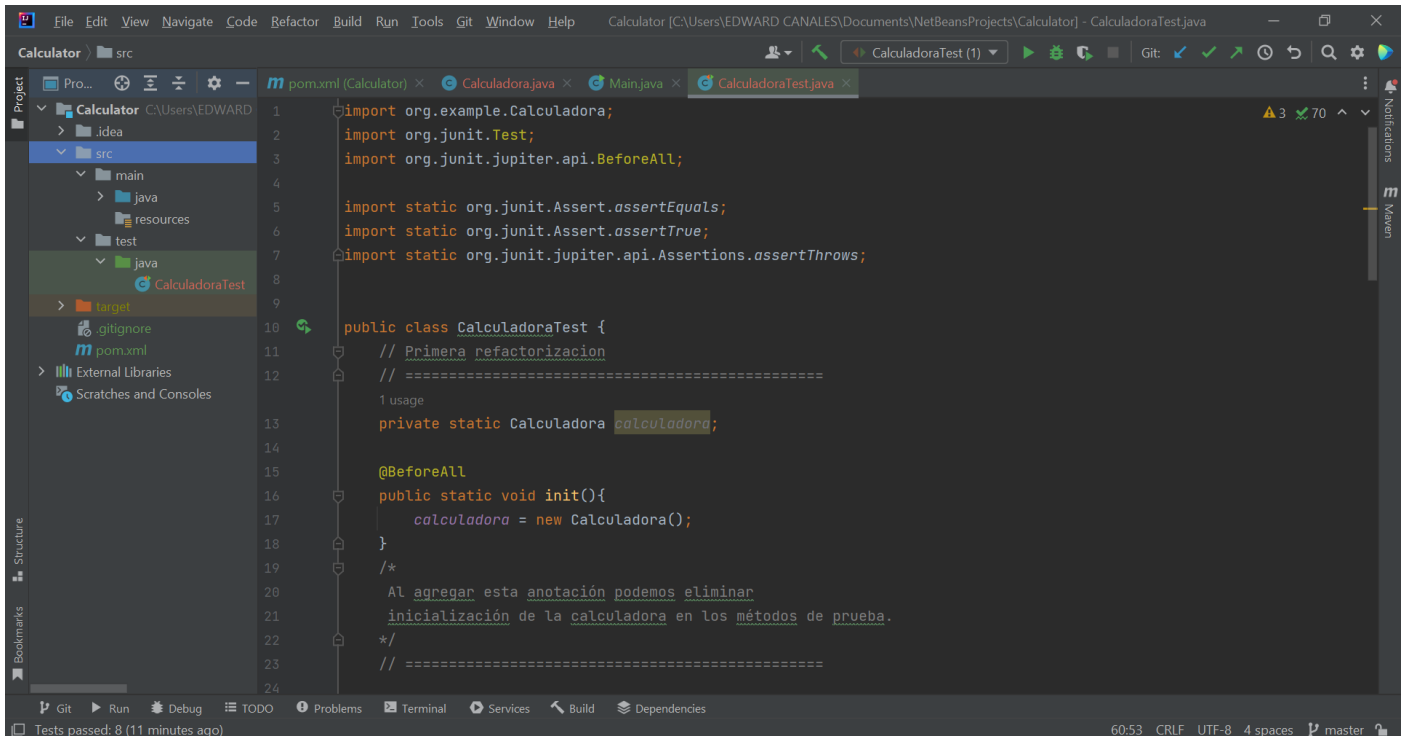
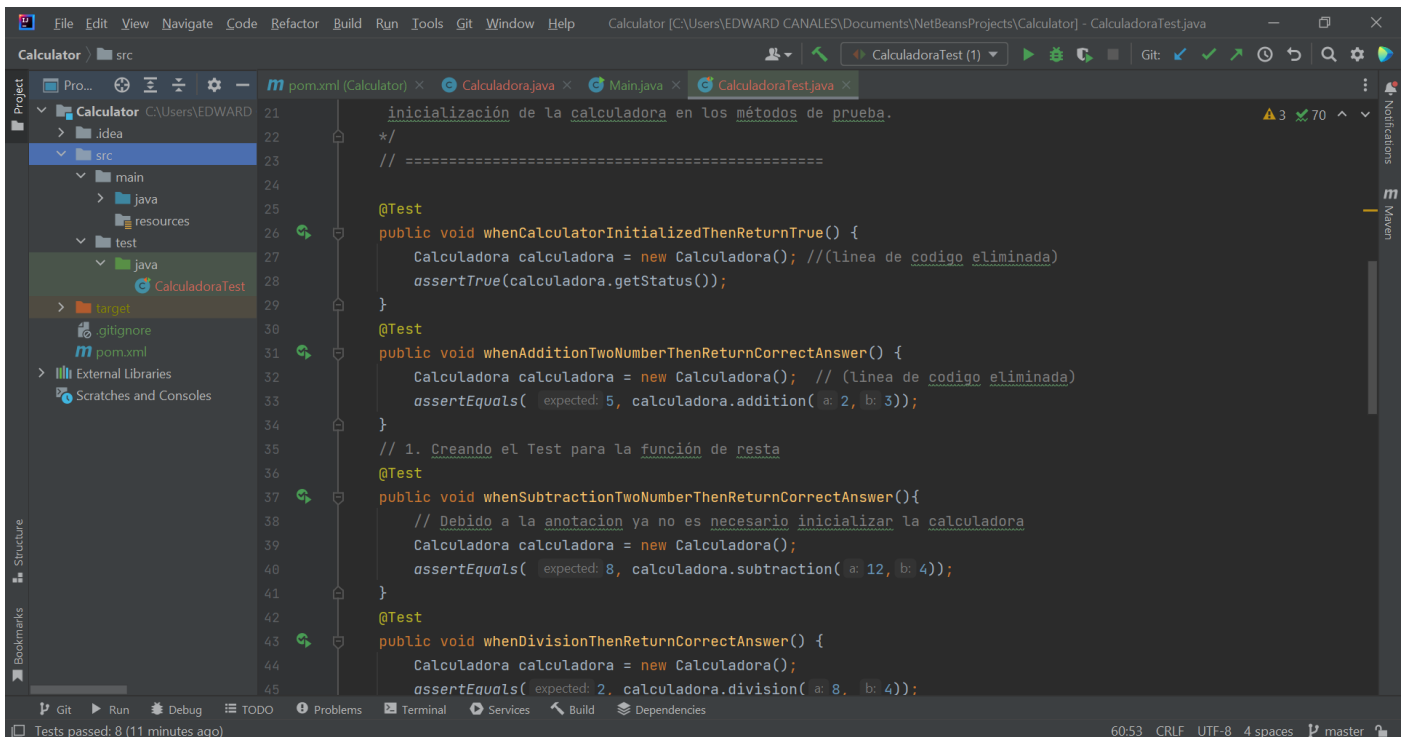


TEST



```
1 import org.example.Calculadora;
2 import org.junit.Test;
3 import org.junit.jupiter.api.BeforeAll;
4
5 import static org.junit.Assert.assertEquals;
6 import static org.junit.Assert.assertTrue;
7 import static org.junit.jupiter.api.Assertions.assertThrows;
8
9
10 public class CalculadoraTest {
11     // Primera refactorización
12     // =====
13     1 usage
14     private static Calculadora calculadora;
15
16     @BeforeAll
17     public static void init(){
18         calculadora = new Calculadora();
19     }
20     /*
21     Al agregar esta anotación podemos eliminar
22     inicialización de la calculadora en los métodos de prueba.
23     */
24     // =====
```

Tests passed: 8 (11 minutes ago)



```
21 inicialización de la calculadora en los métodos de prueba.
22 */
23 // =====
24
25 @Test
26 public void whenCalculatorInitializedThenReturnTrue() {
27     Calculadora calculadora = new Calculadora(); //(línea de código eliminada)
28     assertTrue(calculadora.getStatus());
29 }
30
31 @Test
32 public void whenAdditionTwoNumberThenReturnCorrectAnswer() {
33     Calculadora calculadora = new Calculadora(); //(línea de código eliminada)
34     assertEquals( expected: 5, calculadora.addition( a: 2, b: 3));
35 }
36 // 1. Creando el Test para la función de resta
37 @Test
38 public void whenSubtractionTwoNumberThenReturnCorrectAnswer(){
39     // Debido a la anotación ya no es necesario inicializar la calculadora
40     Calculadora calculadora = new Calculadora();
41     assertEquals( expected: 8, calculadora.subtraction( a: 12, b: 4));
42 }
43
44 @Test
45 public void whenDivisionThenReturnCorrectAnswer() {
46     Calculadora calculadora = new Calculadora();
47     assertEquals( expected: 2, calculadora.division( a: 8, b: 4));
48 }
```

Tests passed: 8 (11 minutes ago)

```
Calculator > src
pom.xml (Calculator) x Calculadora.java x Main.java x CalculadoraTest.java x
36
37
38 @Test
39 public void whenSubtractionTwoNumberThenReturnCorrectAnswer(){
40 // Debido a la anotacion ya no es necesario inicializar la calculadora
41 Calculadora calculadora = new Calculadora();
42 assertEquals( expected: 8, calculadora.subtraction( a: 12, b: 4));
43 }
44
45 @Test
46 public void whenDivisionThenReturnCorrectAnswer() {
47 Calculadora calculadora = new Calculadora();
48 assertEquals( expected: 2, calculadora.division( a: 8, b: 4));
49 }
50
51 @Test
52 public void whenDivisionByZeroThenThrowException() {
53 Calculadora calculadora = new Calculadora();
54 Throwable exception = assertThrows(IllegalArgumentException.class, () -> {
55     calculadora.division( a: 5, b: 0);
56 });
57 assertEquals( expected: "No se puede dividir por cero", exception.getMessage());
58 }
59
60 // Agregando más funciones a la clase calculadora utilizando los principios de TDD
61 // =====
62 @Test
63 public void whenMultiplicationThenReturnCorrectAnswer(){
64 Calculadora calculadora = new Calculadora();
65
66 Tests passed: 8 (11 minutes ago) 60:53 CRLF UTF-8 4 spaces master
```

```
Calculator > src
pom.xml (Calculator) x Calculadora.java x Main.java x CalculadoraTest.java x
51
52     calculadora.division( a: 5, b: 0);
53 });
54 assertEquals( expected: "No se puede dividir por cero", exception.getMessage());
55 }
56
57 // Agregando más funciones a la clase calculadora utilizando los principios de TDD
58 // =====
59 @Test
60 public void whenMultiplicationThenReturnCorrectAnswer(){
61 Calculadora calculadora = new Calculadora();
62 assertEquals( expected: 21, calculadora.multiplication( a: 3, b: 7));
63 }
64
65 @Test
66 public void whenResidueThenReturnCorrectAnswer(){
67 Calculadora calculadora = new Calculadora();
68 assertEquals( expected: 2, calculadora.residue( a: 38, b: 3));
69 }
70
71 @Test
72 public void whenPowerThenReturnCorrectAnswer(){
73 Calculadora calculadora = new Calculadora();
74 assertEquals( expected: 27, calculadora.power( a: 3, b: 3));
75 }
76
77 Tests passed: 8 (12 minutes ago) 60:53 CRLF UTF-8 4 spaces master
```

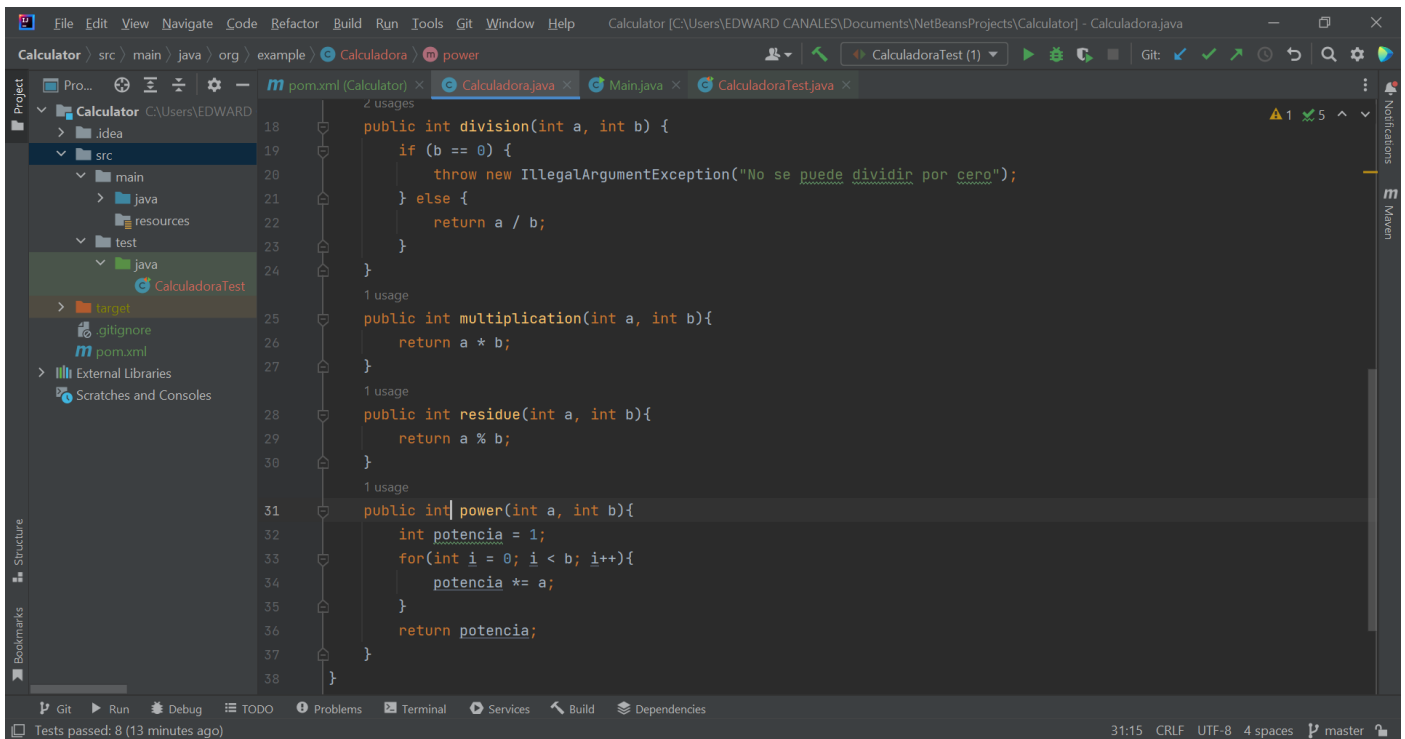
```
62 }
63
64 @Test
65 public void whenResidueThenReturnCorrectAnswer(){
66     Calculadora calculadora = new Calculadora();
67     assertEquals( expected: 2, calculadora.residue( a: 38, b: 3));
68 }
69
70 @Test
71 public void whenPowerThenReturnCorrectAnswer(){
72     Calculadora calculadora = new Calculadora();
73     assertEquals( expected: 27, calculadora.power( a: 3, b: 3));
74 }
75
76 /*
77  *
78  * COMENTARIO
79  *
80  * 1) Al ejecutar el programa pese a que está la anotación @BeforeAll
81  * me salia mensaje de error diciendo que la variable calculador tiene valor null,
82  * no pude solucionarlo y por eso tuve que inicializar el objeto calculadora en prueba.
83  *
84  * 2) Agregué primero las pruebas para las funciones multiplicación, residuo y potencia
85  * y luego implementé las funciones respectivamente. Corrí las pruebas y todas salieron
86  * en color verde, es decir, fueron exitosas todas las pruebas
87  *
88  */
89 */
```

Tests passed: 8 (12 minutes ago) 60:53 CRLF UTF-8 4 spaces master

PRODUCCION

```
1 package org.example;
2
3 public class Calculadora {
4     private boolean status;
5
6     public Calculadora() {
7         this.status = true;
8     }
9
10    public boolean getStatus() {
11        return status;
12    }
13
14    public int addition(int a, int b) {
15        return a + b;
16    }
17
18    public int subtraction(int a, int b){
19        return a - b;
20    }
21
22    public int division(int a, int b) {
23        if (b == 0) {
24
25        }
26    }
27 }
```

Tests passed: 8 (12 minutes ago) 31:15 CRLF UTF-8 4 spaces master



PRUEBAS EN VERDE

