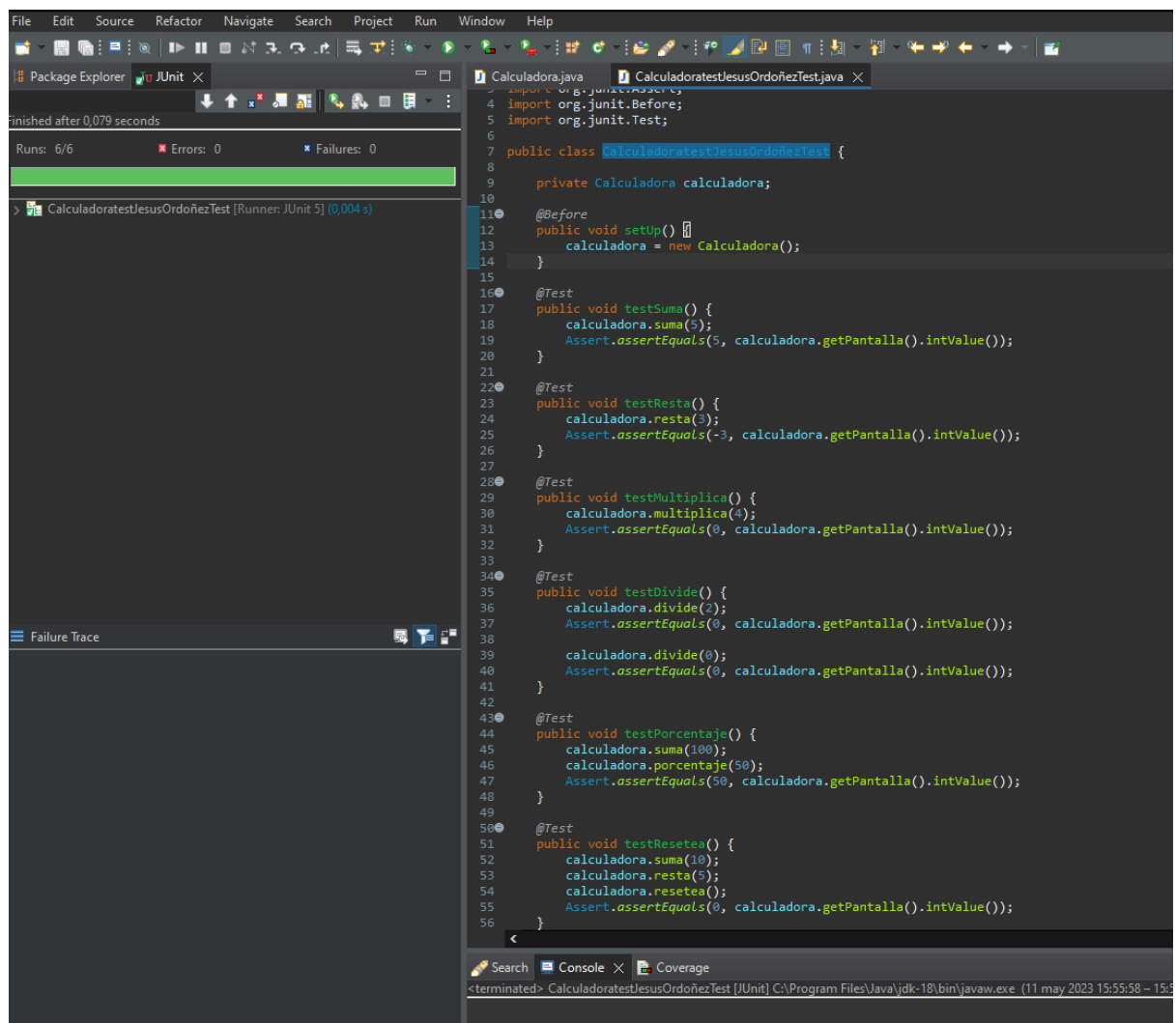


En este ejemplo, se utiliza la anotación `@Before` para crear una instancia de `Calculadora` antes de cada prueba, lo que asegura que cada prueba comienza con una calculadora "limpia". Luego, se definen diferentes métodos de prueba utilizando la anotación `@Test` para cada método de la clase `Calculadora`.

Dentro de cada método de prueba, se llaman a los métodos correspondientes de la calculadora y se utilizan los métodos de la clase `Assert` para verificar si los resultados son los esperados. Por ejemplo, se utiliza `Assert.assertEquals()` para comparar el valor de la pantalla después de llamar al método `suma()`.



The screenshot shows an IDE with the following components:

- Package Explorer:** Shows the project structure with a folder named `JUnit`.
- JUnit Test Results:** A green bar indicates that all tests passed. The summary shows "Runs: 6/6", "Errors: 0", and "Failures: 0". The test class is `CalculadoratestJesusOrdoñezTest` and it took 0.004 seconds to run.
- Source Editor:** Displays the code for `CalculadoratestJesusOrdoñezTest.java`. The code includes imports for `org.junit.Before` and `org.junit.Test`. It defines a class `CalculadoratestJesusOrdoñezTest` with a private `Calculadora` instance. The `@Before` method `setUp()` initializes the `calculadora` object. There are six `@Test` methods: `testSuma()`, `testResta()`, `testMultiplica()`, `testDivide()`, `testPorcentaje()`, and `testResetea()`. Each test method calls a corresponding method on the `calculadora` instance and uses `Assert.assertEquals()` to verify the result.
- Console:** Shows the command prompt output, indicating that the tests terminated successfully.

```
import org.junit.Before;
import org.junit.Test;

public class CalculadoratestJesusOrdoñezTest {
    private Calculadora calculadora;

    @Before
    public void setUp() {
        calculadora = new Calculadora();
    }

    @Test
    public void testSuma() {
        calculadora.suma(5);
        Assert.assertEquals(5, calculadora.getPantalla().intValue());
    }

    @Test
    public void testResta() {
        calculadora.resta(3);
        Assert.assertEquals(-3, calculadora.getPantalla().intValue());
    }

    @Test
    public void testMultiplica() {
        calculadora.multiplica(4);
        Assert.assertEquals(0, calculadora.getPantalla().intValue());
    }

    @Test
    public void testDivide() {
        calculadora.divide(2);
        Assert.assertEquals(0, calculadora.getPantalla().intValue());

        calculadora.divide(0);
        Assert.assertEquals(0, calculadora.getPantalla().intValue());
    }

    @Test
    public void testPorcentaje() {
        calculadora.suma(100);
        calculadora.porcentaje(50);
        Assert.assertEquals(50, calculadora.getPantalla().intValue());
    }

    @Test
    public void testResetea() {
        calculadora.suma(10);
        calculadora.resta(5);
        calculadora.resetea();
        Assert.assertEquals(0, calculadora.getPantalla().intValue());
    }
}
```

Ahora explico como se utilizan los métodos y para qué sirven:

- Método suma(Integer número): Este método toma un número como parámetro y lo suma al número actual en la pantalla de la calculadora. Es decir, incrementa el valor de la pantalla por el valor del número proporcionado.
- Método resta(Integer número): Este método toma un número como parámetro y lo resta al número actual en la pantalla de la calculadora. Es decir, disminuye el valor de la pantalla por el valor del número proporcionado.
- Método multiplica(Integer número): Este método toma un número como parámetro y multiplica el número actual en la pantalla de la calculadora por el valor del número proporcionado. El resultado se guarda en la pantalla.
- Método divide(Integer número): Este método toma un número como parámetro y divide el número actual en la pantalla de la calculadora por el valor del número proporcionado, siempre y cuando el número proporcionado sea diferente de cero. Si el número proporcionado es cero, no se realiza la división y el valor en la pantalla se mantiene sin cambios.
- Método porcentaje(Integer número): Este método toma un número como parámetro y cambia el número en la pantalla de la calculadora al porcentaje del valor indicado. Es decir, multiplica el número actual en la pantalla por el número proporcionado y luego divide el resultado entre 100. El resultado se guarda en la pantalla.
- Método resetea(): Este método establece el número en la pantalla de la calculadora a cero, reiniciando así el valor actual.
- Método getPantalla(): Este método devuelve el número actual en la pantalla de la calculadora.