

Test de clases Java

JUNIT

Blas Barragán Román



JUnit

IMPLEMENTAR CLASE JAVA

Implementamos una clase en java con sus funciones

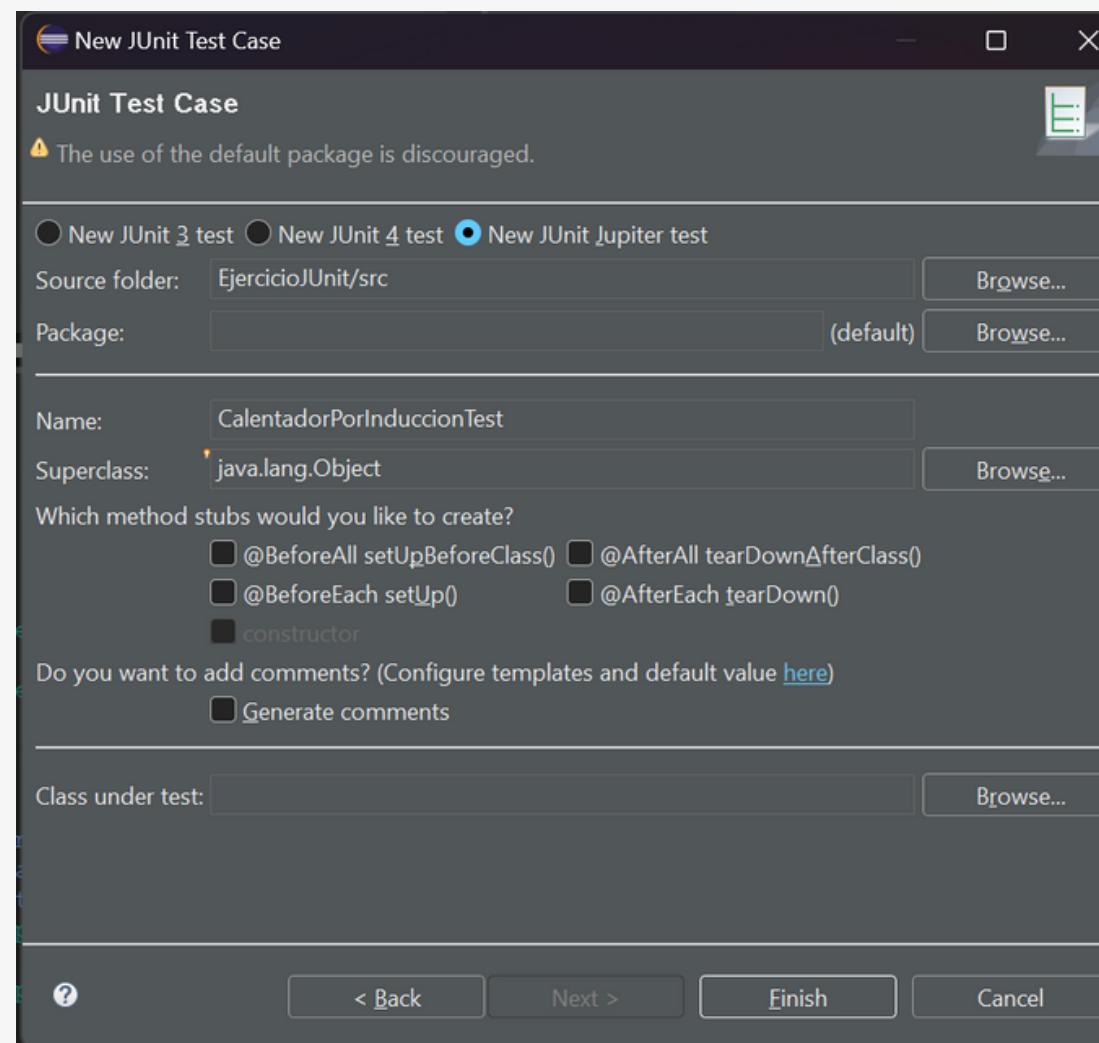
```
*CalentadorPorInduccion.java x
1 public class CalentadorPorInduccion {
2     private boolean encendido;
3     private int temperatura;
4
5     public CalentadorPorInduccion() {
6         this.encendido = false;
7
8         this.temperatura = 0;
9     }
10
11    public boolean encender() {
12        if (!encendido) {
13            encendido = true;
14            System.out.println("El calentador por inducción está encendido.");
15        } else {
16            System.out.println("El calentador por inducción ya está encendido.");
17        }
18        return encendido;
19    }
20
21    public boolean apagar() {
22        if (encendido) {
23            encendido = false;
24            temperatura = 0;
25            System.out.println("El calentador por inducción está apagado.");
26        } else {
27            System.out.println("El calentador por inducción ya está apagado.");
28        }
29        return encendido;
30    }
31
32    public int ajustarTemperatura(int nuevaTemperatura) {
33        if (temperatura != nuevaTemperatura) {
34            temperatura = nuevaTemperatura;
35            System.out.println("La temperatura se ha ajustado a " + temperatura + " grados Celsius.");
36        } else {
37            System.out.println("No se puede ajustar la temperatura porque el calentador está apagado.");
38        }
39        return temperatura;
40    }
41}
```



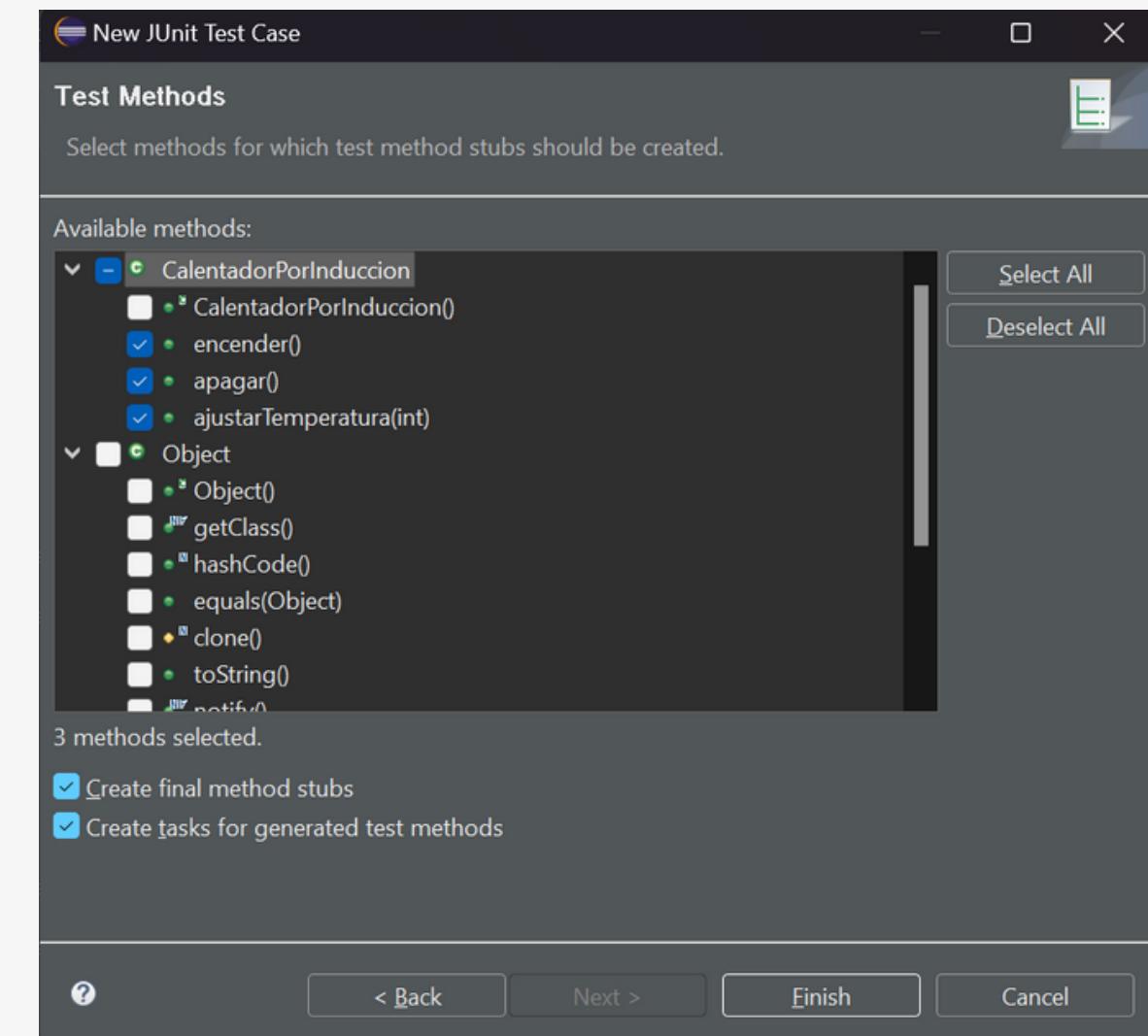
JUnit

CREACION CLASE TEST

Creamos una nueva clase JUnit Test



Seleccionamos los metodos de nuestra clase



IMPLEMENTAR FUNCIONES TEST

Implementamos el comportamiento de las distintas funciones de test creadas sin olvidar los distintos métodos "Assert" según nuestras necesidades.

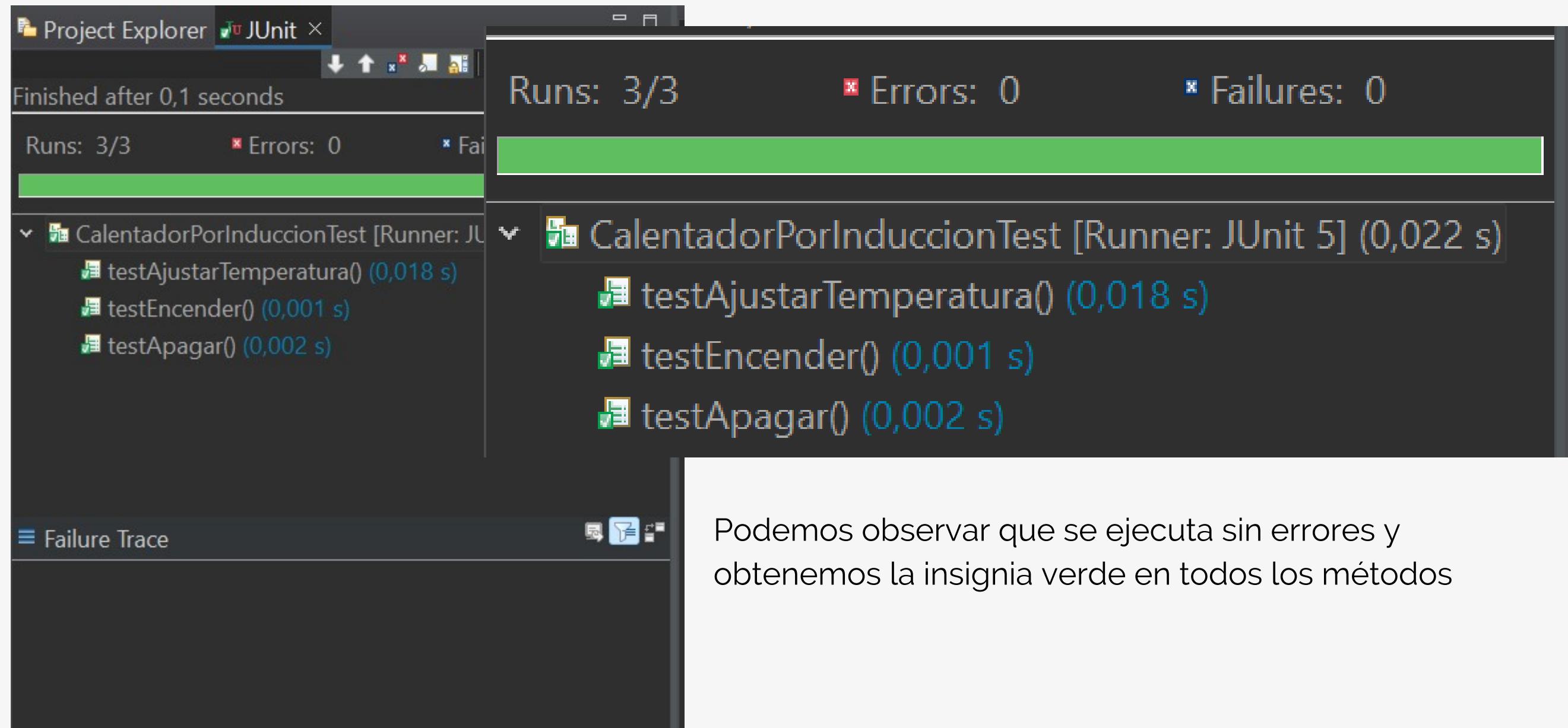


```
CalentadorPorInduccionTest.java
1
2 import static org.junit.jupiter.api.Assertions.*;
3
4
5 class CalentadorPorInduccionTest {
6
7     @Test
8     final void testEncender() {
9         boolean valorEsperado = true;
10        CalentadorPorInduccion calent = new CalentadorPorInduccion();
11        boolean resultado = calent.encender();
12        assertTrue(resultado);
13    }
14
15
16    @Test
17    final void testApagar() {
18        boolean valorEsperado = false;
19        CalentadorPorInduccion calent = new CalentadorPorInduccion();
20        boolean resultado = calent.apagar();
21        assertFalse(resultado);
22    }
23
24    @Test
25    final void testAjustarTemperatura() {
26        int tempDeseada = 850;
27        CalentadorPorInduccion calent = new CalentadorPorInduccion();
28        int tempAjustada = calent.ajustarTemperatura(850);
29        assertEquals(tempDeseada, tempAjustada);
30    }
31
32
33 }
```

JUnit

EJECUTAR TEST

Ejecutamos el test y observador los resultados obtenidos.



RESULTADO

Cambiamos parametros en el codigo para forzar los errores

The image shows a Java code editor and two JUnit execution panes. The code editor displays `CalentadorPorInduccionTest.java` with three test methods: `testEncender()`, `testApagar()`, and `testAjustarTemperatura()`. The `testApagar()` and `testAjustarTemperatura()` methods contain assertions that will fail due to parameter changes. The top JUnit pane shows the results of the first run: 3 runs, 0 errors, and 2 failures. The bottom JUnit pane shows the results of a subsequent run where all tests have passed (0 errors, 0 failures). The failure traces in both panes provide detailed information about the assertion failures.

```
*CalentadorPorInduccionTest.java *
1
2 import static org.junit.jupiter.api.Assertions.*;
3
4 class CalentadorPorInduccionTest {
5
6     @Test
7     final void testEncender() {
8         boolean valorEsperado = true;
9         CalentadorPorInduccion calent = new CalentadorPorInduccion();
10        boolean resultado = calent.apagar();
11        assertTrue(resultado);
12    }
13
14    @Test
15    final void testApagar() {
16        boolean valorEsperado = false;
17        CalentadorPorInduccion calent = new CalentadorPorInduccion();
18        boolean resultado = calent.apagar();
19        assertFalse(resultado);
20    }
21
22    @Test
23    final void testAjustarTemperatura() {
24        int tempDeseada = 850;
25        CalentadorPorInduccion calent = new CalentadorPorInduccion();
26        int tempAjustada = calent.ajustarTemperatura(85);
27        assertEquals(tempDeseada, tempAjustada);
28    }
29
30
31
32
33 }
```

Project Explorer JUnit X
Finished after 0,105 seconds
Runs: 3/3 Errors: 0 Failures: 2

CalentadorPorInduccionTest [Runner: JUnit 5] (0,023 s)
testEncender() (0,002 s)
testApagar() (0,001 s)

Failure Trace
! org.opentest4j.AssertionFailedError: expected: <850> but was: <85>
at org.junit.jupiter.api.AssertionFailureBuilder.build(AssertionFailureBuilder.java:151)
at CalentadorPorInduccionTest.testAjustarTemperatura(CalentadorPorInduccionTest.java:29)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

Project Explorer JUnit X
Finished after 0,105 seconds
Runs: 3/3 Errors: 0 Failures: 2

CalentadorPorInduccionTest [Runner: JUnit 5] (0,023 s)
testEncender() (0,002 s)
testApagar() (0,001 s)

Failure Trace
! org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
at org.junit.jupiter.api.AssertionFailureBuilder.build(AssertionFailureBuilder.java:151)
at CalentadorPorInduccionTest.testEncender(CalentadorPorInduccionTest.java:13)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)
at java.base/java.util.ArrayList.forEach(ArrayList.java:1511)

Observamos las distintas causas que han generado los fallos de ejecucion.