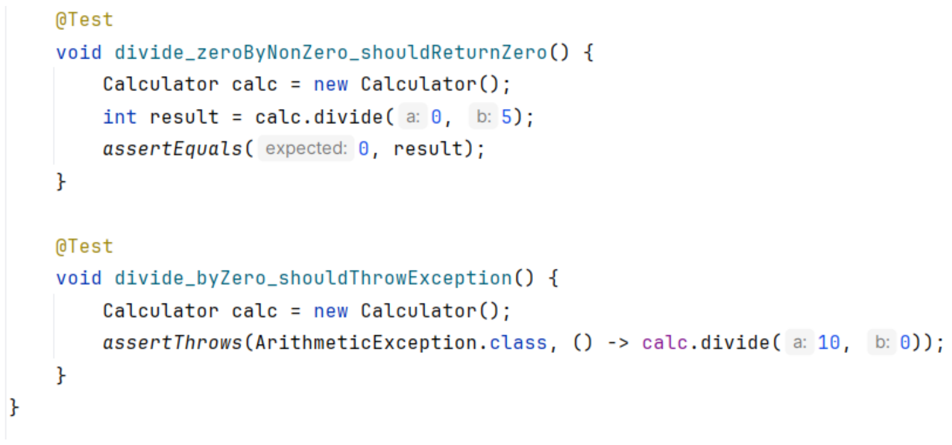
```java
@Test
void multiply_positiveAndNegative_shouldReturnNegative() {
    Calculator calc = new Calculator();
    int result = calc.multiply( a: 6,  b: -3);
    assertEquals( expected: -18, result);
}

@Test
void multiply_twoNegativeNumbers_shouldReturnPositive() {
    Calculator calc = new Calculator();
    int result = calc.multiply( a: -4,  b: -5);
    assertEquals( expected: 20, result);
}

@Test
void multiply_byZero_shouldReturnZero() {
    Calculator calc = new Calculator();
    int result = calc.multiply( a: 100,  b: 0);
    assertEquals( expected: 0, result);
}

@Test
void multiply_byOne_shouldReturnSameNumber() {
    Calculator calc = new Calculator();
    int result = calc.multiply( a: 7,  b: 1);
    assertEquals( expected: 7, result);
}
```

Figure 4: Multiplication Test Cases





```
@Test
void divide_twoPositiveNumbers_shouldReturnQuotient() {
    Calculator calc = new Calculator();
    int result = calc.divide( a: 20,  b: 4);
    assertEquals( expected: 5, result);
}

@Test
void divide_positiveByNegative_shouldReturnNegative() {
    Calculator calc = new Calculator();
    int result = calc.divide( a: 15,  b: -3);
    assertEquals( expected: -5, result);
}

@Test
void divide_twoNegativeNumbers_shouldReturnPositive() {
    Calculator calc = new Calculator();
    int result = calc.divide( a: -20,  b: -4);
    assertEquals( expected: 5, result);
}

@Test
void divide_withRemainder_shouldReturnInteger() {
    Calculator calc = new Calculator();
    int result = calc.divide( a: 7,  b: 2);
    assertEquals( expected: 3, result);
}
```



```
    @Test
    void divide_zeroByNonZero_shouldReturnZero() {
        Calculator calc = new Calculator();
        int result = calc.divide( a: 0,  b: 5);
        assertEquals( expected: 0, result);
    }

    @Test
    void divide_byZero_shouldThrowException() {
        Calculator calc = new Calculator();
        assertThrows(ArithmeticException.class, () -> calc.divide( a: 10,  b: 0));
    }
}
```

Figure 5: Division Test Cases