

Carles Bernabeu Molió

Pràctica: JUnit

**PART 1:** Captures de les funcions comentades.

```
1  /**
2   * This Java source file was generated by the Gradle 'init' task.
3   * @author Carles Bernabeu Molió
4   */
5   package CalculadoraJUnit;
6
7   /**
8   * Esta clase representa una calculadora básica que puede realizar operaciones aritméticas simples
9   * y mantener un registro del último resultado y la última operación realizada.
10  */
11  public class CalculadoraJUnit {
12      private float lastResult;
13      private String lastOp;
14
15      /**
16       * Obtiene el último resultado obtenido por la calculadora.
17       *
18       * @return el último resultado obtenido
19       */
20      public float getLastResult() {
21          return this.lastResult;
22      }
23
24      /**
25       * Obtiene la última operación realizada por la calculadora.
26       *
27       * @return la última operación realizada
28       */
29      public String getLastOp() {
30          return this.lastOp;
31      }
32
33      /**
```

```
31  }
32
33      /**
34       * Realiza la operación de suma entre dos números.
35       *
36       * @param op1 el primer operando
37       * @param op2 el segundo operando
38       * @return el resultado de la suma
39       */
40      public float suma(float op1, float op2) {
41          float result = op1 + op2;
42          this.lastResult = result;
43          this.lastOp = "Suma";
44          return result;
45      }
46
47      /**
48       * Realiza la operación de resta entre dos números.
49       *
50       * @param op1 el primer operando
51       * @param op2 el segundo operando
52       * @return el resultado de la resta
53       */
54      public float resta(float op1, float op2) {
55          float result = op1 - op2;
56          this.lastResult = result;
57          this.lastOp = "Resta";
58          return result;
59      }
60
61      /**
62       * Realiza la operación de multiplicación entre dos números.
63       *
```

```
61  /**
62   * Realiza la operación de multiplicación entre dos números.
63   *
64   * @param op1 el primer operando
65   * @param op2 el segundo operando
66   * @return el resultado de la multiplicación
67   */
68  public float multiplica(float op1, float op2) {
69      float result = op1 * op2;
70      this.lastResult = result;
71      this.lastOp = "Multiplica";
72      return result;
73  }
74
75  /**
76   * Realiza la operación de división entre dos números.
77   *
78   * @param op1 el dividendo
79   * @param op2 el divisor
80   * @return el resultado de la división
81   */
82  public float divideix(float op1, float op2) {
83      float result = op1 / op2;
84      this.lastResult = result;
85      this.lastOp = "Divideix";
86      return result;
87  }
88
89  /**
90   * Comprueba si el primer número es mayor que el segundo.
91   *
92   * @param op1 el primer número
93   * @param op2 el segundo número
```

```
86      return result;
87  }
88
89  /**
90   * Comprueba si el primer número es mayor que el segundo.
91   *
92   * @param op1 el primer número
93   * @param op2 el segundo número
94   * @return true si el primer número es mayor que el segundo, false en caso contrario
95   */
96  public boolean majorQue(float op1, float op2) {
97      return op1 > op2;
98  }
99
100  /**
101   * Restablece el último resultado y la última operación realizada a sus valores predeterminados.
102   */
103  public void restablecer() {
104      this.lastResult = 0;
105      this.lastOp = "Ninguna";
106  }
107  }
```

## PARTE 2: Captures de la classe de proves.

```

1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/UnitTests/JUnit5TestClass.java to edit this template
4   * @author Carles Bernabeu Molió
5   */
6  package TestPackages;
7
8  import CalculadoraJUnit.CalculadoraJUnit;
9  import org.junit.jupiter.api.*;
10 import java.time.LocalDateTime;
11 import static org.junit.jupiter.api.Assertions.assertEquals;
12 import static org.junit.jupiter.api.Assertions.assertFalse;
13 import static org.junit.jupiter.api.Assertions.assertTrue;
14
15 public class CalculadoraJUnitTest {
16     private CalculadoraJUnit calculadora;
17     private static LocalDateTime startTime;
18
19     @BeforeEach
20     public void setUp() {
21         calculadora = new CalculadoraJUnit();
22         System.out.println(x: "Creamos objeto Calculadora");
23     }
24
25     @AfterEach
26     public void tearDown() {
27         calculadora = null;
28         System.out.println(x: "Calculadora = null");
29     }
30
31     @BeforeAll
32     public static void fechaInicio() {
33         startTime = LocalDateTime.now();

```

```

31     @BeforeAll
32     public static void fechaInicio() {
33         startTime = LocalDateTime.now();
34         System.out.println("Inicio del conjunto de pruebas: " + startTime);
35     }
36
37     @AfterAll
38     public static void fechaFinTiempo() {
39         LocalDateTime endTime = LocalDateTime.now();
40         System.out.println("Fin del conjunto de pruebas: " + endTime);
41         long duration = java.time.Duration.between(startTime, endTime).toMillis();
42         System.out.println("Duración total del conjunto de pruebas: " + duration + " milisegundos");
43     }
44
45     @Test
46     public void testSuma() {
47         assertEquals(expected: 5, actual: calculadora.suma(op1: 2, op2: 3));
48         System.out.println(x: "Comprobamos suma");
49     }
50
51     @Test
52     public void testResta() {
53         assertEquals(expected: -1, actual: calculadora.resta(op1: 2, op2: 3));
54         System.out.println(x: "Comprobamos resta");
55     }
56
57     @Test
58     public void testMultiplica() {
59         assertEquals(expected: 6, actual: calculadora.multiplica(op1: 2, op2: 3));
60         System.out.println(x: "Comprobamos multiplicación");
61     }
62
63     @Test

```

```

61     }
62
63     @Test
64     public void testDivideix() {
65         assertEquals(expected: 2, actual: calculadora.divideix(op1: 6, op2: 3));
66         System.out.println(x: "Comprobamos división");
67     }
68
69     @Test
70     public void testMayorQue() {
71         assertTrue(condition: calculadora.mayorQue(op1: 5, op2: 3));
72         assertFalse(condition: calculadora.mayorQue(op1: 3, op2: 5));
73         System.out.println(x: "Comprobamos mayor");
74     }
75
76     @Test
77     public void testRestablir() {
78         calculadora.resta(op1: 2, op2: 3);
79         calculadora.restablecer();
80         assertEquals(expected: 0, actual: calculadora.getLastResult());
81         assertEquals(expected: "Ninguna", actual: calculadora.getLastOp());
82         System.out.println(x: "Restablecemos calculadora");
83     }
84 }
85
86
87 // TODO add test methods here.
88 // The methods must be annotated with annotation @Test. For example:
89 //
90 // @Test
91 // public void hello() {}
92
93

```

### PART 3: Captura de l'execució del test de JUnit.

The screenshot displays an IDE environment with the following components:

- Project Tree (Left):** Shows the project structure, including 'CalculadoraJUnitTest.java' under the 'Test Packages' folder.
- Source Code (Right):** Displays the source code of the test class, which includes imports for JUnit and the 'Calculadora' class, and a package declaration 'package TestPackages;'. The code defines several test methods: 'testResta()', 'testRestablir()', 'testSuma()', 'testMultiplica()', 'testMayorQue()', and 'testDivideix()'. Each method is annotated with '@Test'.
- Test Results (Bottom):** Shows the results of the test run. A green bar at the top indicates 'Tests passed: 100.00 %'. Below this, a list of test methods is shown, all marked as 'passed' with their respective execution times in parentheses. The list includes:
  - testResta() passed (0.003 s)
  - testRestablir() passed (0.003 s)
  - testSuma() passed (0.002 s)
  - testMultiplica() passed (0.003 s)
  - testMayorQue() passed (0.004 s)
  - testDivideix() passed (0.003 s)