

Práctica 18

Entornos del Desarrollo

1

Se solicita programar la clase Calculadora vista en teoría de forma que, además de incluir el método suma (visto en transparencias), incluya un método para restar 2 números, otro para multiplicar 2 números y otro para dividir entre 2 números enteros.

Al crear la clase se hizo con un constructor que admite 2 variables, a y b, que son números enteros. Se crearon los métodos suma, resta, multiplicación y división que utilizan estas variables para obtener los resultados.

```
2 usages
1 public class Calculadora {
    5 usages
2     private int a;
    5 usages
3     private int b;
4
5     1 usage
6     public Calculadora(int a, int b) {
7         this.a = a;
8         this.b = b;
9     }
10
11     1 usage
12     public int suma(){
13         return this.a + this.b;
14     }
15
16     1 usage
17     public int resta(){
18         return this.a - this.b;
19     }
20
21     1 usage
22     public int multiplicacion (){
23         return this.a * this.b;
24     }
25
26     1 usage
27     public int division(){
28         return this.a / this.b;
29     }
30 }
```

Deberás realizar los test con JUnit a todos los métodos indicados anteriormente.

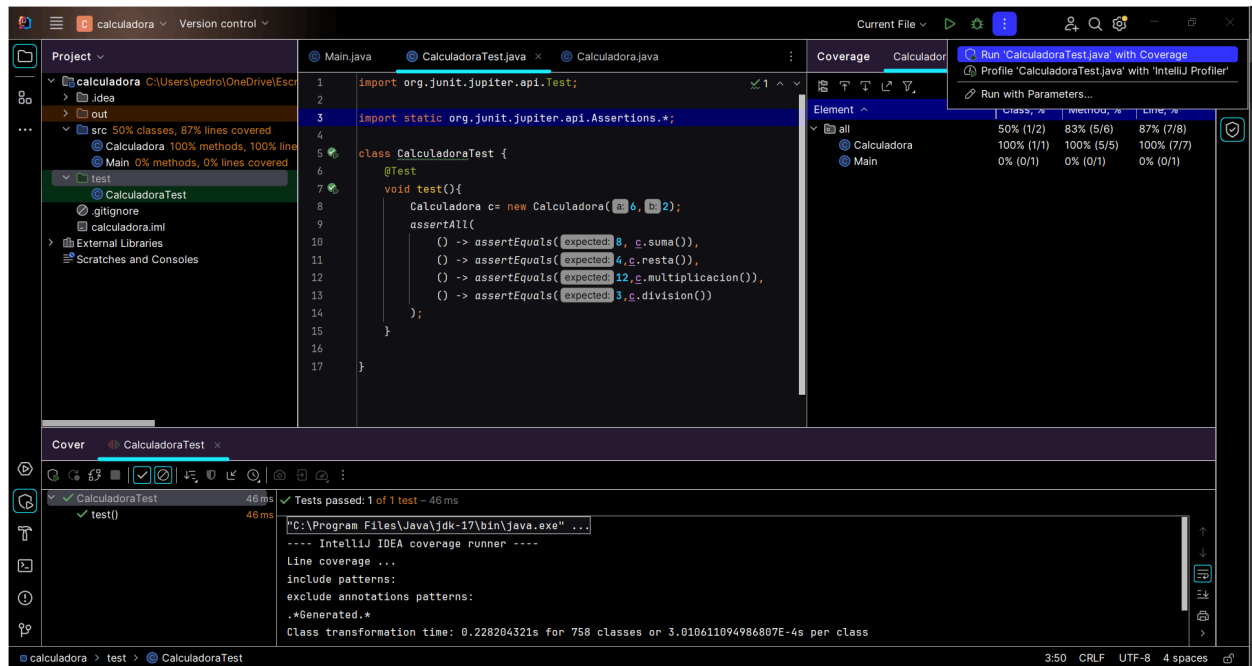
Creamos la carpeta test, a la que marcamos como **Test Sources Root**. A continuación en esa carpeta creamos la clase **CalculadoraTest** que marcamos como clase test y escribimos el siguiente código



The screenshot shows an IDE with a project named 'calculadora'. The project structure on the left includes a 'test' folder under 'src', which contains the 'CalculadoraTest' class. The code in the editor is as follows:

```
1 import org.junit.jupiter.api.Test;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class CalculadoraTest {
6     @Test
7     void test(){
8         Calculadora c= new Calculadora(a: 6, b: 2);
9         assertEquals(8, c.suma());
10        assertEquals(4, c.resta());
11        assertEquals(12, c.multiplicacion());
12        assertEquals(3, c.division());
13    }
14 }
15
16
17 }
```

Una vez escritos los test iniciamos la clase tal y como se indica en la siguiente imagen y comprobamos los resultados.



3

Comprueba que los test pasan satisfactoriamente con los resultados esperados y fuerza en todos los casos con valores que den como resultado test fallido. Prueba también con dividir entre 0. (Mínimo 5 pruebas con cada clase).

Las pruebas correctas pasaron los test esperados(con resultados correctos). Para ello se crearon nuevos objetos Calculadora con diferentes valores de variables y se introdujeron test para cada método y los resultados esperados.

The screenshot shows the IntelliJ IDEA IDE with a Java test file named `CalculadoraTest.java`. The code contains several assertions for a calculator class. The coverage report on the right shows that the `Calculadora` class has 50% line coverage (1/2), 83% method coverage (5/6), and 87% line coverage (7/8). The `Main` class has 0% coverage for all metrics.

Element	Class, %	Method, %	Line, %
all	50% (1/2)	83% (5/6)	87% (7/8)
Calculadora	100% (1/1)	100% (5/5)	100% (7/7)
Main	0% (0/1)	0% (0/1)	0% (0/1)

The test results at the bottom show that the `test()` method passed, with 1 of 1 tests passing in 50 ms.

Se probó modificar uno de los test de forma que todos sus resultados fueran incorrectos y se obtuvo un aviso de fallo de test para cada operación

Suma

The screenshot shows the IntelliJ IDEA IDE with the same Java test file. The test results at the bottom show that the `test()` method failed, with 1 of 1 tests failing in 21 ms. The error message indicates an `AssertionFailedError` with the expected value of 5 and the actual value of 9.

```
org.opentest4j.AssertionFailedError:
Expected :5
Actual   :9
<Click to see difference>

<5 internal lines>
> at CalculadoraTest.lambda$test$12(CalculadoraTest.java:31) <12 internal lines>
> at CalculadoraTest.test(CalculadoraTest.java:30) <31 internal lines>
> at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
> at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>

org.opentest4j.AssertionFailedError:
Expected :7
```

Resta

```
✖ Tests failed: 1 of 1 test - 21 ms

org.opentest4j.AssertionFailedError:
Expected :7
Actual   :3
<Click to see difference>

> <5 internal lines>
>   at CalculadoraTest.lambda$test$13(CalculadoraTest.java:32) <12 internal lines>
>   at CalculadoraTest.test(CalculadoraTest.java:30) <31 internal lines>
>   at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
>   at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>
```

Multipliación

```
✖ Tests failed: 1 of 1 test - 21 ms

org.opentest4j.AssertionFailedError:
Expected :88
Actual   :18
<Click to see difference>

> <5 internal lines>
>   at CalculadoraTest.lambda$test$14(CalculadoraTest.java:33) <12 internal lines>
>   at CalculadoraTest.test(CalculadoraTest.java:30) <31 internal lines>
>   at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
>   at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>
```

División

```
org.opentest4j.AssertionFailedError:
Expected :6
Actual   :2
<Click to see difference>

> <5 internal lines>
>   at CalculadoraTest.lambda$test$15(CalculadoraTest.java:34) <12 internal lines>
>   at CalculadoraTest.test(CalculadoraTest.java:30) <31 internal lines>
>   at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
>   at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>
```

Dividir por 0

En este caso hemos modificado otra prueba y se le ha dado a la variable b el valor 0. El test falla y devuelve el siguiente resultado, advirtiéndolo de la división entre 0:

```
"C:\Program Files\Java\jdk-17\bin\java.exe" ...  
---- IntelliJ IDEA coverage runner ----  
Line coverage ...  
include patterns:  
exclude annotations patterns:  
.*Generated.*  
  
java.lang.ArithmeticException: / by zero  
  
    at Calculadora.division(Calculadora.java:20)  
    at CalculadoraTest.lambda$test$19(CalculadoraTest.java:41) <12 internal lines>  
    at CalculadoraTest.test(CalculadoraTest.java:37) <31 internal lines>  
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>  
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>
```