

## Fichero Calculadora comentado:

```
1 public class Calculadora {
2
3     private float lastResult;
4     private String lastOp;
5
6     /**
7      * Obtiene el último resultado obtenido.
8      *
9      * @return El último resultado obtenido.
10     */
11     public float getLastResult() {
12         return this.lastResult;
13     }
14
15     public String getLastOp() {
16         return this.lastOp;
17     }
18
19     /**
20     * Realiza una suma entre dos números.
21     *
22     * @param op1 El primer número.
23     * @param op2 El segundo número.
24     * @return El resultado de la suma.
25     */
26     public float suma(float op1, float op2) {
27         float result = op1 + op2;
28         this.lastResult = result;
29         this.lastOp = "Suma";
30         return result;
31     }
32
33     /**
34     * Realiza una resta entre dos números.
35     *
36     * @param op1 El primer número.
37     * @param op2 El segundo número.
38     * @return El resultado de la resta.
39     */
40 }
```

```
public float resta(float op1, float op2) {
    float result = op1 - op2;
    this.lastResult = result;
    this.lastOp = "Resta";
    return result;
}

/**
 * Realiza una multiplicacion entre dos números.
 *
 * @param op1 El primer número.
 * @param op2 El segundo número.
 * @return El resultado de la multiplicacion.
 */
public float multiplica(float op1, float op2) {
    float result = op1 * op2;
    this.lastResult = result;
    this.lastOp = "Multiplica";
    return result;
}

/**
 * Realiza una suma entre dos números.
 *
 * @param op1 El primer número.
 * @param op2 El segundo número.
 * @return El resultado de la division.
 */
public float divideix(float op1, float op2) {
    float result = op1 / op2;
    this.lastResult = result;
    this.lastOp = "Divideix";
    return result;
}
```

```

/**
 * Compara dos números para determinar si el primero es mayor que el
 * segundo.
 *
 * @param op1 El primer número.
 * @param op2 El segundo número.
 * @return true si el primer número es mayor que el segundo, false en caso
 * contrario.
 */
public boolean mayorQue(float op1, float op2) {
    if (op1 > op2) {
        return true;
    }
    return false;
}

/**
 * Restablece el estado de la calculadora, poniendo el último resultado a
 * cero y la última operación a "Ninguna".
 */
public void restablecer() {
    this.lastResult = 0;
    this.lastOp = "Ninguna";
}
}

```

## Fichero Test comentado

```

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import junit.Junit;

public class Exem_CalculadoraTest {
    private Junit junit;

    @BeforeEach
    public void setUp() {
        junit = new Junit();
    }

    @AfterEach
    public void tearDown() {
        junit.restablecer();
    }

    @Test
    public void testSuma() {
        float resultado = junit.suma(op1: 5, op2: 3);
        assertEquals(expected: 8, actual: resultado, delta: 0);
        assertEquals(expected: "Suma", actual: junit.getLastOp());
    }

    @Test
    public void testResta() {
        float resultado = junit.resta(op1: 10, op2: 4);
        assertEquals(expected: 6, actual: resultado, delta: 0);
        assertEquals(expected: "Resta", actual: junit.getLastOp());
    }
}

```

```

@Test
public void testMultiplica() {
    float resultado = junit.multiplica(op1: 2, op2: 5);
    assertEquals(expected:10, actual: resultado, delta:0);
    assertEquals(expected:"Multiplica", actual: junit.getLastOp());
}

@Test
public void testDivideix() {
    float resultado = junit.divideix(op1: 10, op2: 2);
    assertEquals(expected:5, actual: resultado, delta:0);
    assertEquals(expected:"Divideix", actual: junit.getLastOp());
}

@Test
public void testMayorQue() {
    assertTrue(condition: junit.mayorQue(op1: 5, op2: 3));
    assertFalse(condition: junit.mayorQue(op1: 3, op2: 5));
}

```

## Ejecución del Test

The screenshot shows an IDE with the following components:

- Source Editor:** Displays the code for `CalculadoraTest.java`. The code includes imports for JUnit, a private `Calculadora junit` instance, and three test methods: `setUp()`, `tearDown()`, and `testSuma()`. The `testSuma()` method calls `junit.suma(5, 3)` and asserts the result is 8 and the last operation is "Suma".
- Output Console:** Shows the results of the test run. It indicates that 5 tests were run, with 0 failures, 0 errors, and 0 skipped. The output concludes with "BUILD SUCCESS" and a total time of 2.441 s.

```

Source
4   import org.junit.jupiter.api.AfterEach;
5   import org.junit.jupiter.api.AfterAll;
6   import org.junit.jupiter.api.BeforeEach;
7   import org.junit.jupiter.api.BeforeAll;
8   import org.junit.jupiter.api.Test;
9   import static org.junit.jupiter.api.Assertions.*;
10
11  public class CalculadoraTest {
12      private Calculadora junit;
13
14      @BeforeEach
15      public void setUp() {
16          this.junit = new Calculadora();
17      }
18
19      @AfterEach
20      public void tearDown() {
21          junit.restablecer();
22      }
23
24      @Test
25      public void testSuma() {
26          float resultado = junit.suma(op1: 5, op2: 3);
27          assertEquals(expected:8, actual: resultado, delta:0);
28          assertEquals(expected:"Suma", actual: junit.getLastOp());
29      }
30
Output x
alumno - /home/alumno x  Test (CalculadoraTest) x
Results:
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
-----
BUILD SUCCESS
-----
Total time: 2.441 s

```