

Método factorial del proyecto_factorial

Análisis de Caja blanca

Grafo	Nodo	Línea (Condición o Detalle)
	1	Línea 4 (Inicio)
	2	Línea 5 ($n < 0$)
	3	Línea 6 (Exception)
	4	Línea 8
	5	Línea 9 ($i = 2$)
	6	Línea 9 ($i \leq n$)
	7	Línea 10
	8	Línea 9 ($i++$)
	9	Línea 12
	10	Línea 13 (Final)

Método de cálculo	Complejidad	Comentarios
Número de regiones	3	
$n.aristas - n.nodos + 2$	$11 - 10 + 2 = 3$	
$n.condiciones + 1$	$2 + 1 = 3$	Nodos 2 y 6

Camino	Nodos
CAM1	1 - 2 - 3 - 10
CAM2	1 - 2 - 4 - 5 - 6 - 9 - 10
CAM3	1 - 2 - 4 - 5 - 6 - 7 - 8 - 6 - 9 - 10

Análisis de Caja negra

Tipo de dato	Descripción	Clases equivalencia	Valor límite
Rango de valores de entrada	El número debe estar entre 0 y 127	Clase válida Valor entre 0 y 127	ALV1: 0 ALV2: 127
		Clases no válidas Número negativo Mayor de 127	ALV3: -1 ALV4: 128

Conjetura de errores:

- CE1: Comprobar los casos especiales del cálculo del factorial de 0 y 1, ya que siguen un criterio distinto al de resto de números enteros
- CE2: Comprobar que se haga correctamente el control de errores para el caso en el que se introduzca algo que no sea un número entero (una letra o número decimal).

ID Caso de prueba	Entrada	ID de Clase / Camino / ALV / CE	Salida
C1	0	CAM2 / ALV1 / CE1	Factorial(0) = 1,000000
C2	1	CAM2 / CE1	Factorial(1) = 1,000000
C3	127	ALV2 / CAM3	Factorial(127) = Infinity
C4	-1	CAM1 / ALV3	Error. El número tiene que ser ≥ 0
C5	128	ALV4	Error. El dato tecleado no es válido
C6	4	CAM3	Factorial(4) = 24,000000
C7	2.2	CE2	Error. El dato tecleado no es válido

```
package proyecto_factorial;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author Iván Estévez
 */
public class FactorialTest {

    Factorial factorial;

    public FactorialTest() {
    }

    @BeforeClass
    public static void setUpClass() {
        System.out.println("*** Iniciando paquete de pruebas ***");
    }

    @AfterClass
    public static void tearDownClass() {
        System.out.println("*** Finalizando paquete de pruebas ***");
    }

    @Before
    public void setUp() {
        System.out.println("*** Iniciando prueba ***");
        factorial = new Factorial();
    }

    @After
    public void tearDown() {
        System.out.println("*** Prueba finalizada ***");
    }

    // TODO add test methods here.
    // The methods must be annotated with annotation @Test. For example:
    //
    // @Test
    // public void hello() {}
}
```

```

@Test
public void testC1() throws Exception {
    // Arrange
    byte entrada = 0;
    // Act
    float resultado = factorial.factorial(entrada);
    // Assert
    assertEquals(1,resultado,0); // Añado tercer valor para evitar el
Deprecated
    }

```

```

@Test
public void testC2() throws Exception {
    byte entrada = 1;
    float resultado = factorial.factorial(entrada);
    assertEquals(1,resultado,0);
}

```

```

@Test
public void testC3() throws Exception {
    byte entrada = 127;
    float resultado = factorial.factorial(entrada);
    assertEquals(Float.POSITIVE_INFINITY,resultado,0);
}

```

```

@Test(expected = Exception.class)
public void testC4() throws Exception {
    byte entrada = -1;
    float resultado = factorial.factorial(entrada);
}

```

//C5 no se puede probar ya que en un byte no cabe '128'. Esa responsabilidad recae en 'main'

```

@Test
public void testC6() throws Exception {
    byte entrada = 4;
    float resultado = factorial.factorial(entrada);
    assertEquals(24,resultado,0);
}

```

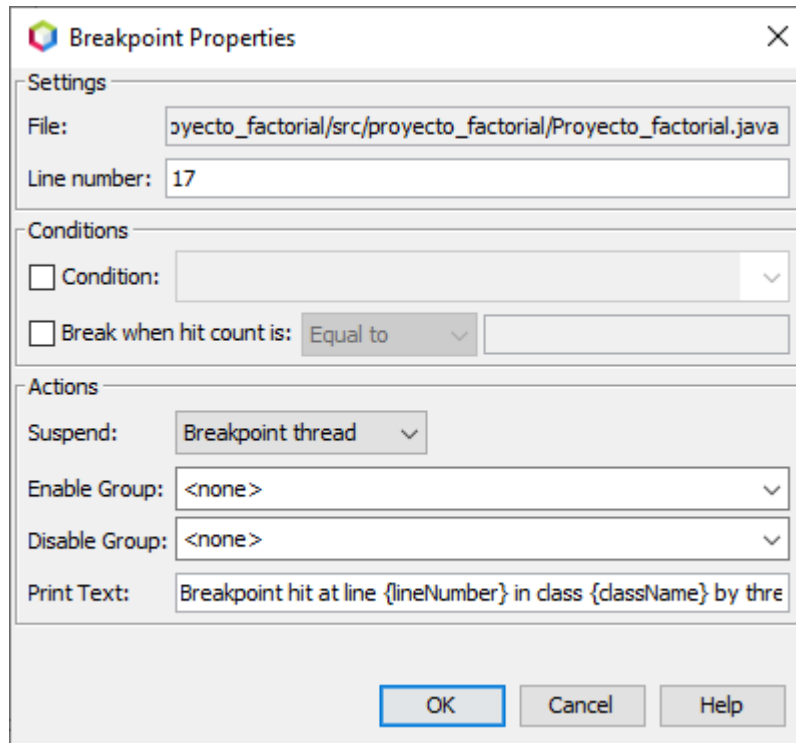
//C7 no se puede probar ya que un byte no admite decimales. Esa responsabilidad recae en 'main'

```

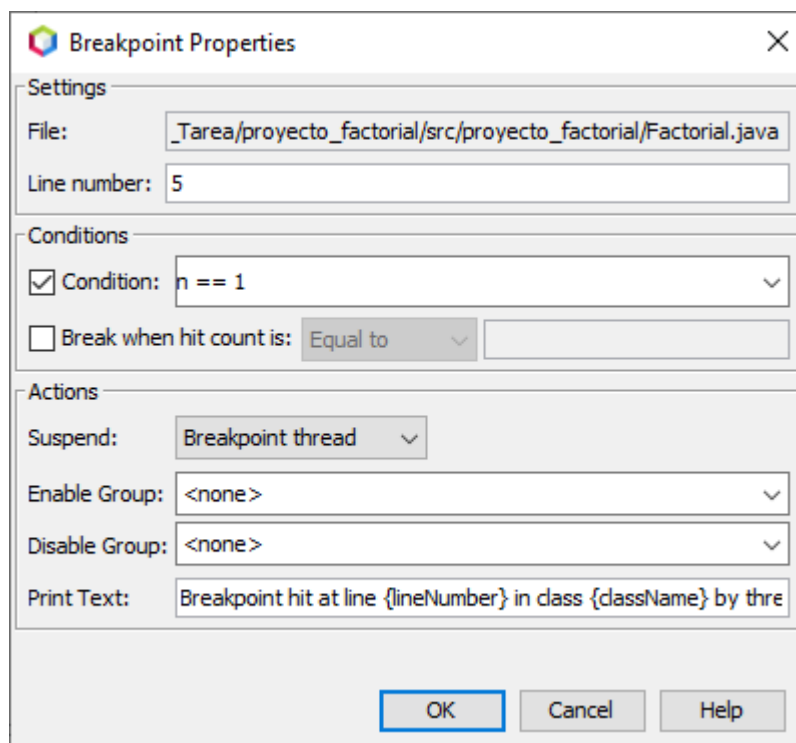
}

```

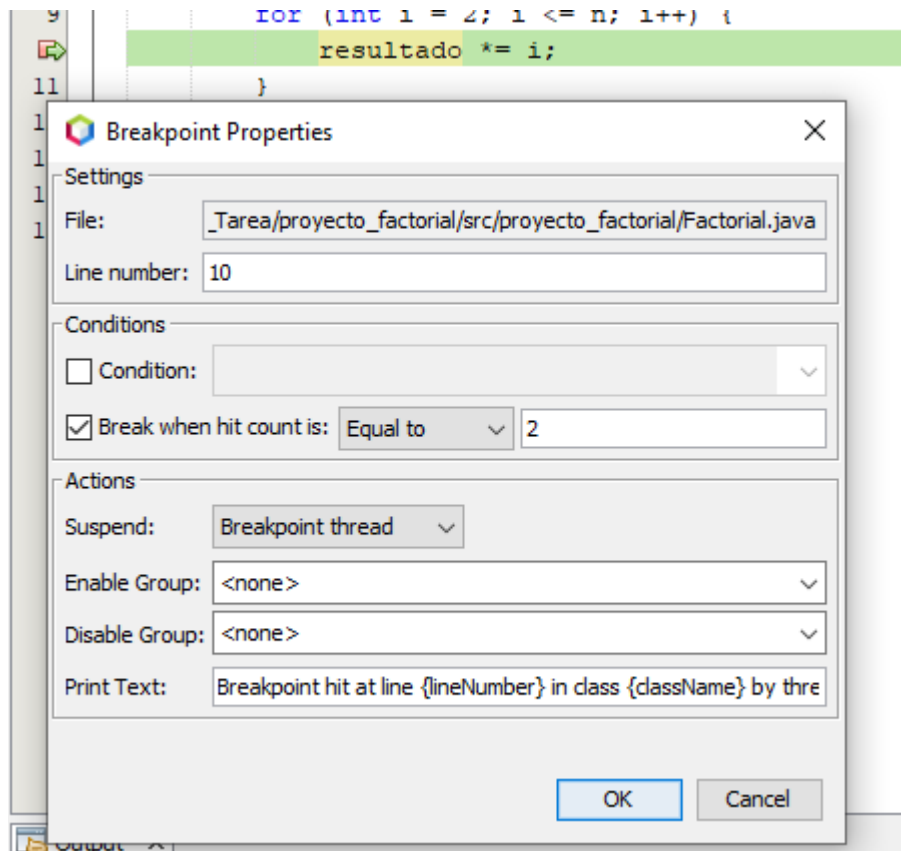
Punto de parada sin condición al crear el objeto en la función main.



Punto de parada con condición que valore que una de las variables del programa tenga un valor determinado.



Punto de parada con condición donde haya un bucle, que solo deberá parar la segunda vez que pasa.



Método busca del proyecto_arrays

Análisis de Caja blanca

Grafo	Nodo	Línea (Condición o Detalle)
	1	Línea 5 (Inicio)
	2	Líneas 6-8 (Variables)
	3	Línea 10 ($a \geq z$)
	4	Línea 10 ($resultado == false$)
	5	Línea 11 (m)
	6	Línea 12 ($v[m] == c$)
	7	Línea 13 ($resultado = true$)
	8	Línea 17 ($v[m] < c$)
	9	Línea 18 ($a = m + 1$)
	10	Línea 21 ($z = m - 1$)
	11	Línea 25 (return)
	12	Línea 26 (Final)

Método de cálculo	Complejidad	Comentarios
Número de regiones	5	
$n.aristas - n.nodos + 2$	$15 - 12 + 2 = 5$	
$n.condiciones + 1$	$4 + 1 = 5$	Nodos 3, 4, 6 y 8

Camino	Nodos
CAM1	1 - 2 - 3 - 11 - 12
CAM2	1 - 2 - 3 - 4 - 11 - 12
CAM3	1 - 2 - 3 - 4 - 5 - 6 - 7 - 3 - 11 - 12

CAM4	1 - 2 - 3 - 4 - 5 - 6 - 7 - 3 - 4 - 11 - 12
CAM5	1 - 2 - 3 - 4 - 5 - 6 - 8 - 9 - 3 - 11 - 12
CAM6	1 - 2 - 3 - 4 - 5 - 6 - 8 - 9 - 3 - 4 - 11 - 12
CAM7	1 - 2 - 3 - 4 - 5 - 6 - 8 - 10 - 3 - 11 - 12
CAM8	1 - 2 - 3 - 4 - 5 - 6 - 8 - 10 - 3 - 4 - 11 - 12

Análisis de Caja negra

Tipo de dato	Descripción	Clases equivalencia	Valor límite
Array de caracteres	Array de hasta 10 caracteres	Clase válida Array entre 0 y 10 caracteres	ALV1: 0 ALV2: 1 ALV3: 10
		Clases no válidas Pasar como parámetro un array nulo	

Conjetura de errores:

- CE1: El array es nulo
- CE2: No hay ningún límite en el array, por lo que podría haber más de 10 caracteres
- CE3: La introducción de caracteres en el array no está validada, por lo que no hay forma de garantizar que estén ordenados.

ID Caso de prueba	Entrada		ID de Clase / Camino / ALV / CE	Salida
	C	V		
C1	'B'	{}	ALV1	false
C2A	'B'	{'A'}	ALV2	false
C2B	'B'	{'B'}	ALV2	true
C3A	'B'	{'A','B','C','D','E','F','G','H','I','J'}	ALV3	true
C3B	'R'	{'A','B','C','D','E','F','G','H','I','J'}	ALV3	false
C4	'B'	null	CE1	NullPointerException


```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package buscarcaracter;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author mskin
 */
public class OperacionsArraysTest {

    OperacionsArrays busca;

    public OperacionsArraysTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }

    @AfterClass
    public static void tearDownClass() {
    }

    @Before
    public void setUp() {
        busca = new OperacionsArrays();
    }

    @After
    public void tearDown() {
    }

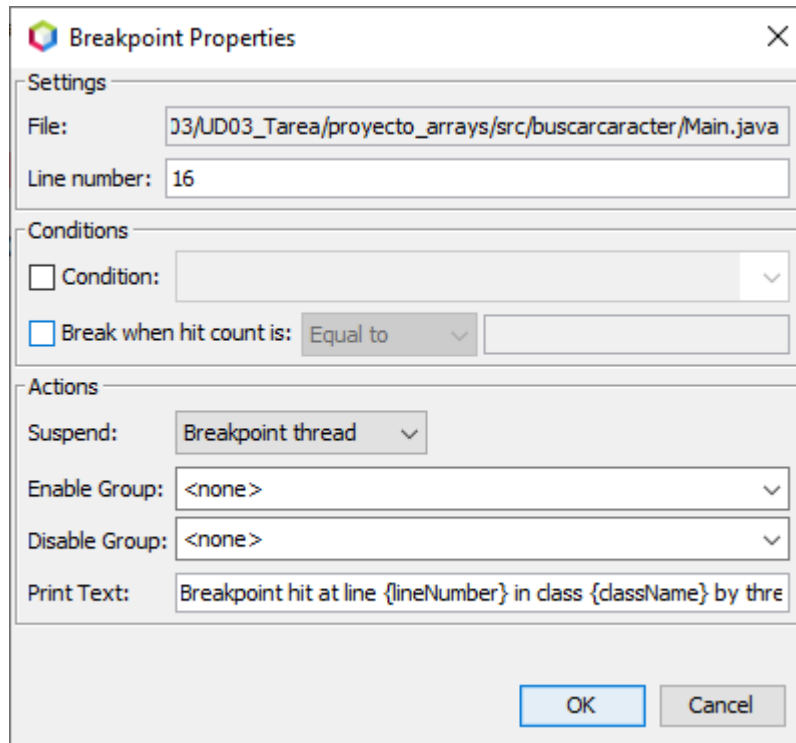
    // TODO add test methods here.
    // The methods must be annotated with annotation @Test. For example:
```

```
//  
// @Test  
// public void hello() {}  
  
@Test  
public void testC1() {  
    // Arrange  
    char entrada = 'B';  
    char[] array = {};  
    // Act  
    boolean resultado = busca.busca(entrada,array);  
    // Assert  
    assertFalse(resultado);  
}  
  
@Test  
public void testC2A() {  
    // Arrange  
    char entrada = 'B';  
    char[] array = {'A'};  
    // Act  
    boolean resultado = busca.busca(entrada,array);  
    // Assert  
    assertFalse(resultado);  
}  
  
@Test  
public void testC2B() {  
    // Arrange  
    char entrada = 'B';  
    char[] array = {'B'};  
    // Act  
    boolean resultado = busca.busca(entrada,array);  
    // Assert  
    assertTrue(resultado);  
}  
  
@Test  
public void testC3A() {  
    // Arrange  
    char entrada = 'B';  
    char[] array = {'A','B','C','D','E','F','G','H','I','J'};  
    // Act  
    boolean resultado = busca.busca(entrada,array);  
    // Assert  
    assertTrue(resultado);  
}
```

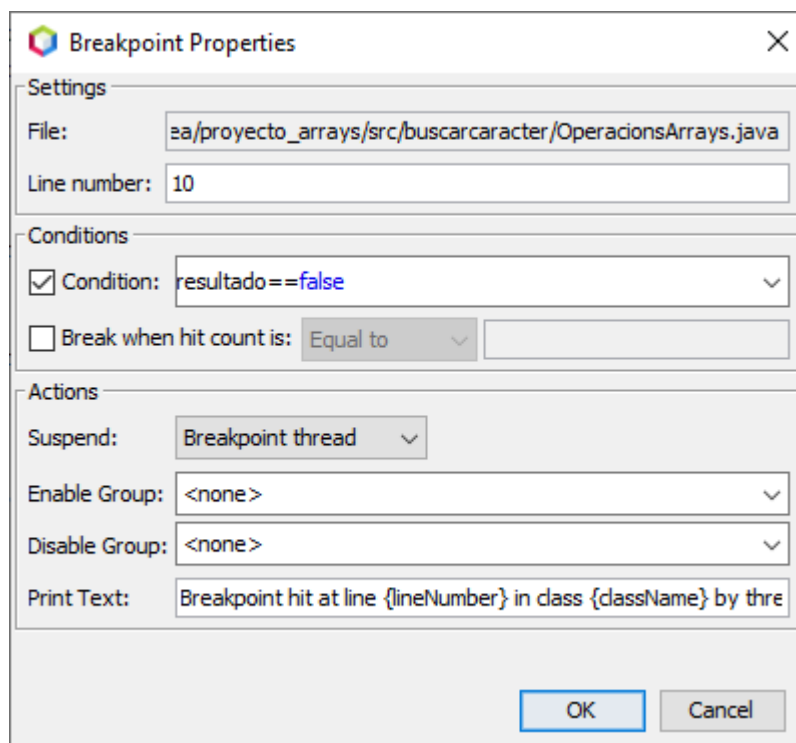
```
@Test
public void testC3B() {
    // Arrange
    char entrada = 'R';
    char[] array = {'A','B','C','D','E','F','G','H','I','J'};
    // Act
    boolean resultado = busca.busca(entrada,array);
    // Assert
    assertFalse(resultado);
}

@Test(expected = NullPointerException.class)
public void testC4() {
    char entrada = 'B';
    char[] array = null;
    boolean resultado = busca.busca(entrada,array);
}
}
```


Punto de parada sin condición al crear el objeto en la función main.



Punto de parada con condición que valore que una de las variables del programa tenga un valor determinado.



Punto de parada con condición donde haya un bucle, que solo deberá parar la segunda vez que pasa.

 Breakpoint Properties

×

Settings

File:

Line number:

Conditions

☐ Condition:

☒ Break when hit count is:

Equal to

Actions

Suspend:

Breakpoint thread

Enable Group:

<none>

Disable Group:

<none>

Print Text:

OK

Cancel