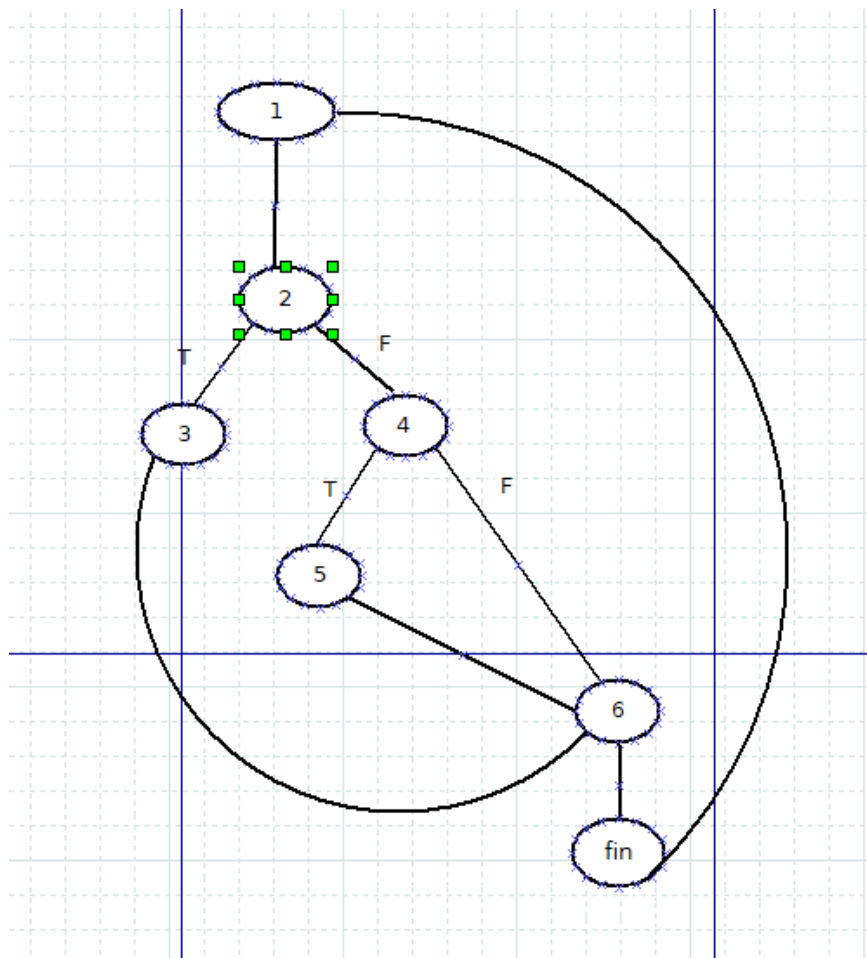


## Ejercicio 1. Consultar Positivo

```
public int consultarPositivo(int num){  
    int res=0; 1  
    if (num<0){ 2  
        res = -1; 3  
    }else if(num>0){ 4  
        res =1; 5  
    }  
    return res; 6  
}
```



**Complejidad de McCabe:**  $Aristas-nodos+2=9-7+2=4$

**Caminos Independientes=3**

## Ejercicio 2. Factura eléctrica

```
public int calcularPrezokWh(){
    int prezokWh=0;
    if (consumokWh<=300){
        prezokWh=9;
    }else if(consumokWh<=600){
        prezokWh=8;
    }else if(consumokWh<=1000){
        prezokWh=6;
    }else if (consumokWh<=2000){
        prezokWh=5;
    }
    return prezokWh;
}
```

### Clases de equivalencia:

**Rango de valores válidos: Consumo=0kWh-2000kWh**

**Rango de valores no válidos:Consumo<0 o Consumo>2000KWh**

### Valores límite:

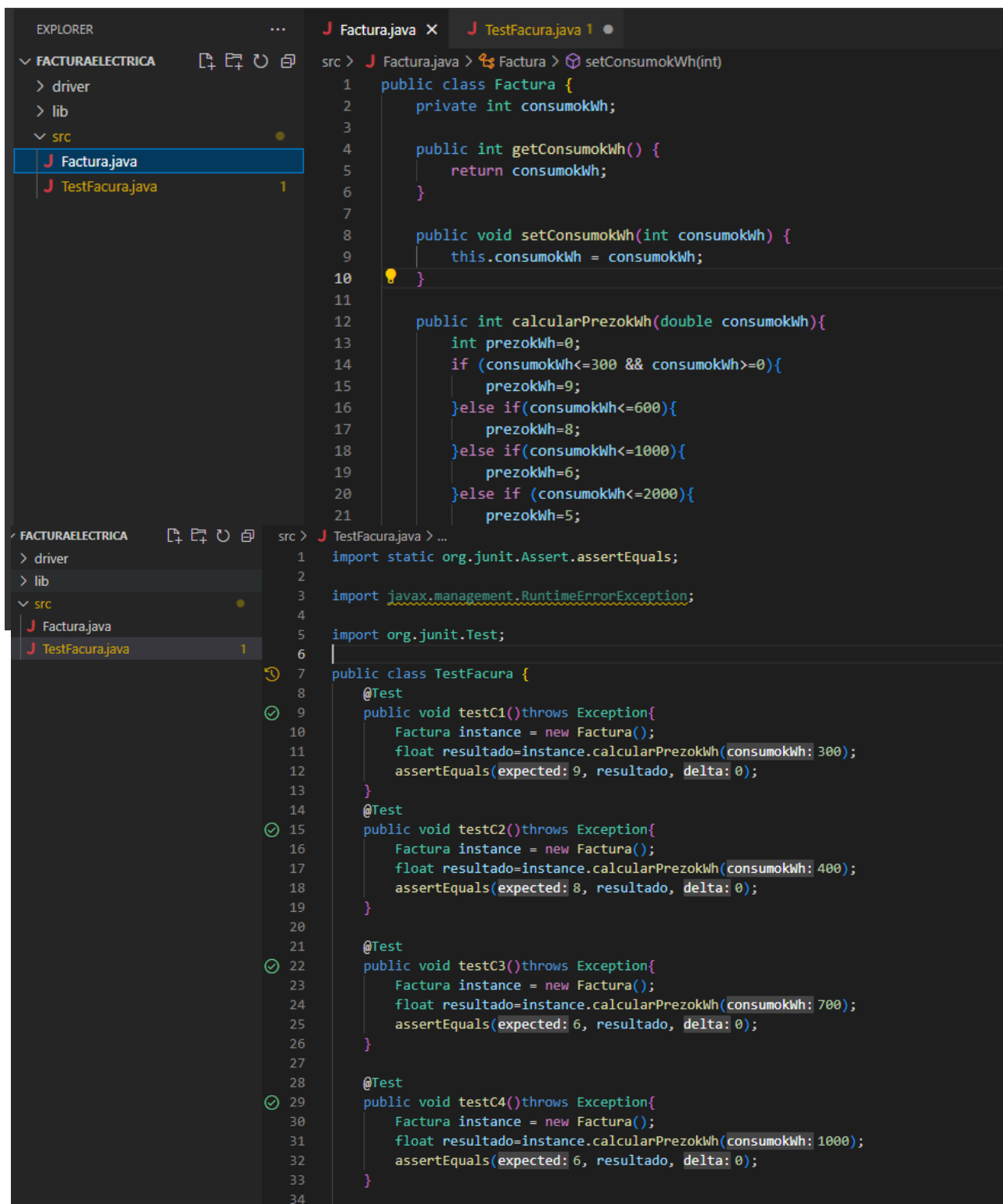
Min Permitido=0    Min no permitido=-0,1

Max Permitido=2000    Max no permitido=2000,1

## Conjetura de errores

- Consumos negativos.
- Caracteres distintos a números.
- Consumos mayores al máximo permitido en este caso 2000.

## Comprobación JUNIT:



The screenshot shows an IDE with two files open: `Factura.java` and `TestFactura.java`. The `Factura` class has methods for getting and setting consumption, and a method for calculating the price based on consumption. The `TestFactura` class contains four test methods that verify the calculation of the price for different consumption values.

```
src > J Factura.java > Factura > setConsumokWh(int)
1 public class Factura {
2     private int consumokWh;
3
4     public int getConsumokWh() {
5         return consumokWh;
6     }
7
8     public void setConsumokWh(int consumokWh) {
9         this.consumokWh = consumokWh;
10    }
11
12    public int calcularPrezokWh(double consumokWh){
13        int prezokWh=0;
14        if (consumokWh<=300 && consumokWh>=0){
15            prezokWh=9;
16        }else if(consumokWh<=600){
17            prezokWh=8;
18        }else if(consumokWh<=1000){
19            prezokWh=6;
20        }else if (consumokWh<=2000){
21            prezokWh=5;
22        }
23    }
24 }
```

```
src > J TestFactura.java > ...
1 import static org.junit.Assert.assertEquals;
2
3 import javax.management.RuntimeErrorException;
4
5 import org.junit.Test;
6
7 public class TestFactura {
8     @Test
9     public void testC1()throws Exception{
10         Factura instance = new Factura();
11         float resultado=instance.calcularPrezokWh(consumokWh: 300);
12         assertEquals(expected: 9, resultado, delta: 0);
13     }
14     @Test
15     public void testC2()throws Exception{
16         Factura instance = new Factura();
17         float resultado=instance.calcularPrezokWh(consumokWh: 400);
18         assertEquals(expected: 8, resultado, delta: 0);
19     }
20
21     @Test
22     public void testC3()throws Exception{
23         Factura instance = new Factura();
24         float resultado=instance.calcularPrezokWh(consumokWh: 700);
25         assertEquals(expected: 6, resultado, delta: 0);
26     }
27
28     @Test
29     public void testC4()throws Exception{
30         Factura instance = new Factura();
31         float resultado=instance.calcularPrezokWh(consumokWh: 1000);
32         assertEquals(expected: 6, resultado, delta: 0);
33     }
34 }
```

```

34
35     @Test
36     public void testC5() throws Exception {
37         Factura instance = new Factura();
38         float resultado = instance.calcularPrezokWh(consumokWh: 1500);
39         assertEquals(expected: 5, resultado, delta: 0);
40     }
41
42     @Test
43     public void testC6() throws Exception {
44         Factura instance = new Factura();
45         float resultado = instance.calcularPrezokWh(consumokWh: 2000);
46         assertEquals(expected: 5, resultado, delta: 0);
47     }
48
49     @Test(expected = Exception.class)
50     public void testC7() throws Exception {
51         Factura instance = new Factura();
52         float resultado = instance.calcularPrezokWh(consumokWh: 2500);
53         assertEquals(expected: 0, resultado, delta: 0);
54     }
55
56
57
58 }
59

```

La última prueba no la pasa porque es un valor mayor al límite.

### Ejercicio 3. Arrays

```

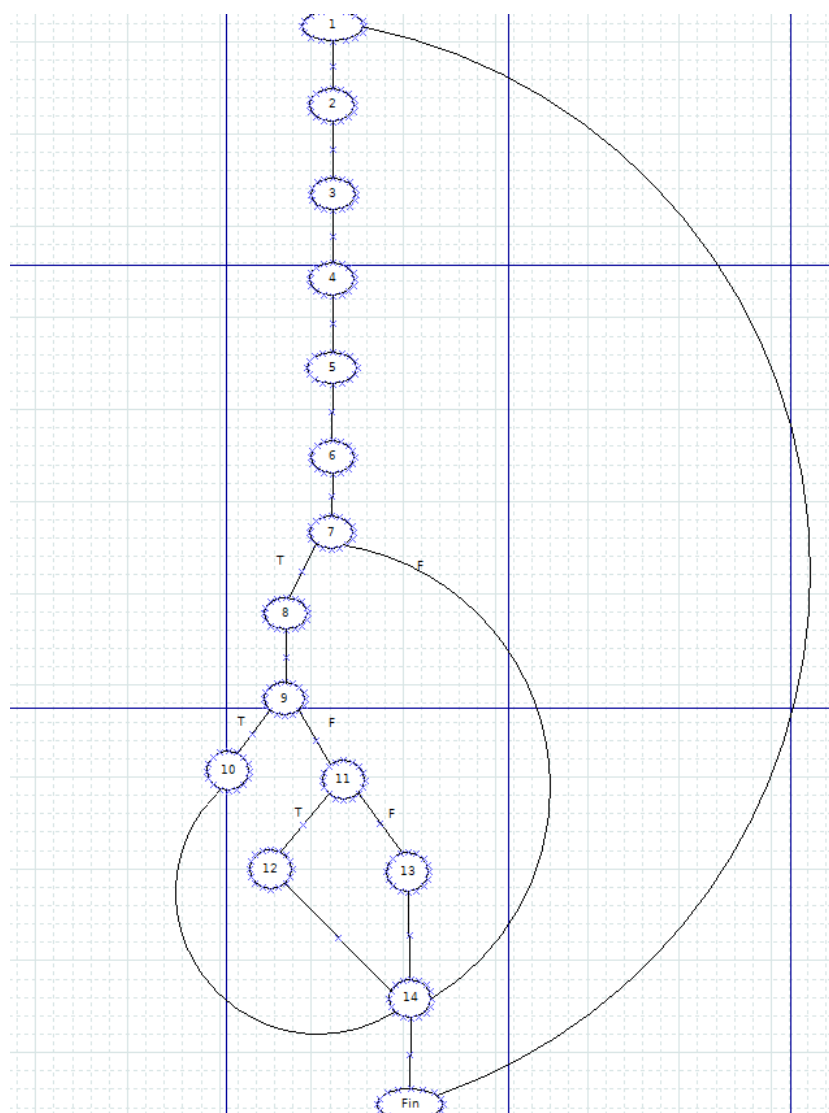
public boolean busca(char c, char[] v) {
    int a, 1
    int z; 2
    int m; 3
    a = 0; 4
    z = v.length - 1; 5
    boolean resultado = false; 6
    while (a <= z && resultado == false) { 7
        m = (a + z) / 2; 8

```

```

if (v[m] == c) { 9
    resultado=true; 10
}
else
{
    if (v[m] < c) { 11
        a = m + 1; 12
    }
    else{
        z = m - 1; 13
    }
}
}
return resultado; 14

```



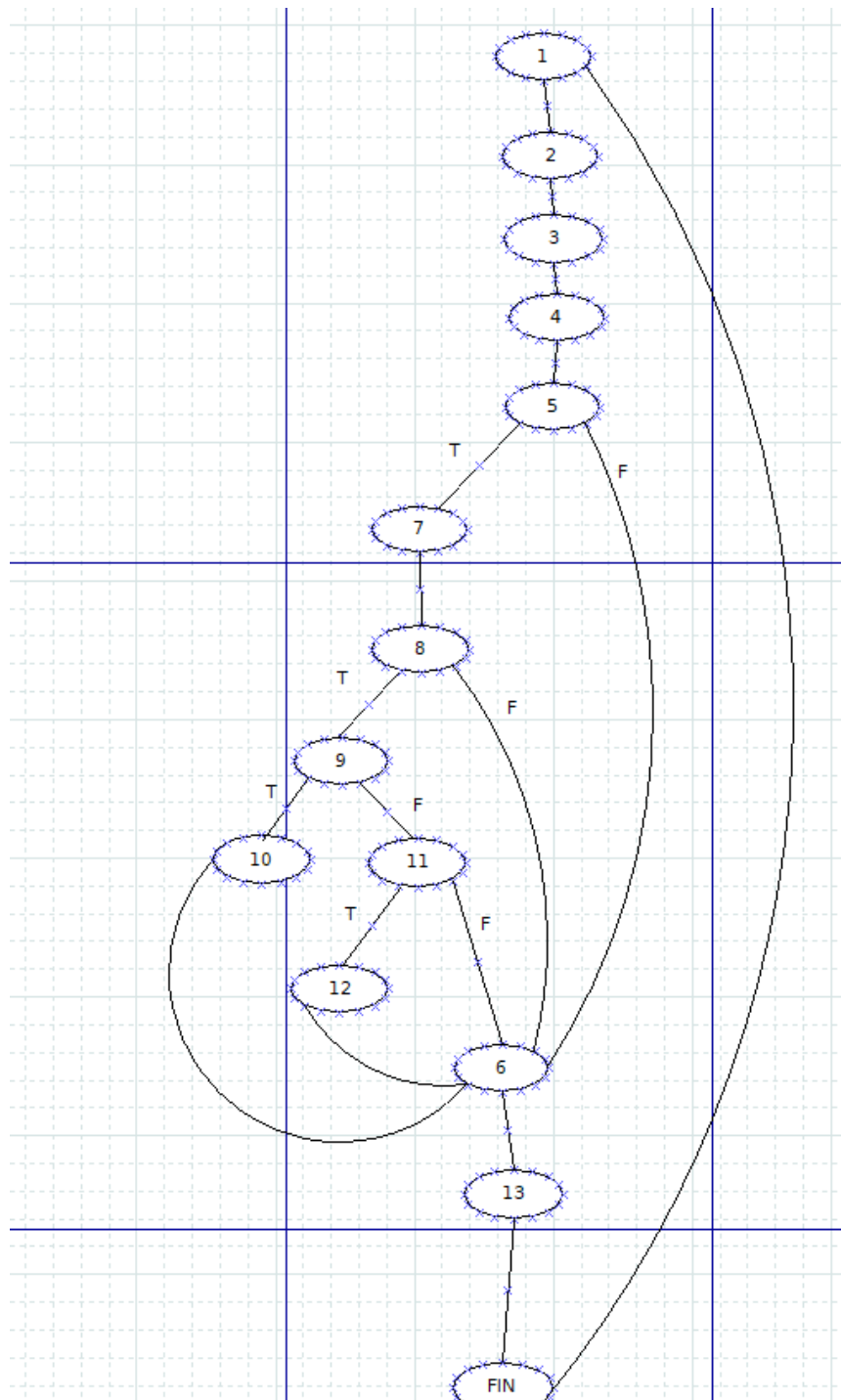
}

**Complejidad de McCabe:**  $Aristas-nodos+2=17-15+2=4$

**Caminos Independientes=3**

#### Ejercicio 4. Acrónimos

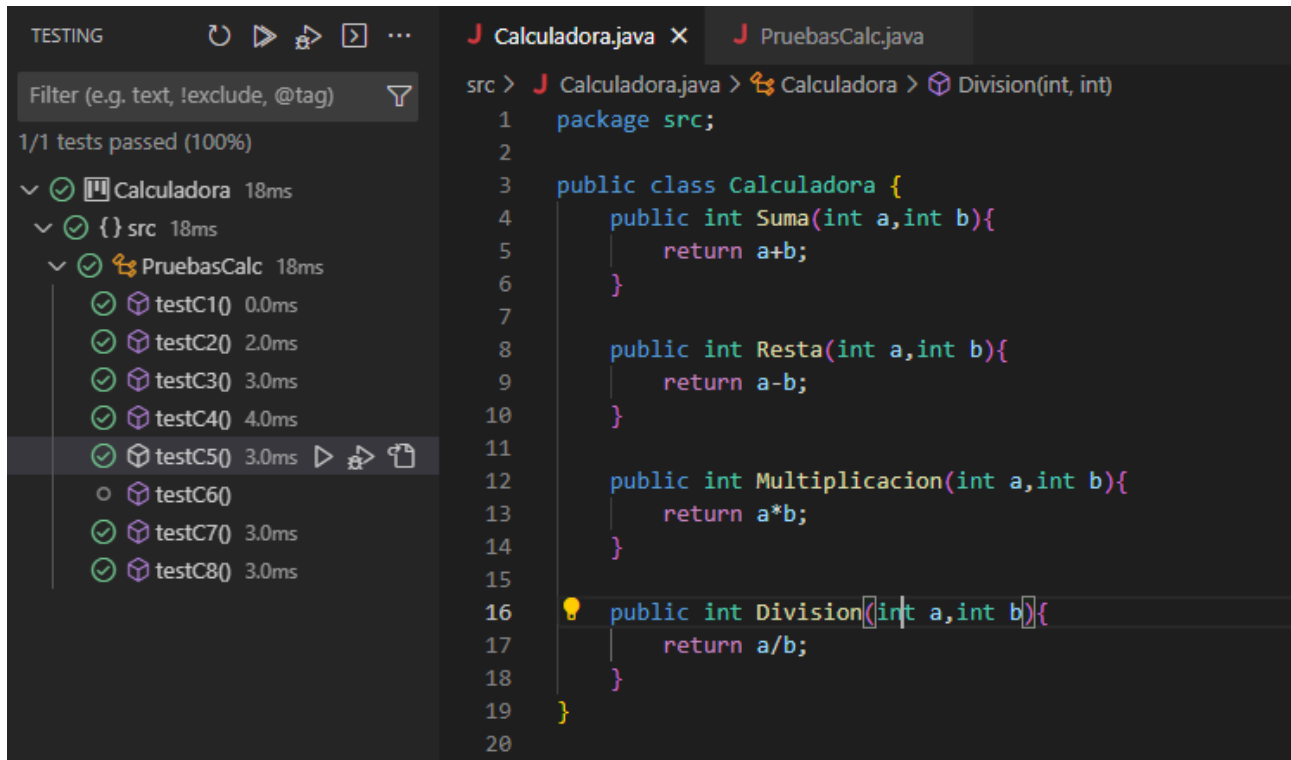
```
public String obterAcronimo(String cadena){  
    String resultado=""; 1  
    char caracter; 2  
    int n=cadena.length(); 3  
    for(int i=0;i<n;i++){ 4  
        int i=0 4  
        i<n 5  
        caracter=cadena.charAt(i); 7  
        i++ 6  
        if(caracter!=' '){ 8  
            if (i==0){ 9  
                resultado=resultado+caracter+'.';10  
            }  
            else{  
                if(cadena.charAt(i-1)==' '){ 11  
                    resultado=resultado+caracter+'.'; 12  
                }  
            }  
        }  
    }  
    return resultado; 13  
}
```



**Complejidad de McCabe:**  $\text{Aristas} - \text{nodos} + 2 = 18 - 14 + 2 = 6$

**Caminos Independientes=4**

## Ejercicio 5: Pruebas unitarias con JUnit



TESTING

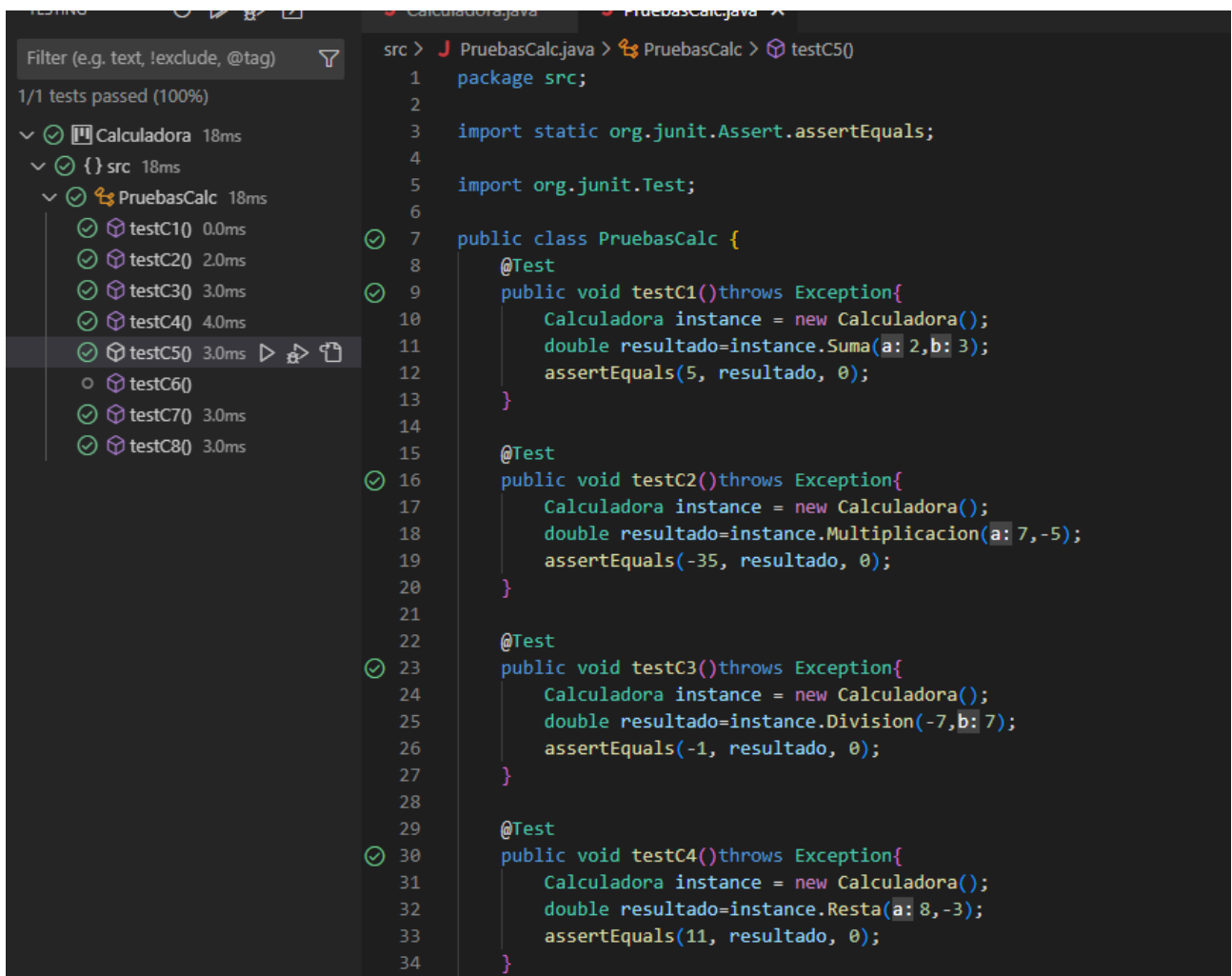
Filter (e.g. text, !exclude, @tag)

1/1 tests passed (100%)

- ✓ Calculadora 18ms
  - ✓ {} src 18ms
    - ✓ PruebasCalc 18ms
      - ✓ testC1() 0.0ms
      - ✓ testC2() 2.0ms
      - ✓ testC3() 3.0ms
      - ✓ testC4() 4.0ms
      - ✓ testC5() 3.0ms
      - testC6()
      - ✓ testC7() 3.0ms
      - ✓ testC8() 3.0ms

src > Calculadora.java > Calculadora > Division(int, int)

```
1 package src;
2
3 public class Calculadora {
4     public int Suma(int a,int b){
5         return a+b;
6     }
7
8     public int Resta(int a,int b){
9         return a-b;
10    }
11
12    public int Multiplicacion(int a,int b){
13        return a*b;
14    }
15
16    public int Division(int a,int b){
17        return a/b;
18    }
19 }
20
```



TESTING

Filter (e.g. text, !exclude, @tag)

1/1 tests passed (100%)

- ✓ Calculadora 18ms
  - ✓ {} src 18ms
    - ✓ PruebasCalc 18ms
      - ✓ testC1() 0.0ms
      - ✓ testC2() 2.0ms
      - ✓ testC3() 3.0ms
      - ✓ testC4() 4.0ms
      - ✓ testC5() 3.0ms
      - testC6()
      - ✓ testC7() 3.0ms
      - ✓ testC8() 3.0ms

src > PruebasCalc.java > PruebasCalc > testC5()

```
1 package src;
2
3 import static org.junit.Assert.assertEquals;
4
5 import org.junit.Test;
6
7 public class PruebasCalc {
8     @Test
9     public void testC1()throws Exception{
10         Calculadora instance = new Calculadora();
11         double resultado=instance.Suma(a: 2,b: 3);
12         assertEquals(5, resultado, 0);
13     }
14
15     @Test
16     public void testC2()throws Exception{
17         Calculadora instance = new Calculadora();
18         double resultado=instance.Multiplicacion(a: 7,-5);
19         assertEquals(-35, resultado, 0);
20     }
21
22     @Test
23     public void testC3()throws Exception{
24         Calculadora instance = new Calculadora();
25         double resultado=instance.Division(-7,b: 7);
26         assertEquals(-1, resultado, 0);
27     }
28
29     @Test
30     public void testC4()throws Exception{
31         Calculadora instance = new Calculadora();
32         double resultado=instance.Resta(a: 8,-3);
33         assertEquals(11, resultado, 0);
34     }
35 }
```



```
@Test
public void testC5()throws Exception{
    Calculadora instance = new Calculadora();
    double resultado=instance.Suma(a: 9,b: 3);
    assertEquals(12, resultado, 0);
}

@Test
public void testC6()throws Exception{
    Calculadora instance = new Calculadora();
    double resultado=instance.Multiplicacion(a: 5,b: 2);
    assertEquals(10, resultado, 0);
}

@Test
public void testC7()throws Exception{
    Calculadora instance = new Calculadora();
    double resultado=instance.Division(-10,b: 5);
    assertEquals(-2, resultado, 0);
}

@Test
public void testC8()throws Exception{
    Calculadora instance = new Calculadora();
    double resultado=instance.Multiplicacion(a: 2,-1);
    assertEquals(-2, resultado, 0);
}
}
```

**f**