

Universidad del Valle de Guatemala
Departamento de ingeniería
Algoritmos y estructura de datos
Kevin Macario 17369
Rodrigo Urrutia 16139

Hoja de Trabajo 4 - Calculadora

Link de GitHub: <https://github.com/RodrigoUrrutiaC/HT4-AED>

Ventajas / desventajas al utilizar el patrón Singleton en general (ya que su comportamiento es muy similar a una variable global). Cree que su uso es adecuado en este programa?

Sí fue de utilidad implementar este patrón de diseño en el programa, ya que con este se puede garantizar que solo exista una instancia del objeto. Una de las mayores ventajas es que proporciona su acceso globalmente a las clases utilizadas y que al garantizar que solo hay una instancia del objeto se previene que hayan instancias de objetos inutilizadas, por lo que previene el desaprovechamiento o mal uso de la memoria. La desventaja más significativa es que resulta incomodo a la hora de acceder a ella, debido a que se debe de acceder por medio de la propiedad instance.

Evidencias de las pruebas JUnit

CalculadoraTest

Prueba fallida:

```
55      @Test
56      public void testCalcular() {
57          System.out.println("calcular");
58          String exp = "2 2 +";
59          int x = 1;
60          int y = 1;
61          Calculadora instance = new Calculadora();
62          String expectedResult = "4";
63          String result = instance.calcular(exp, x, y);
64          assertEquals(expResult, result);
65      }
66  }
67
68  }
69
```

ht4.CalculadoraTest > testGetInstance >

Test Results	Output
ht4.CalculadoraTest	
Tests passed: 50.00 %	
1 test passed, 1 test failed. (0.349 s)	
ht4.CalculadoraTest	Failed
testCalcular	passed (0.02 s)
testGetInstance	Failed: expected: <null> but was: <ht4.Calculadora@6504e3b2>

Prueba exitosa:

The screenshot displays an IDE window with a Java test class named `ht4.CalculadoraTest`. The code defines a test method `testCalcular()` that verifies the `calcular` method of the `Calculadora` class. The test method sets up a `Calculadora` instance, defines an expression `"2 2 +"`, and asserts that the calculated result `"4"` matches the expected result.

```
52  /**
53   * Test of calcular method, of class Calculadora.
54   */
55   @Test
56   public void testCalcular() {
57       System.out.println("calcular");
58       String exp = "2 2 +";
59       int x = 1;
60       int y = 1;
61       Calculadora instance = new Calculadora();
62       String expectedResult = "4";
63       String result = instance.calcular(exp, x, y);
64       assertEquals(expResult, result);
65   }
66 }
67
68
69
```

Below the code editor, the **Test Results** tab is active, showing the execution of the `testCalcular` method. A green bar at the top indicates that all tests passed with 100.00% success. The summary shows that both tests passed in 0.241 seconds. The detailed list shows three tests: `ht4.CalculadoraTest` passed, `testCalcular` passed in 0.017 seconds, and `testGetInstance` passed in 0.0 seconds.

Test Results × Output

ht4.CalculadoraTest ×

Tests passed: 100.00 %

Both tests passed. (0.241 s)

- ht4.CalculadoraTest passed
- testCalcular passed (0.017 s)
- testGetInstance passed (0.0 s)

StackFactoryTest

Prueba fallida:

```
3      @Test
4      public void testGetStack() {
5          System.out.println("getStack");
6          String entry1 = "2";
7          String entry2 = "0";
8          StackFactory instance = new StackFactory();
9          Stack expectedResult = new StackVector<>();
10         Stack result = instance.getStack(entry1, entry2);
11         assertEquals(expectedResult, result);
12     }
13 }

ht4.StackFactoryTest >
Test Results x Output
4.CalculadoraTest x ht4.StackFactoryTest x
Tests passed: 0.00 %
No test passed, 1 test failed. (0.101 s)
ht4.StackFactoryTest Failed
testGetStack Failed: expected: <ht4.StackVector@6d9c638> but was: <null>
```

Prueba exitosa:

```
3      @Test
4      public void testGetStack() {
5          System.out.println("getStack");
6          String entry1 = "2";
7          String entry2 = "0";
8          StackFactory instance = new StackFactory();
9          Stack expectedResult = new StackVector<>();
10         Stack result = instance.getStack(entry1, entry2);
11         assertEquals(expectedResult, result);
12     }
13 }

ht4.StackFactoryTest >
Results x Output
CalculadoraTest x ht4.StackFactoryTest x
Tests passed: 100.00 %
The test passed. (0.088 s)
ht4.StackFactoryTest passed
testGetStack passed (0.004 s)
```

Estas clases son las dos únicas a las que se les pueden hacer pruebas JUnit. Las demás son implementadas de Canvas.