



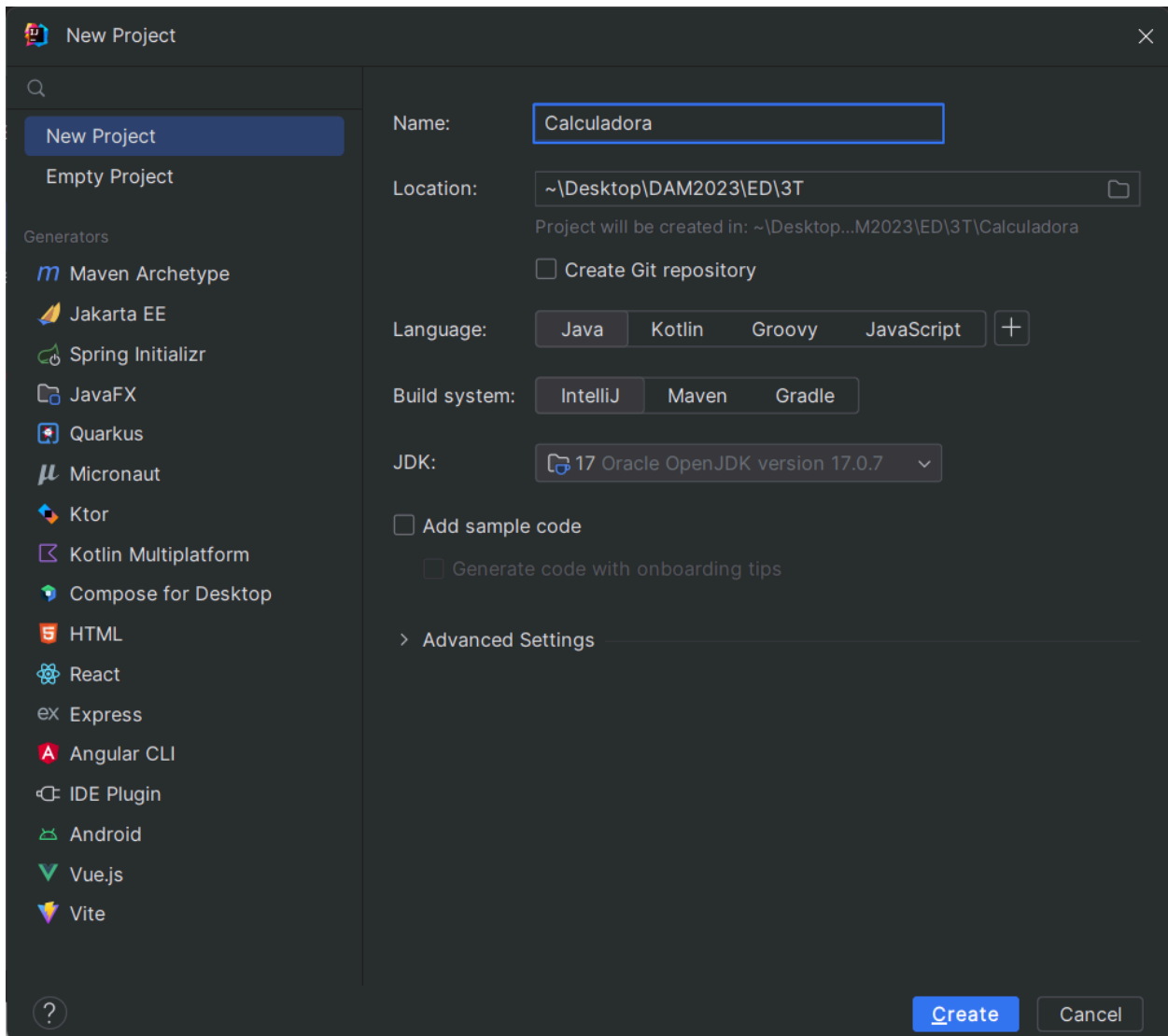
ENTORNOS DE DESARROLLO

PRÁCTICA 18

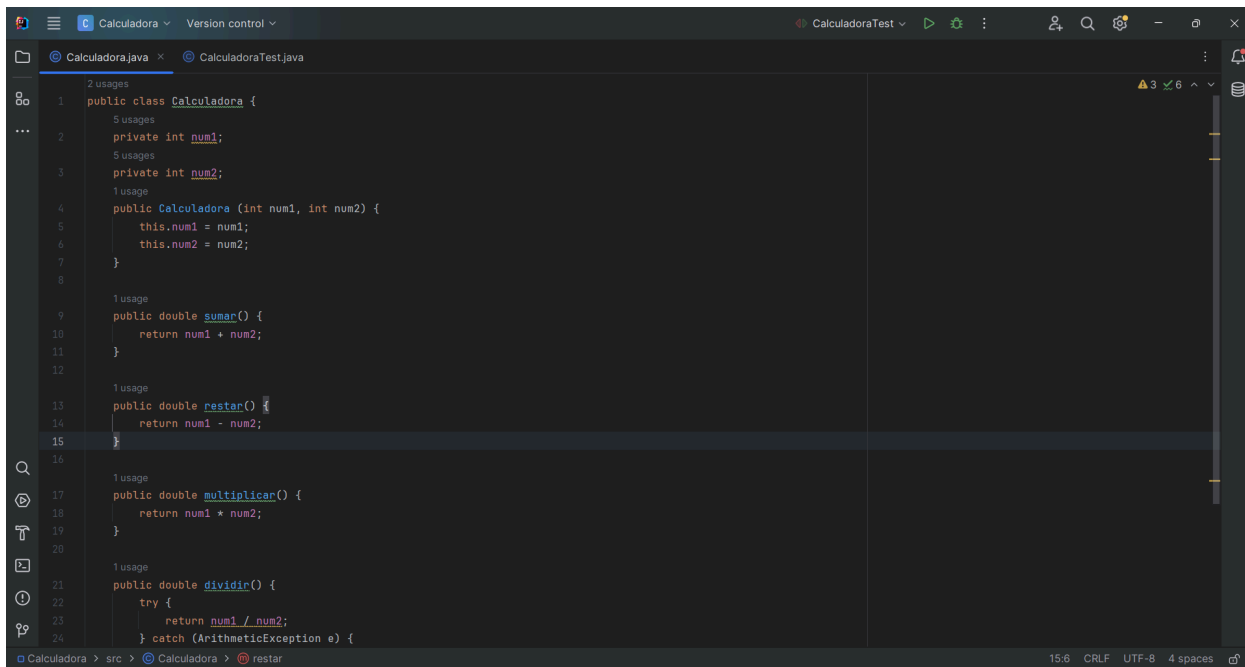
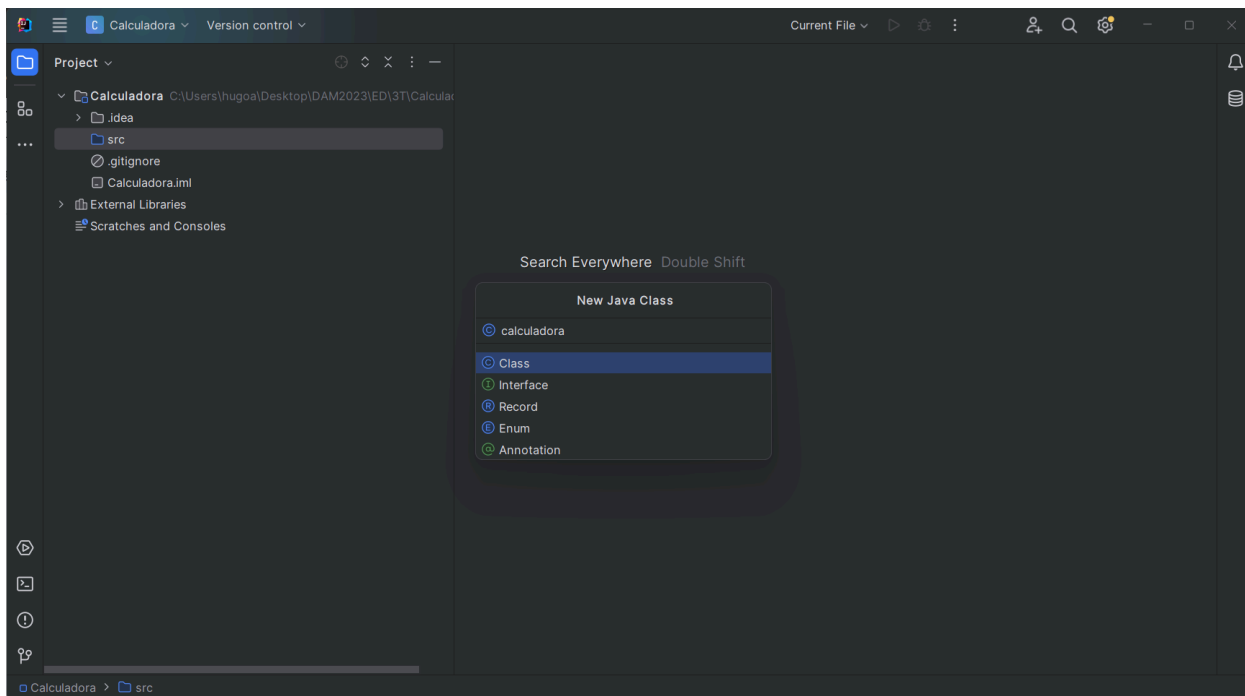


EJERCICIO 1.

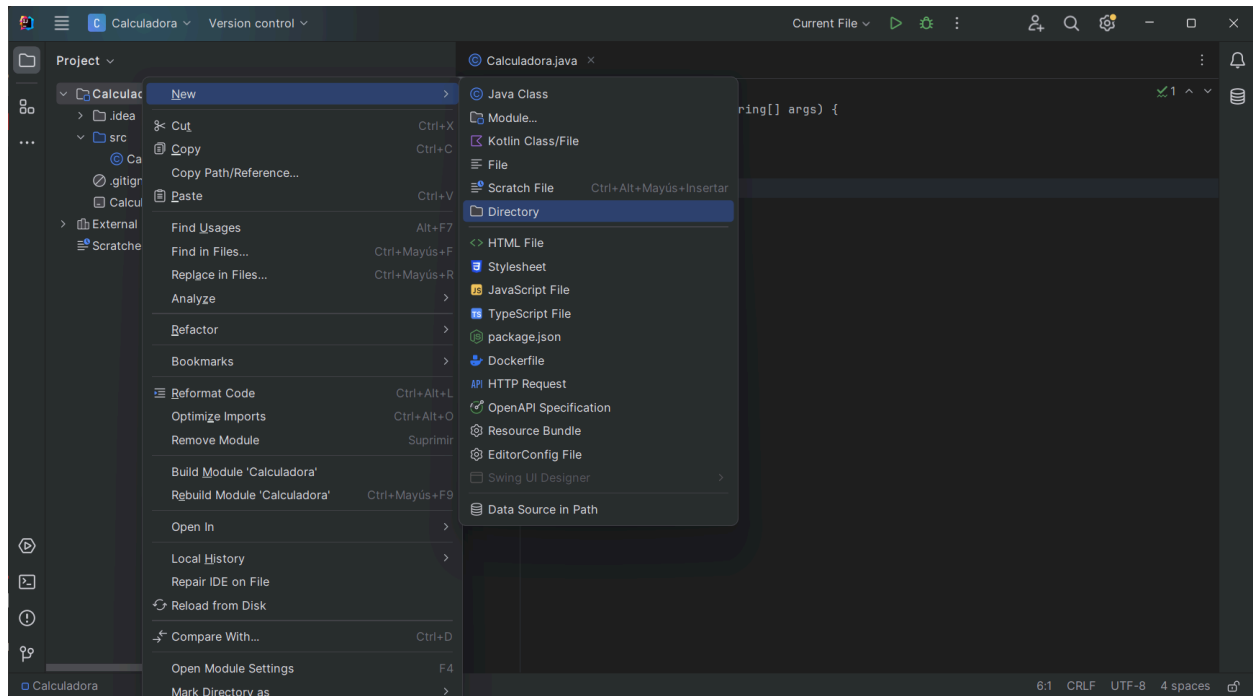
Primero deberemos crear el proyecto.



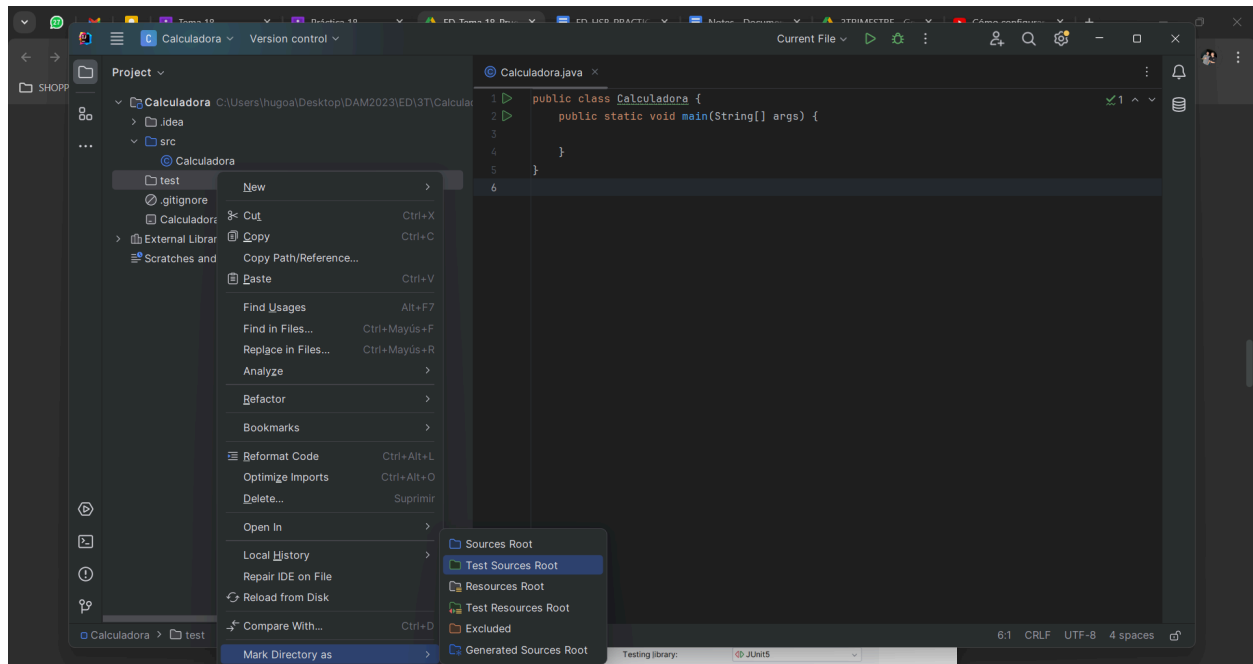
El segundo paso será crear la clase calculadora y rellenarla.



Una vez creada la clase y rellenada, crearemos un directorio donde haremos los tests.

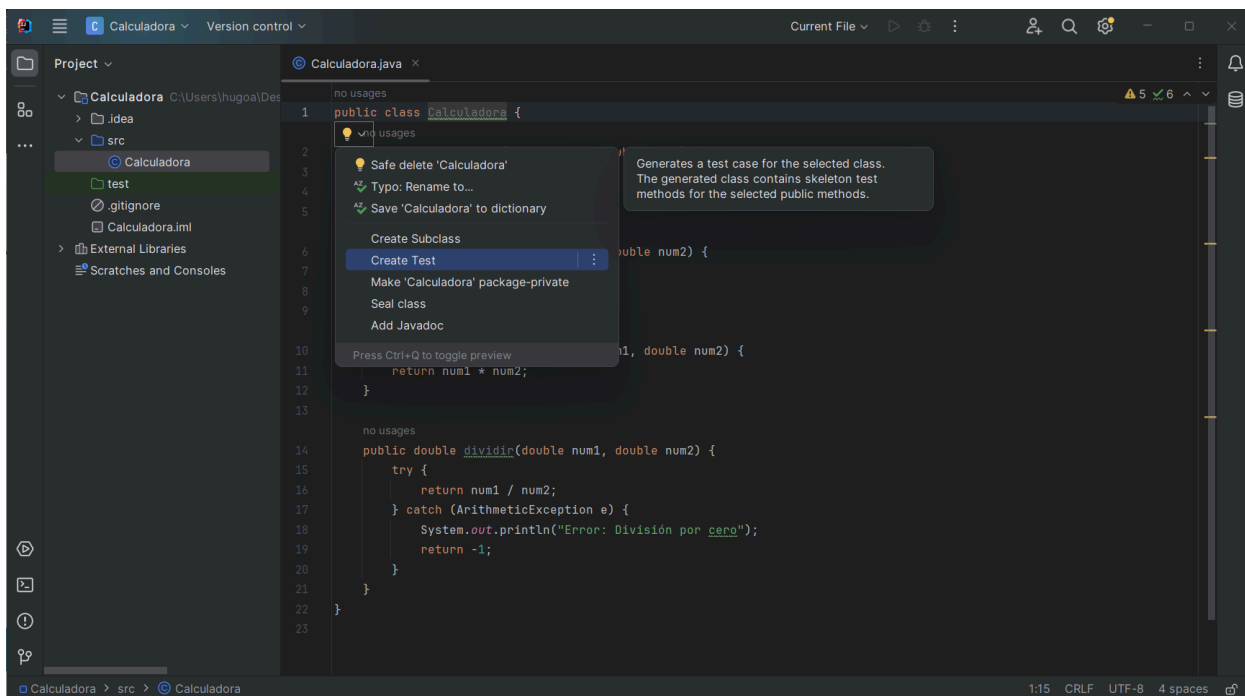


Le indicamos que queremos que sea un directorio para los tests.

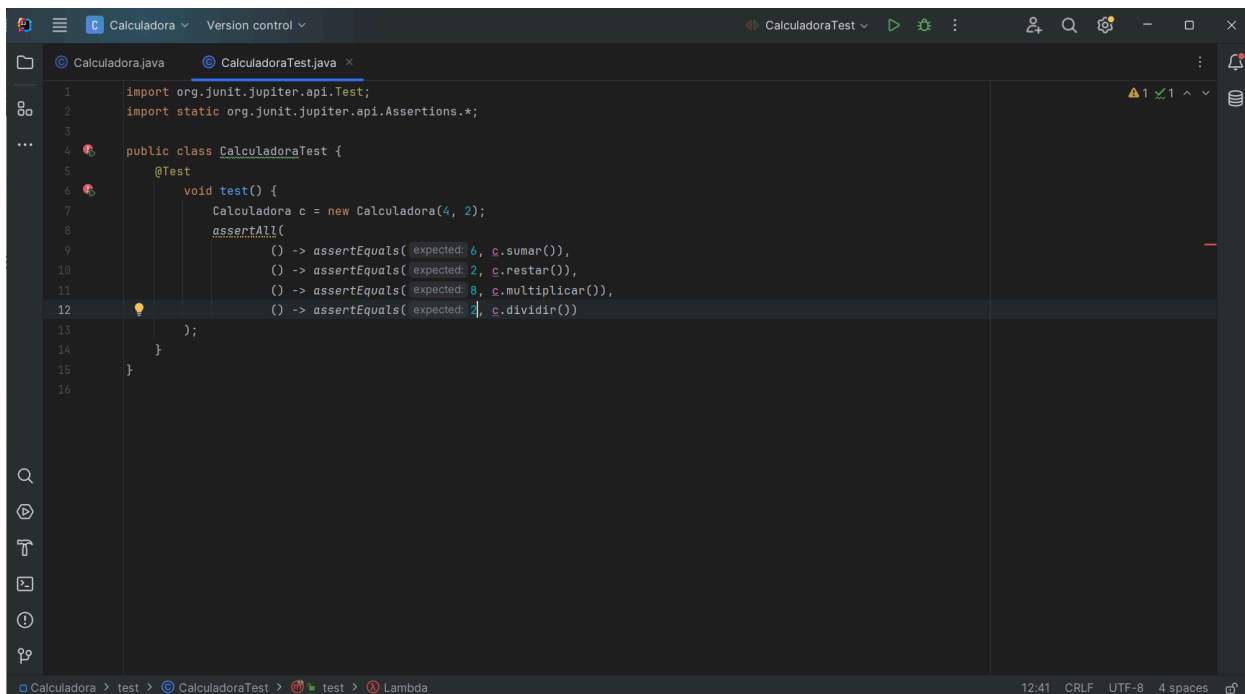


EJERCICIO 2.

Creemos una clase test de la clase Calculadora.



Una vez creada deberemos rellenarla con los tests de las funciones correspondientes de la clase Calculadora.

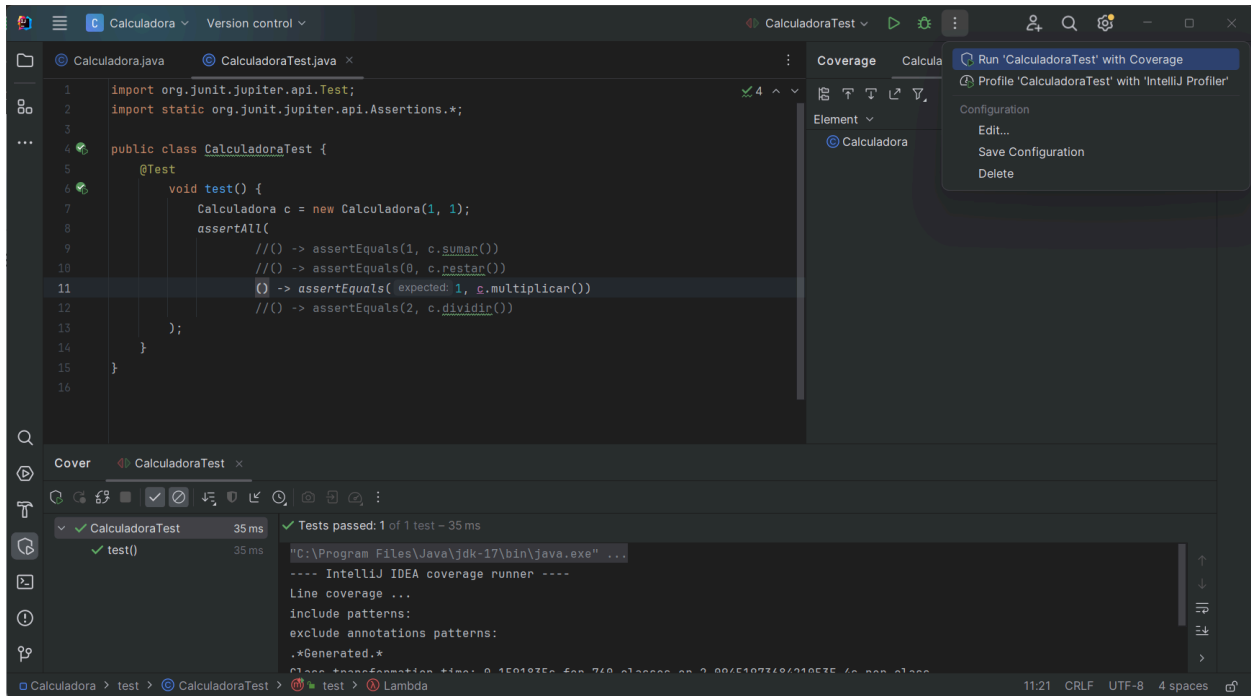


```
1  import org.junit.jupiter.api.Test;
2  import static org.junit.jupiter.api.Assertions.*;
3
4  public class CalculadoraTest {
5      @Test
6      void test() {
7          Calculadora c = new Calculadora(4, 2);
8          assertEquals(6, c.sumar());
9          assertEquals(2, c.restar());
10         assertEquals(8, c.multiplicar());
11         assertEquals(2, c.dividir());
12     }
13 }
14
15
16
```

Calculadora > test > CalculadoraTest > test > Lambda 12:41 CRLF UTF-8 4 spaces

EJERCICIO 3.

Es importante que a la hora de hacer pruebas de nuestro código, ejecutemos el código de la siguiente manera:



Pruebas método sumar

Pruebas correctas:

The screenshot shows the IntelliJ IDEA interface with the `CalculadoraTest.java` file open. The test method `test()` is highlighted, and the `assertEquals` assertion is visible. The `assertEquals` method is called with the expected value `6` and the actual result of `c.sumar()`. The test is successful, and the coverage report shows 100% class coverage and 40% method coverage.

```
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 public class CalculadoraTest {
5     @Test
6     void test() {
7         Calculadora c = new Calculadora(4, 2);
8         assertEquals(6, c.sumar());
9         //() -> assertEquals(2, c.restar());
10        //() -> assertEquals(8, c.multiplicar());
11        //() -> assertEquals(2, c.dividir());
12    }
13 }
14
15
16
```

Coverage: CalculadoraTest

Element	Class, %	Method, %	Line, %
Calculadora	100% (1/1)	40% (2/5)	40% (4/10)

Tests passed: 1 of 1 test - 37 ms

IntelliJ IDEA coverage runner

Line coverage ...

include patterns:

exclude annotations patterns:

.*Generated.*

The screenshot shows the IntelliJ IDEA interface with the `CalculadoraTest.java` file open. The test method `test()` is highlighted, and the `assertEquals` assertion is visible. The `assertEquals` method is called with the expected value `3` and the actual result of `c.sumar()`. The test is successful, and the coverage report shows 100% class coverage and 40% method coverage.

```
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 public class CalculadoraTest {
5     @Test
6     void test() {
7         Calculadora c = new Calculadora(1, 2);
8         assertEquals(3, c.sumar());
9         //() -> assertEquals(6, c.restar());
10        //() -> assertEquals(1, c.multiplicar());
11        //() -> assertEquals(2, c.dividir());
12    }
13 }
14
15
16
```

Coverage: CalculadoraTest

Element	Class, %	Method, %	Line, %
Calculadora	100% (1/1)	40% (2/5)	40% (4/10)

Tests passed: 1 of 1 test - 36 ms

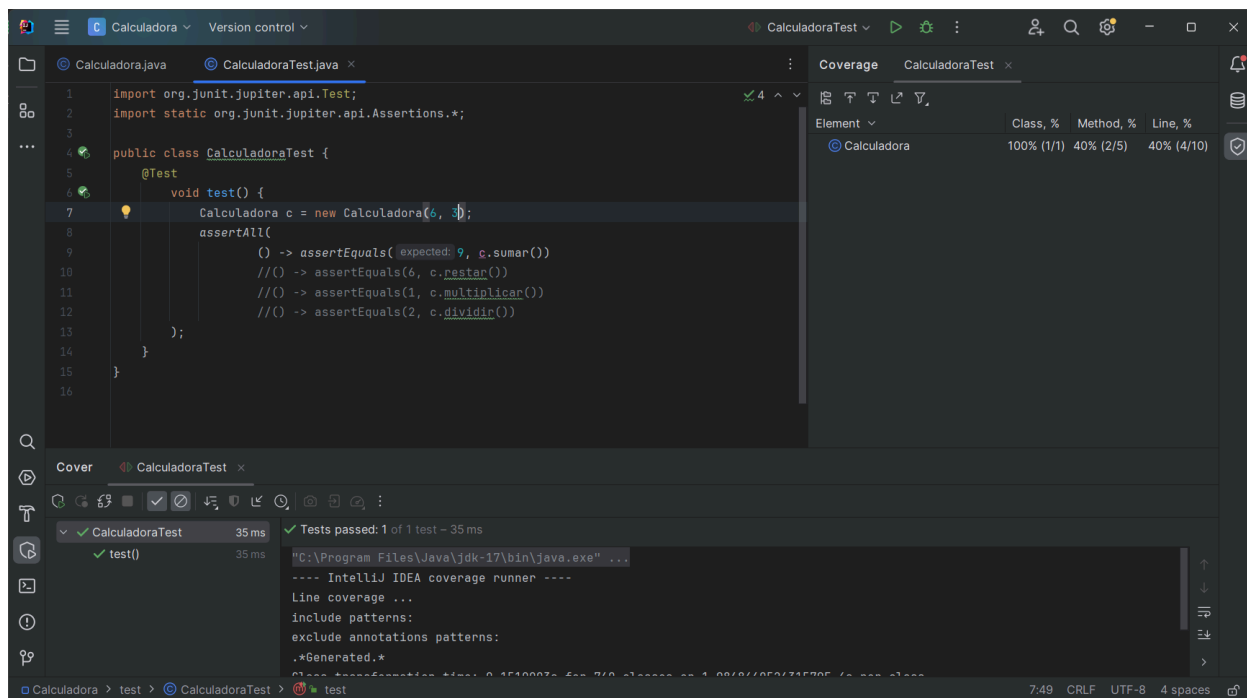
IntelliJ IDEA coverage runner

Line coverage ...

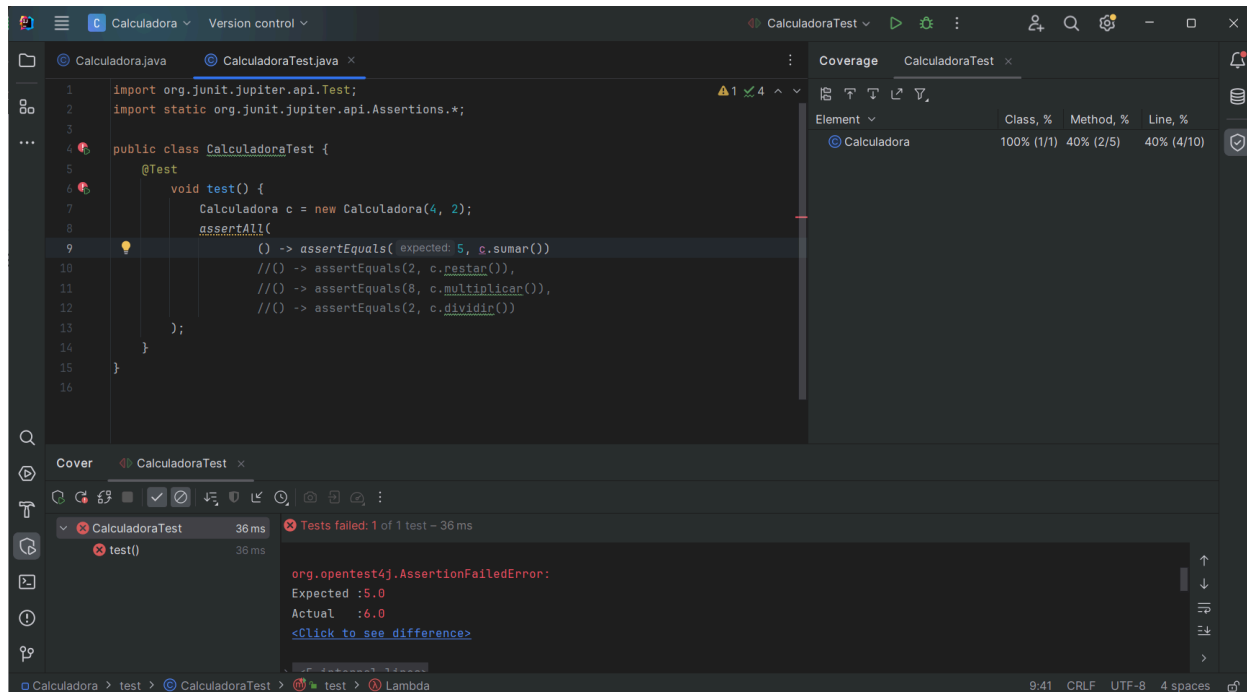
include patterns:

exclude annotations patterns:

.*Generated.*



Pruebas incorrectas:



The screenshot shows an IDE window with the following components:

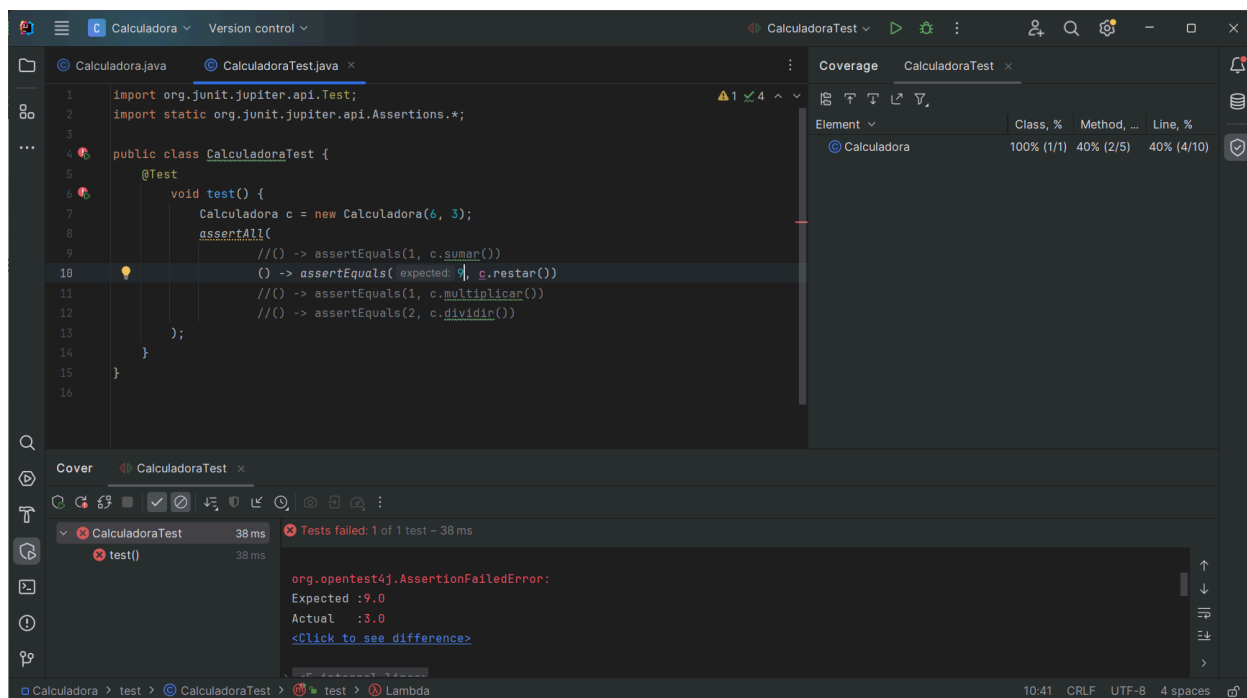
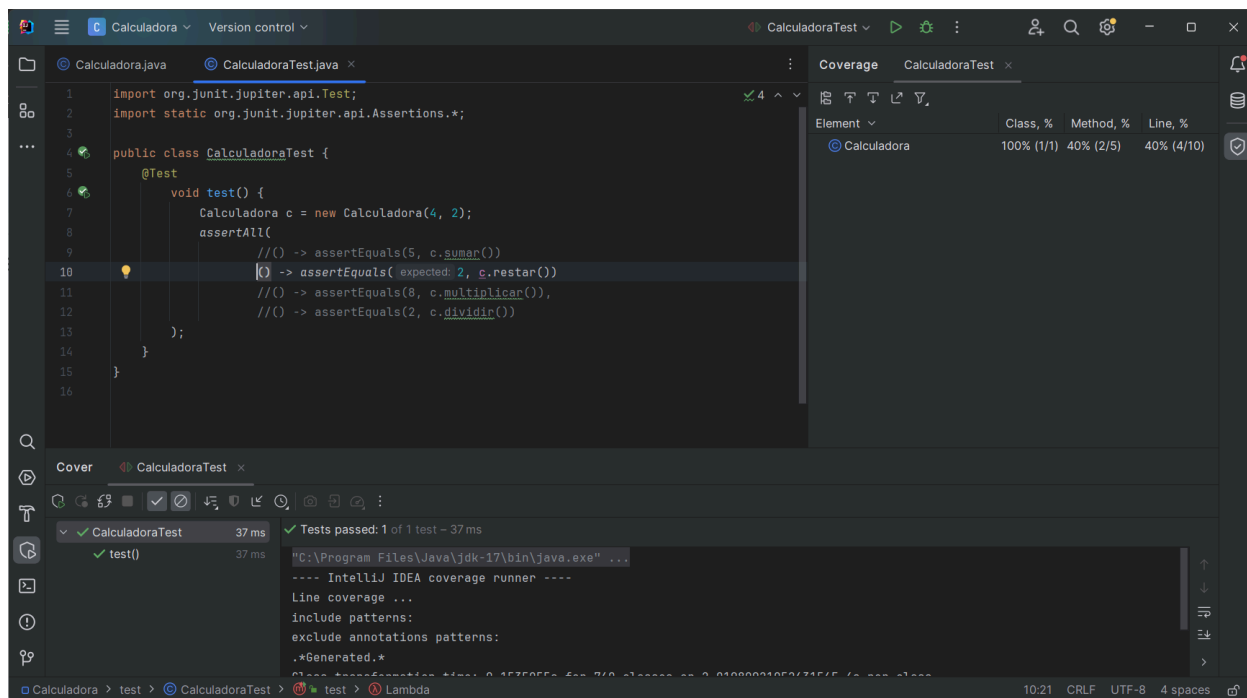
- Editor:** Displays `CalculadoraTest.java`. The code is as follows:

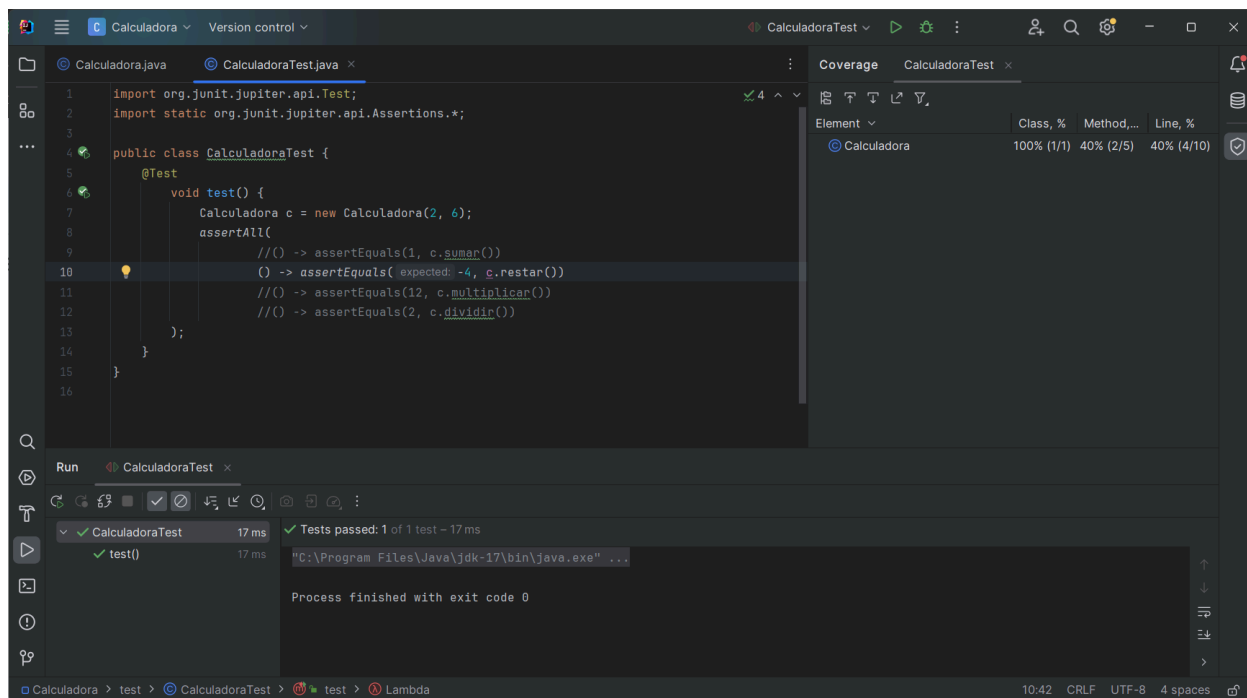
```
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 public class CalculadoraTest {
5     @Test
6     void test() {
7         Calculadora c = new Calculadora(6, 3);
8         assertEquals(
9             () -> assertEquals(expected: 1, c.sumar())
10            //() -> assertEquals(6, c.restar())
11            //() -> assertEquals(1, c.multiplicar())
12            //() -> assertEquals(2, c.dividir())
13        );
14    }
15 }
16
```
- Coverage:** A table showing coverage for `Calculadora`.

Element	Class, %	Method, %	Line, %
Calculadora	100% (1/1)	40% (2/5)	40% (4/10)
- Test Results:** Shows a failed test.
 - Test: `CalculadoraTest` (39 ms)
 - Method: `test()` (39 ms)
 - Failure message: `org.opentest4j.AssertionFailedError: Expected :1.0 Actual :9.0 <Click to see difference>`
- Footer:** `Calculadora > test > CalculadoraTest > test > Lambda`, 9:53, CRLF, UTF-8, 4 spaces.

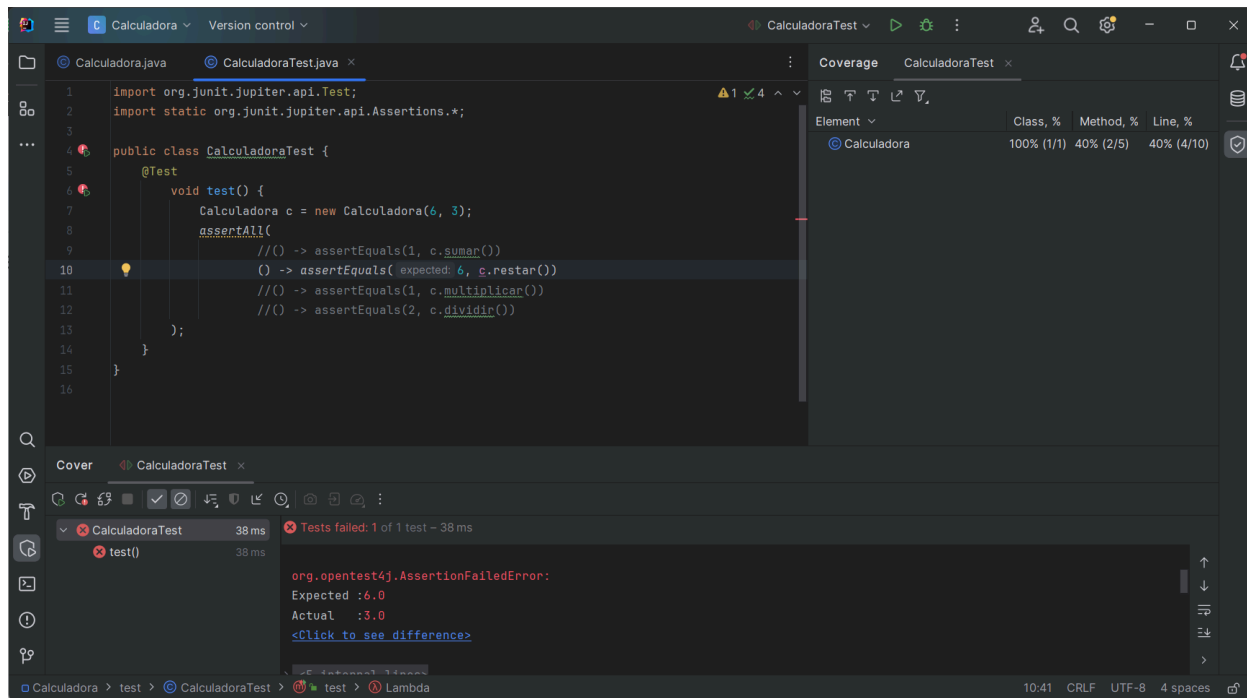
Pruebas método restar

Pruebas correctas:





Pruebas incorrectas:



The screenshot shows an IDE with the following components:

- Editor:** Displays `CalculadoraTest.java` with the following code:

```
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 public class CalculadoraTest {
5     @Test
6     void test() {
7         Calculadora c = new Calculadora(4, 2);
8         assertEquals(5, c.sumar());
9         //() -> assertEquals(5, c.sumar())
10        () -> assertEquals( expected: 6, c.restar());
11        //() -> assertEquals(8, c.multiplicar());
12        //() -> assertEquals(2, c.dividir());
13    }
14 }
15
16
```

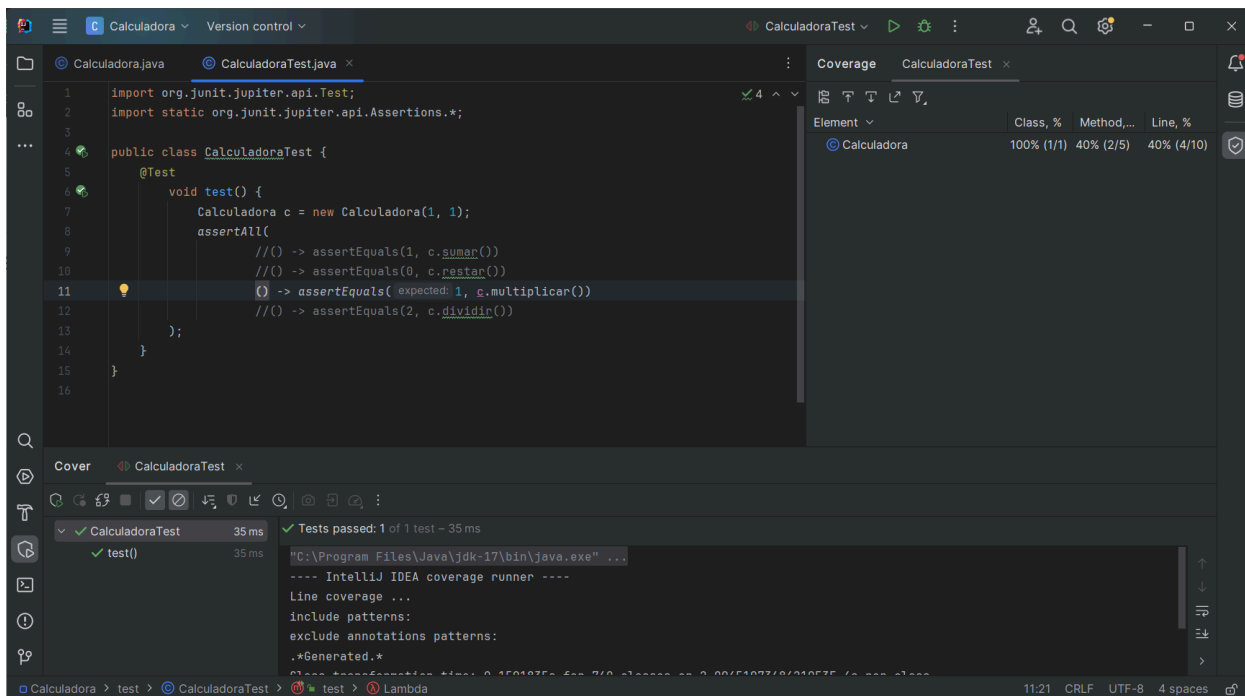
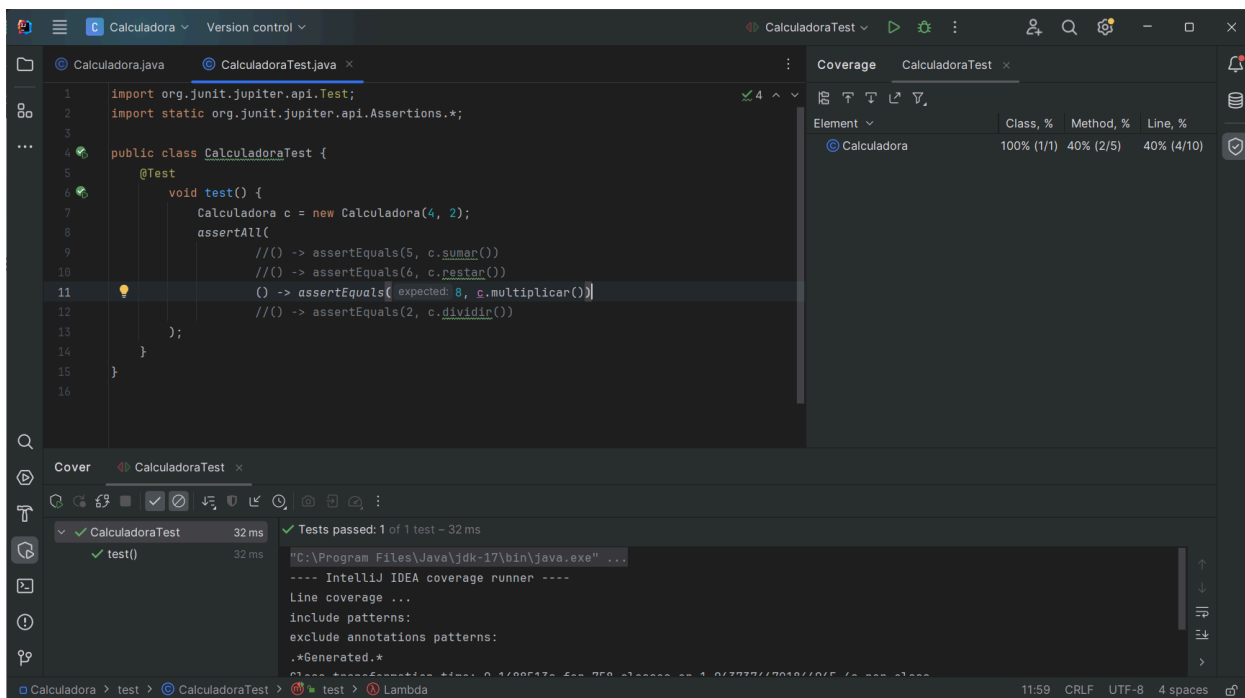
A yellow lightbulb icon is next to line 10, indicating a suggestion or warning.
- Coverage:** A table showing coverage for `Calculadora`:

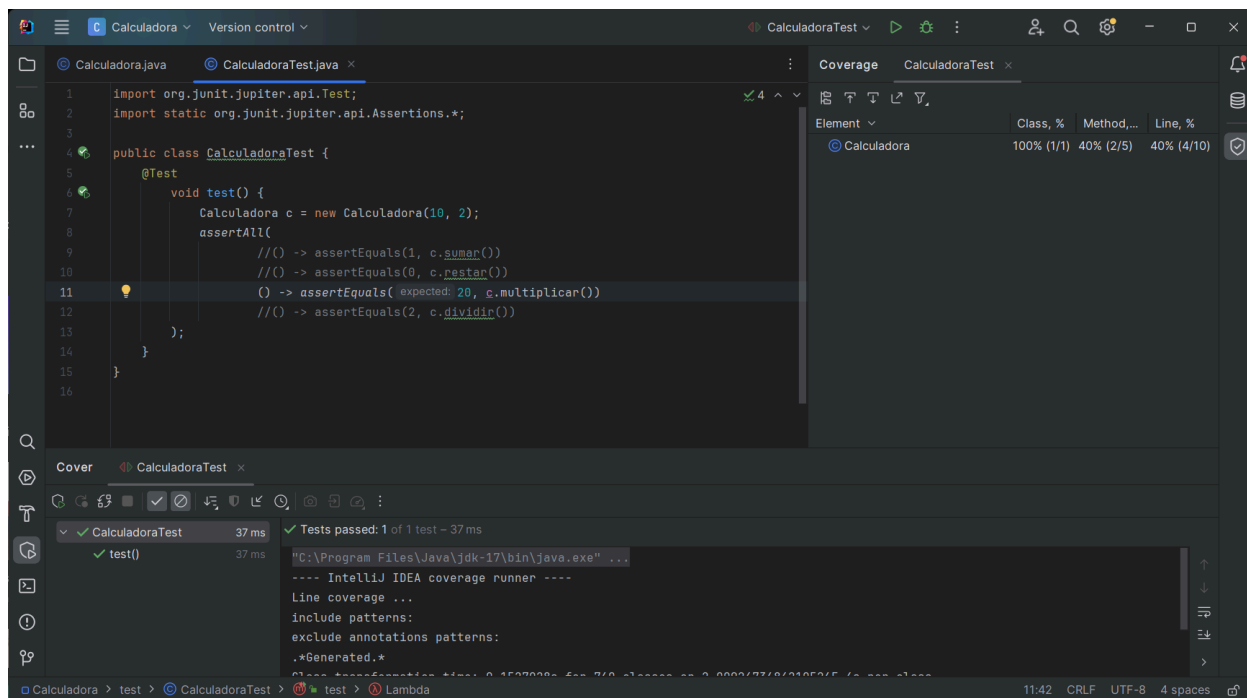
Element	Class, %	Method, %	Line, %
Calculadora	100% (1/1)	40% (2/5)	40% (4/10)
- Test Results:** Shows a failure for `CalculadoraTest.test()`. The error message is:

```
org.opentest4j.AssertionFailedError:
Expected :6.0
Actual   :2.0
<Click to see difference>
```

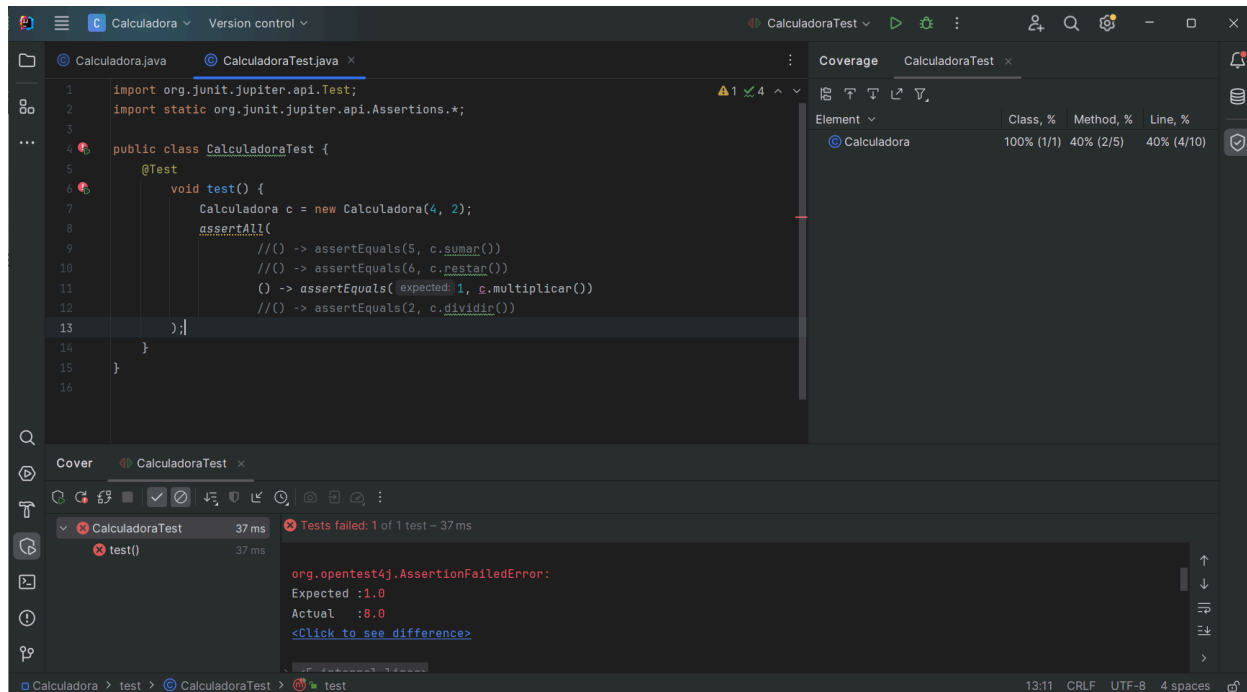
Prueba método multiplicar

Pruebas correctas:





Prueba incorrecta:



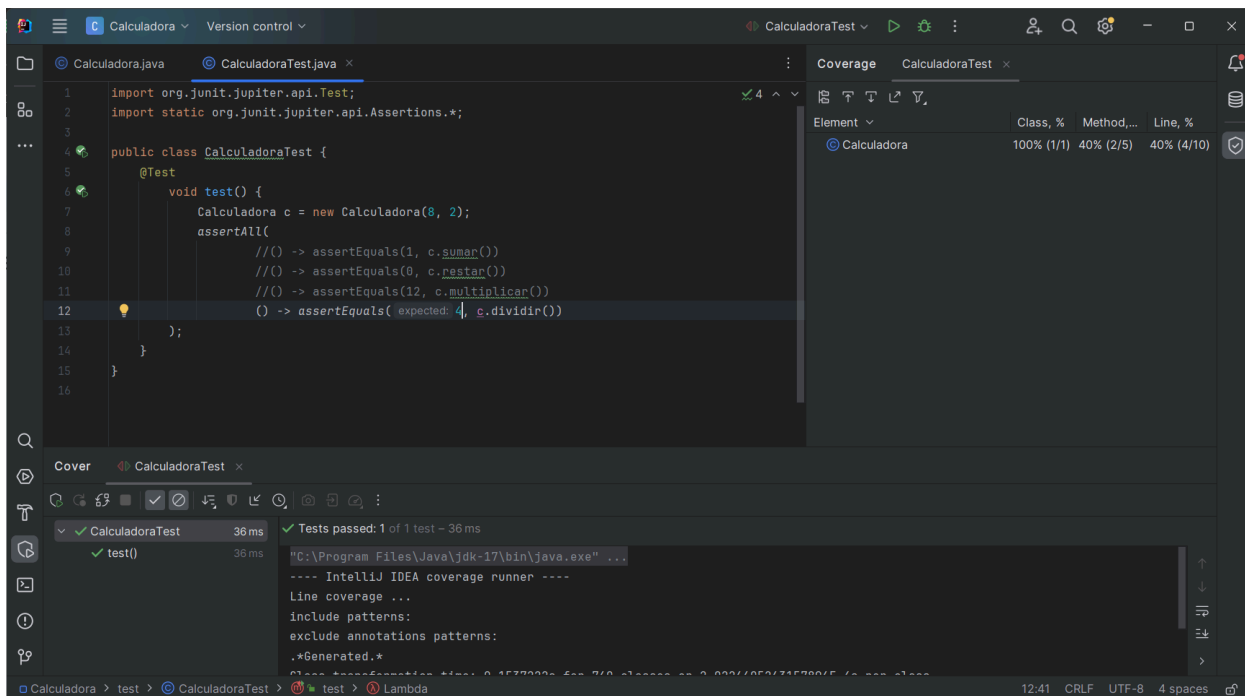
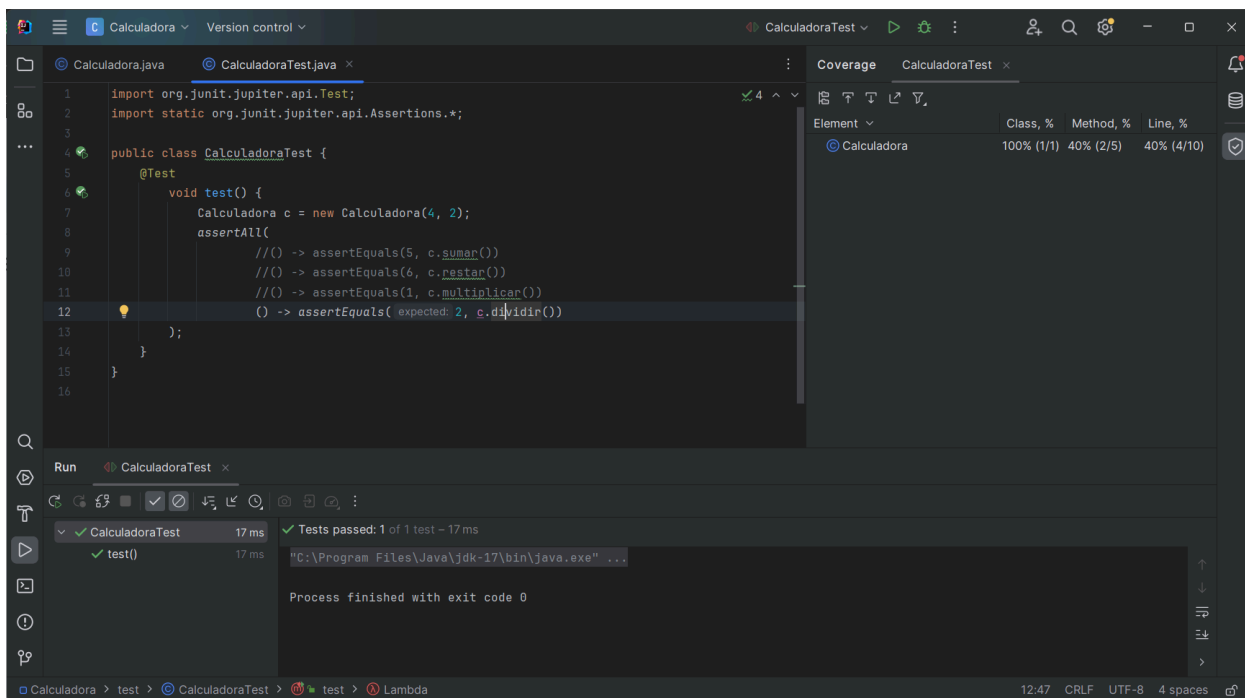
The screenshot shows an IDE with the following components:

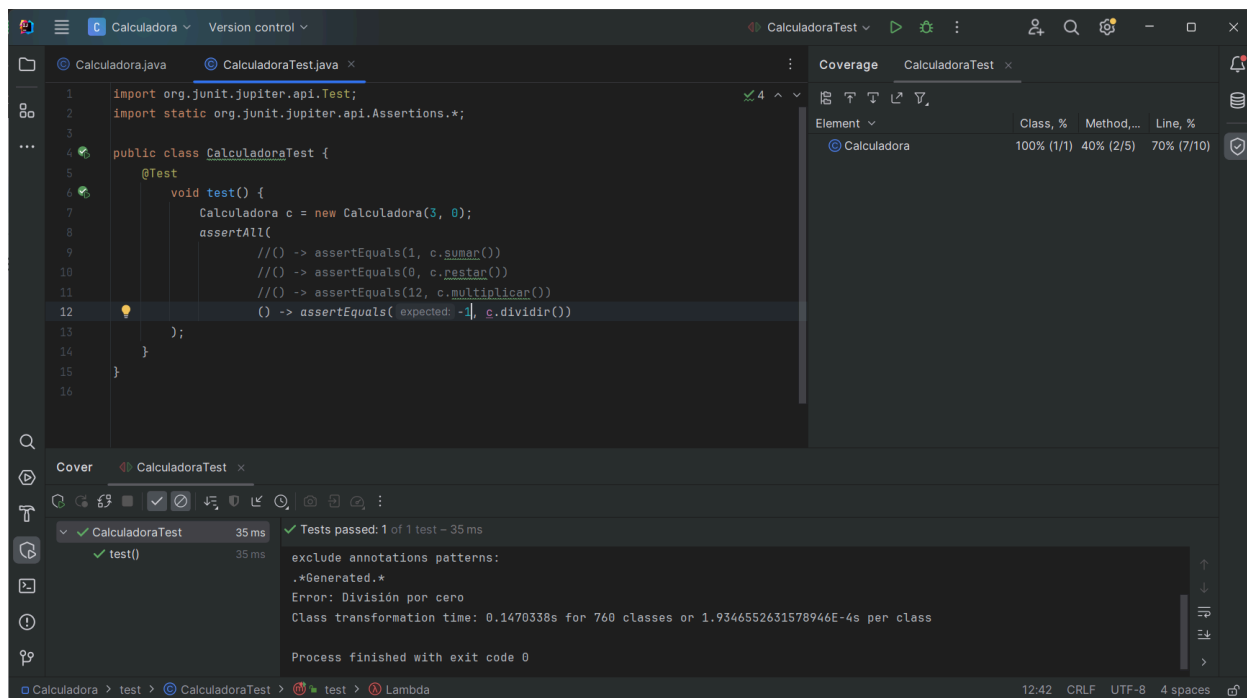
- Editor:** Displays `CalculadoraTest.java`. The code includes imports for JUnit and Assertions, and a `test()` method. Line 11 is highlighted with a lightbulb icon, indicating an issue: `() -> assertEquals(expected: 12, c.multiplicar())`.
- Coverage:** A table showing coverage for `Calculadora`.

Element	Class, %	Method, %	Line, %
Calculadora	100% (1/1)	40% (2/5)	40% (4/10)
- Run and Debug:** Shows a failed test. The test `test()` took 45 ms and failed. The error message is: `org.opentest4j.AssertionFailedError: Expected :12.0 Actual :16.0`. A link to see the difference is provided.

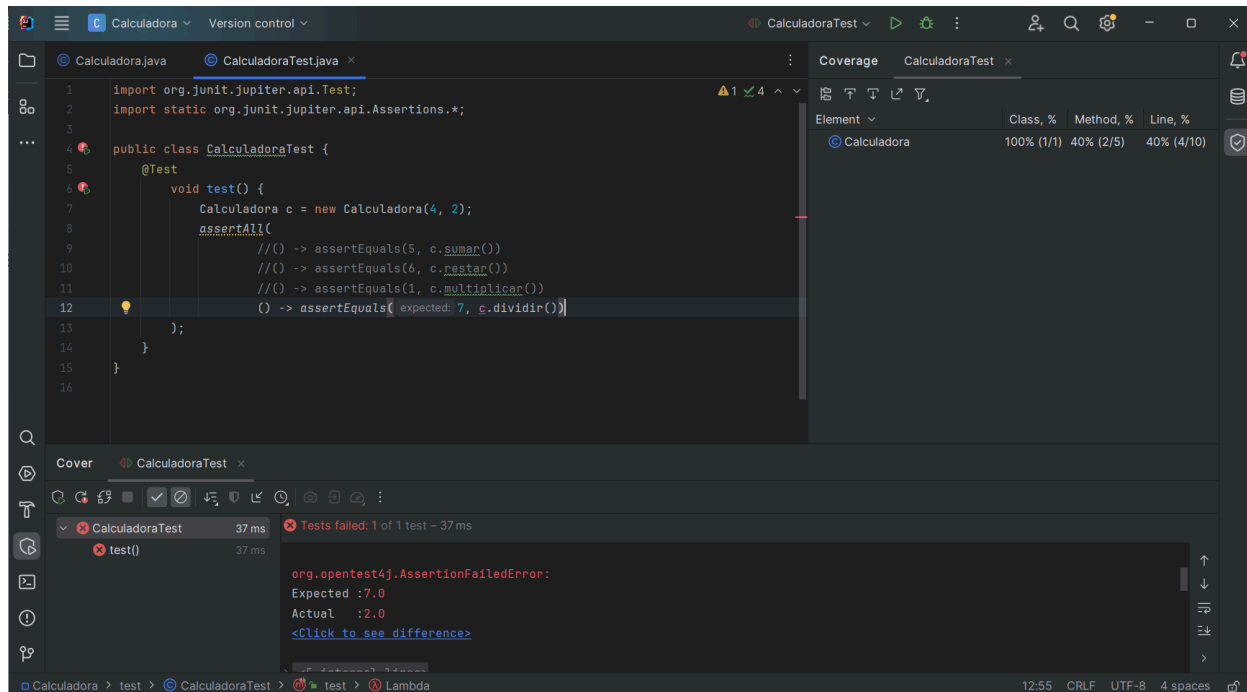
Prueba método dividir

Prueba correcta:





Prueba incorrecta:



The screenshot shows an IDE window with the following components:

- Editor:** Displays `CalculadoraTest.java`. The code includes imports for JUnit and AssertJ, and a `test()` method that creates a `Calculadora` instance and performs several assertions. A red squiggly line indicates a failure on the `assertEquals(2, c.dividir())` line.
- Coverage:** A table showing coverage for `Calculadora`.

Element	Class, %	Method, %	Line, %
Calculadora	100% (1/1)	40% (2/5)	70% (7/10)
- Run and Debug:** Shows the test execution results. The test `test()` failed with an `org.opentest4j.AssertionFailedError`. The error message is: `Expected :2.0 Actual :-1.0`. A link to see the difference is provided.
- Bottom Bar:** Shows the file path `Calculadora > test > CalculadoraTest > test` and the status bar with `7:46 CRLF UTF-8 4 spaces`.