

## Ejercicio de JUNIT:

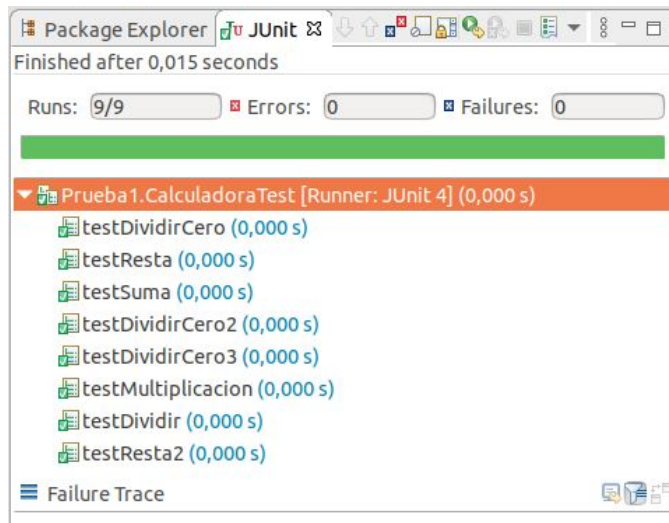
Programa con los métodos de calculadora:

```
CalculadoraMultTest.java  test.java  Ca
1  package Prueba1;
2
3  public class Calculadora {
4
5      private int num1;
6      private int num2;
7
8      public Calculadora() {}
9
10     public Calculadora(int a, int b) {
11         this.num1=a;
12         this.num2=b;
13     }
14
15     public int suma() {
16         int result = num1 + num2;
17         return result;
18     }
19
20     public int resta() {
21         int resul;
22
23         if (resta2())
24             resul = num1 - num2;
25         else
26             resul = num2 - num1;
27         return resul;
28     }
29
30     public boolean resta2() {
31         if (num1 >= num2)
32             return true;
33         else
34             return false;
35     }
36
37     public Integer divide2() {
38         if (num2 == 0)
39             return null;
40
41         int resul = num1 / num2;
42
43         return resul;
44     }
45
46     public int multiplicar() {
47         int result = num1 * num2;
48         return result;
49     }
50
51     public int dividir() {
52         int result = num1 / num2;
53         return result;
54     }
55 }
```

Casos de prueba para el programa definido anteriormente:

```
CalculadoraMultTest.java  test.java  Calculadora.java  Calculado
1 package Prueba1;
2
3 import static org.junit.Assert.*;
4
5
6
7 public class CalculadoraTest {
8
9     //Si la suma devuelve el resultado esperado está correcto.
10    @Test
11    public void testSuma() {
12        Calculadora calc = new Calculadora (20,10);
13        int resultado = calc.suma();
14        assertEquals("Fallo en la suma" ,30, resultado);
15    }
16
17    //Si la resta devuelve el resultado esperado está correcto.
18    @Test
19    public void testResta() {
20        Calculadora calc = new Calculadora (20,10);
21        int resultado = calc.resta();
22        assertEquals("Fallo en la Resta", 10, resultado);
23    }
24
25    //Si en la resta el num1 es mayor al num2 devuelve True.
26    @Test
27    public void testResta2() {
28        Calculadora calc = new Calculadora (20,10);
29        boolean resultado = calc.resta2();
30        assertTrue("Fallo en la Resta", resultado);
31    }
32
33    //Si en la resta el num1 es menor al num2 devuelve False.
34    @Test
35    public void testResta3() {
36        Calculadora calc = new Calculadora (10,20);
37        assertFalse("Resta devuelve el resultado", calc.resta2());
38    }
39
40    //Prueba para cuando el num2 sea igual a 0
41    @Test
42    public void testDividirCero() {
43        try {
44            Calculadora calc = new Calculadora (20,0);
45            int resultado = calc.dividir();
46            fail("FALLO");
47        } catch (ArithmeticException e) {
48            // TODO: handle exception
49        }
50    }
51
52
53    //Devuelve null si el num2 es igual a 0
54    @Test
55    public void testDividirCero2() {
56        Calculadora calc = new Calculadora (20,0);
57        Integer resultado = calc.divide2();
58        assertNull("FALLO", resultado);
59    }
60
61    //Devuelve NotNull si la division es correcta.
62    @Test
63    public void testDividirCero3() {
64        Calculadora calc = new Calculadora (20,1);
65        int resultado = calc.divide2();
66        assertNotNull("FALLO", resultado);
67    }
68
69    //Si la multiplicación devuelve el resultado esperado está correcto.
70    @Test
71    public void testMultiplicacion() {
72        Calculadora calc = new Calculadora (20,10);
73        int resultado = calc.multiplicar();
74        assertEquals("Fallo en la multiplicación", 200, resultado);
75    }
76
77    //Si la división devuelve el resultado esperado está correcto.
78    @Test
79    public void testDividir() {
80        Calculadora calc = new Calculadora (20,10);
81        int resultado = calc.dividir();
82        assertEquals("Fallo al dividir", 2, resultado);
83    }
84
85
86
87 }
88
```

Resultados de los casos de prueba anteriores:



Caso de prueba para la multiplicación con parámetros:

```
CalculadoraMultTest.java  test.java  Calculadora.java
1  package Prueba1;
2
3  import static org.junit.Assert.assertEquals;
4
5  import java.util.Arrays;
6  import java.util.Collection;
7
8  import org.junit.Test;
9  import org.junit.runner.RunWith;
10 import org.junit.runners.Parameterized;
11 import org.junit.runners.Parameterized.Parameters;
12
13 @RunWith(Parameterized.class)
14 class CalculadoraMultTest {
15
16     private int num1;
17     private int num2;
18     private int resul;
19
20     public CalculadoraMultTest(int num1, int num2, int resul) {
21
22         this.num1 = num1;
23         this.num2 = num2;
24         this.resul = resul;
25
26     }
27
28     @Parameters
29     public static Collection<Object[]> numeros() {
30         return Arrays.asList(new Object[][] {
31             {1, 10, 10}, {2, 10, 20}, {20, 20, 400}
32         });
33     }
34
35     @Test
36     public void testMultiplica() {
37         Calculadora calc = new Calculadora (num1, num2);
38         int resultado = calc.multiplicar();
39         assertEquals(resul, resultado);
40     }
41
42 }
43
```

Caso de prueba para la suma con parámetros:

```
CalculadoraMultTest.java CalculadoraSumaTest.java Calculadora.java
1 package Prueba1;
2
3 import static org.junit.Assert.assertEquals;
4
12
13 @RunWith(Parameterized.class)
14 public class CalculadoraSumaTest {
15
16     private int num1;
17     private int num2;
18     private int resul;
19
20     public CalculadoraSumaTest(int num1, int num2, int resul) {
21
22         this.num1 = num1;
23         this.num2 = num2;
24         this.resul = resul;
25     }
26
27
28     @Parameters
29     public static Collection<Object[]> numeros() {
30         return Arrays.asList(new Object[][] {
31             {20, 10, 30}, {50, 10, 60}, {100, 20, 120}
32         });
33     }
34
35     @Test
36     public void testSuma() {
37         Calculadora calc = new Calculadora (num1, num2);
38         int resultado = calc.suma();
39         assertEquals(resul, resultado);
40     }
41
42 }
```

Caso de prueba para la resta con parámetros:

```
CalculadoraMultTe CalculadoraSumaTe Calculadora.java Calcula
1 package Prueba1;
2
3 import static org.junit.Assert.assertEquals;
4
5 import java.util.Arrays;
6 import java.util.Collection;
7
8 import org.junit.Test;
9 import org.junit.runner.RunWith;
10 import org.junit.runners.Parameterized;
11 import org.junit.runners.Parameterized.Parameters;
12
13 @RunWith(Parameterized.class)
14 public class CalculadoraRestaTest {
15
16     private int num1;
17     private int num2;
18     private int resul;
19
20     public CalculadoraRestaTest(int num1, int num2, int resul) {
21
22         this.num1 = num1;
23         this.num2 = num2;
24         this.resul = resul;
25     }
26
27
28     @Parameters
29     public static Collection<Object[]> numeros() {
30         return Arrays.asList(new Object[][] {
31             {20, 10, 10}, {50, 10, 40}, {100, 20, 80}
32         });
33     }
34
35     @Test
36     public void testResta() {
37         Calculadora calc = new Calculadora (num1, num2);
38         int resultado = calc.resta();
39         assertEquals(resul, resultado);
40     }
41
42 }
```



Clase que ejecuta las diferentes pruebas anteriores que están en clases diferentes:

```
CalculadoraMult  *CalculadoraSum  Calculadora.  
1 package Prueba1;  
2  
3 import org.junit.runner.RunWith;  
6  
7 @RunWith(Suite.class)  
8 @SuiteClasses({ CalculadoraMultTest.class,  
9                 CalculadoraRestaTest.class,  
10                CalculadoraSumaTest.class })  
11  
12 public class TodosTests {  
13  
14 }  
15
```

Resultado de las pruebas anteriores:

Package Explorer JUnit

Finished after 0,012 seconds

Runs: 9/9 Errors: 0 Failures: 0

▼ Prueba1.TodosTests [Runner: JUnit 4] (0,001 s) Failure Trace

- ▼ Prueba1.CalculadoraMultTest (0,001 s)
  - ▶ [0] (0,001 s)
  - ▶ [1] (0,000 s)
  - ▶ [2] (0,000 s)
- ▼ Prueba1.CalculadoraRestaTest (0,000 s)
  - ▶ [0] (0,000 s)
  - ▶ [1] (0,000 s)
  - ▶ [2] (0,000 s)
- ▼ Prueba1.CalculadoraSumaTest (0,000 s)
  - ▶ [0] (0,000 s)
  - ▶ [1] (0,000 s)
  - ▶ [2] (0,000 s)