JUnit

Fundamentals to programming I

Delgado Valencia Franco Andre [1]
Gallegos Condori Anette Isabel [2]

[1]System Engineering School
System Engineering and Informatic Department
Production and Services Faculty
San Agustin National University of Arequipa

28/07/2020

JUnit

Content

1. What is JUnit?
2. Implementation
3. Test-Driven Development
4. Examples
   - BlackJack
   - BlackJackTest
   - MissingChar
   - MissingCharTest
   - TablasDeVerdad
   - TablasDeVerdadTest
   - ADN
   - ADNTest
5. References

What is JUnit?

Tool specially designed to implement and automate the unit test relationship in **JAVA**.

Implementation

1. In a separate class
   - The class inherits from *org.junit.jupiter.api.Test*.
   - Each test case is implemented in a separate method.
   - Test case names begin with *Test*.

```
3⊖ import static org.junit.jupiter.api.Assertions.*;
4
5  import org.junit.jupiter.api.Test;
6
7  class BlackJackTest {
8
10⊕    void testSuma(){..
37 }
```

Implementation

2. Each test case invokes a series of methods in our class and checks the results obtained after invoking them.

- We create the test with *@Test*.
- Depending on the type and amount of data that were used in the class, they will be used in the Test.
- *AssertEquals()* methods checks that the two objects are equals or not.

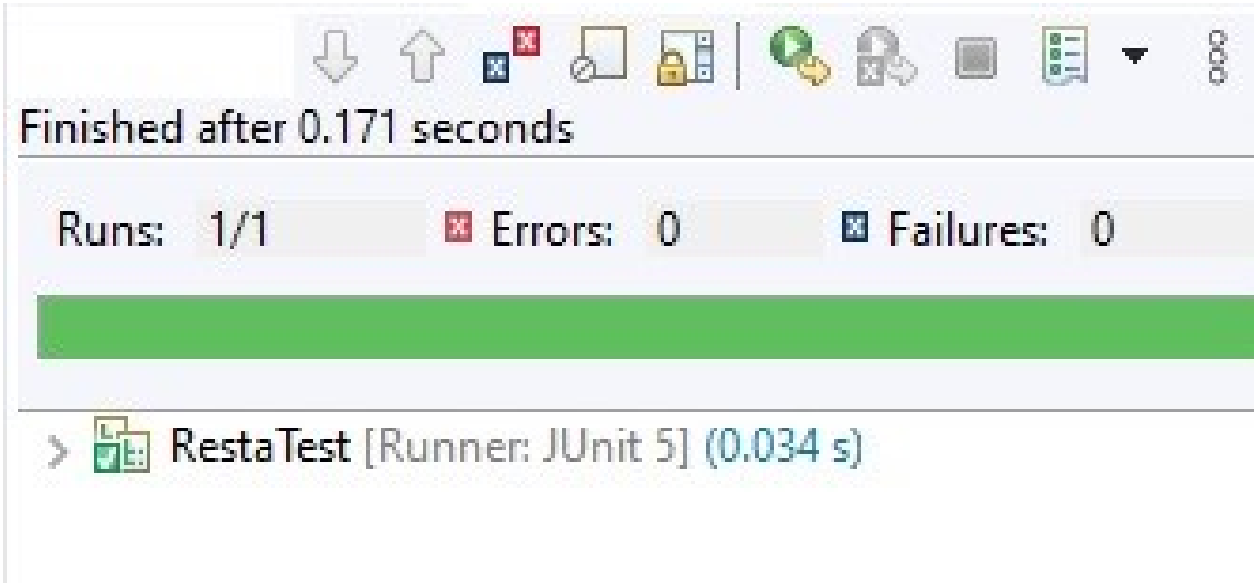
```
class BlackJackTest {

    @Test
    void testSuma(){

        boolean actual = BlackJack.suma(12,5);
        boolean expected = true;
        assertEquals(actual,expected);

        actual = BlackJack.suma(10,11);
        expected = true;
        assertEquals(actual,expected);
```

```
class MissingCharTest {

    @Test
    void testMissingChar() {
        String actual = MissingChar.missingChar("Descisión", 2);
        String expected = "Decisión";
        assertEquals(expected,actual);

        actual = MissingChar.missingChar("Switchs", 6);
        expected = "Switch";
        assertEquals(expected,actual);
```

Note: The Test may vary depending on the class.

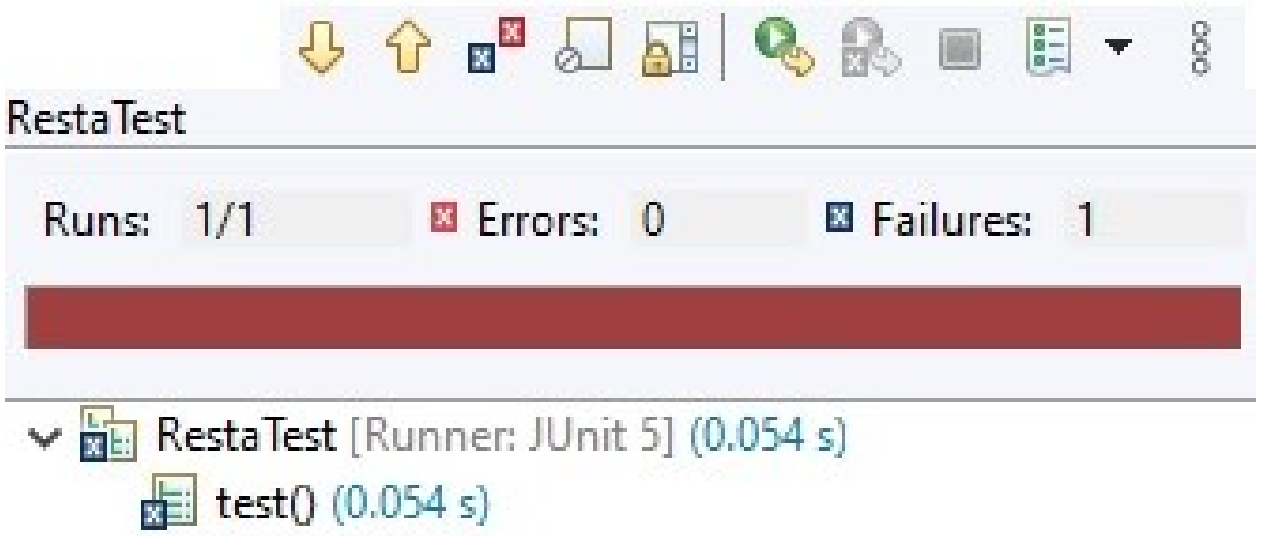
Implementation

3. Finally, we run the test cases in JUnit.
   - If all cases work correctly:



Implementation

- If any test case fails:



Test-Driven Development

It consists of implementing unit test before even starting to write the code of a module.

Traditionally

Tests are carried out after.

Test cases are generally written after implementing the module whose operation they are intended to verify.

TDD

Test are prepared before you start writing the code.

First we write a test case. Then we write the necessary code for the test case to pass successfully.

Vantages

- If we write the test cases first, we define the requirements that we expect our application to meet.
- When we write a test case. We think about the correct way to use a module that does not exist.
- Test cases execution is done automatically.
- Test cases allow us to lose the fear of making modifications to the code.
- Test cases determine when our work done.

Software construction process

It's a cycle

1. Add a new test case.
2. Run test cases.
3. Make small changes to the implementation.
4. Run test cases.
5. Improve code design.
6. Run test cases.
7. Return to the initial step.

Examples

BlackJack

```
1  package junit;
2
3  public class BlackJack{
4
5      public static boolean suma(int a, int b){
6
7          if((a+b)<=21) {
8          return true;
9          }
10         return false;
11     }
12 }
13
```

BlackJackTest

```java
package junit;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class BlackJackTest {

    @Test
    void testSuma(){

        boolean actual = BlackJack.suma(12,5);
        boolean expected = true;
        assertEquals(actual,expected);

        actual = BlackJack.suma(10,11);
        expected = true;
        assertEquals(actual,expected);

        actual = BlackJack.suma(13,7);
        expected = true;
        assertEquals(actual,expected);

        actual = BlackJack.suma(11,12);
        expected = false;
        assertEquals(actual,expected);

        actual = BlackJack.suma(14,9);
        expected = false;
        assertEquals(actual,expected);

        actual = BlackJack.suma(13,10);
        expected = false;
        assertEquals(actual,expected);

    }
}
```

MissingChar

```java
package Básico;
public class MissingChar {
    public static String missingChar(String str,int n) {
        String strx="";
        for(int i=0;i<str.length();i++)
            if(i!=n)
                strx+=str.charAt(i);
        return strx;
    }
}
```

MissingCharTest

```java
package Básico;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class MissingCharTest {

    @Test
    void testMissingChar() {
        String actual = MissingChar.missingChar("Descisión", 2);
        String expected = "Decisión";
        assertEquals(expected,actual);

        actual = MissingChar.missingChar("Switchs", 6);
        expected = "Switch";
        assertEquals(expected,actual);

        actual = MissingChar.missingChar("z", 0);
        expected = "";
        assertEquals(expected,actual);

        actual = MissingChar.missingChar("", 0);
        expected = "";
        assertEquals(expected,actual);
    }
}
```

TablasDeVerdad

```java
package junit;

public class TablasDeVerdad{

    public static boolean negación(boolean a){
        if(!a) {
        return true;
        }
        return false;
     }
    public static boolean conjuncion(boolean a, boolean b){
        if(a && b) {
        return true;
        }
        return false;
     }
    public static boolean disyuncion(boolean a, boolean b){
        if(a || b) {
        return true;
        }
        return false;
     }
    public static boolean condicional(boolean a, boolean b){
        if(!a || b) {
        return true;
        }
        return false;
     }
    public static boolean bicondicional(boolean a, boolean b){
        if((a && b) || (!a && !b)) {
        return true;
        }
        return false;
     }
}
```

TablasDeVerdadTest

```java
package junit;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class TablasDeVerdadTest {

    @Test
      void testNegacion() {

         boolean actual = TablasDeVerdad.negación(true);
         boolean expected = false;
         assertEquals(actual,expected);

         actual = TablasDeVerdad.negación(false);
         expected = true;
         assertEquals(actual,expected);
    }

    @Test
      void testConjuncion() {

         boolean actual = TablasDeVerdad.conjuncion(false, true);
         boolean expected = false;
         assertEquals(actual,expected);

         actual = TablasDeVerdad.conjuncion(true, false);
         expected = false;
         assertEquals(actual,expected);

         actual = TablasDeVerdad.conjuncion(true, true);
         expected = true;
         assertEquals(actual,expected);

         actual = TablasDeVerdad.conjuncion(false, false);
         expected = false;
         assertEquals(actual,expected);
    }
```

TablasDeVerdadTest

```
41    @Test
42     void testDisyuncion() {
43
44        boolean actual = TablasDeVerdad.disyuncion(true, false);
45        boolean expected = true;
46        assertEquals(actual,expected);
47
48        actual = TablasDeVerdad.disyuncion(false, false);
49        expected = false;
50        assertEquals(actual,expected);
51
52        actual = TablasDeVerdad.disyuncion(true, false);
53        expected = true;
54        assertEquals(actual,expected);
55
56        actual = TablasDeVerdad.disyuncion(true, true);
57        expected = true;
58        assertEquals(actual,expected);
59    }
60
61    @Test
62     void testConditional() {
63
64        boolean actual = TablasDeVerdad.condicional(true, false);
65        boolean expected = false;
66        assertEquals(actual,expected);
67
68        actual = TablasDeVerdad.condicional(false, false);
69        expected = true;
70        assertEquals(actual,expected);
71
72        actual = TablasDeVerdad.condicional(false, true);
73        expected = true;
74        assertEquals(actual,expected);
75
76        actual = TablasDeVerdad.condicional(true, true);
77        expected = true;
78        assertEquals(actual,expected);
79    }
```

TablasDeVerdadTest

```
80
81      @Test
82       void testBicondicional() {
83
84          boolean actual = TablasDeVerdad.bicondicional(false, true);
85          boolean expected = false;
86          assertEquals(actual,expected);
87
88          actual = TablasDeVerdad.condicional(false, false);
89          expected = true;
90          assertEquals(actual,expected);
91
92          actual = TablasDeVerdad.condicional(true, false);
93          expected = false;
94          assertEquals(actual,expected);
95
96          actual = TablasDeVerdad.condicional(true, true);
97          expected = true;
98          assertEquals(actual,expected);
99      }
100 }
101
```

ADN

```java
package Avanzado;

public class ADN {
    public static boolean valido(String adn) {
        int contador=0;
        boolean valido;
        do {
            switch (adn.charAt(contador)) {
            case 'A':
                valido=true;
                break;
            case 'C':
                valido=true;
                break;
            case 'G':
                valido=true;
                break;
            case 'T':
                valido=true;
                break;
            default:
                valido=false;
                break;
            }
            contador++;
        }
        while(contador<adn.length()&valido==true);
        return valido;
    }
    public static String mutarAmp(String adn, int a, int b) {
        if(valido(adn)) {
            String adnx;
            adnx=adn.substring(0,b)+adn.substring(a);
            return adnx;
        }
        else return "Cadena no válida";
    }
```

ADN

```
38    public static String mutarDel(String adn, int a, int b) {
39        if(valido(adn)) {
40            String adnx;
41            adnx=adn.substring(0,a)+adn.substring(b);
42            return adnx;
43        }
44        else return "Cadena no válida";
45    }
46    public static String mutarInv(String adn, int a, int b) {
47        if(valido(adn)) {
48            String adnx="";
49            int i;
50            for(i=0;i<a;i++) {
51                adnx=adnx+adn.charAt(i);
52            }
53            for(i=b-1;i>=a;i--) {
54                adnx=adnx+adn.charAt(i);
55            }
56            for(i=b;i<adn.length();i++) {
57                adnx=adnx+adn.charAt(i);
58            }
59            return adnx;
60        }
61        else return "Cadena no válida";
62    }
63 }
```

ADNTest

```java
package Avanzado;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class ADNTest {

    @Test
    public void valido() {
        boolean adn=ADN.valido("AHHFKVNJ");
        assertFalse(adn);// assertFalse(adn) = assertEquals(false, adn);

        adn=ADN.valido("AAHHFGTGS");
        assertFalse(adn);

        adn=ADN.valido("ACGTCGCGATCA");
        assertTrue(adn);// assertTrue(adn) = assertEquals(true, adn);
    }

    @Test
    void mutarAmp() {
        String adn=ADN.mutarAmp("ACCTGAC", 0,3);
        assertEquals("ACCACCTGAC", adn);

        adn=ADN.mutarAmp("ACCTGAC", 1,3);
        assertEquals("ACCCCTGAC", adn);

        adn=ADN.mutarAmp("JASDCJIRDA", 3,6);
        assertEquals("Cadena no válida", adn);
    }
```

ADNTest

```
32      @Test
33      void mutarDel() {
34          String adn=ADN.mutarDel("ACGTTAGCTA", 1, 5);
35          assertEquals("AAGCTA", adn);
36
37          adn=ADN.mutarDel("ACGGCGATGCA", 0, 4);
38          assertEquals("CGATGCA", adn);
39
40          adn=ADN.mutarDel("AHJFSKJDKLAS", 2, 5);
41          assertEquals("Cadena no válida", adn);
42      }
43
44      @Test
45      void mutarInv() {
46          String adn=ADN.mutarInv("AGGTCTCGA", 0, 4);
47          assertEquals("TGGACTCGA", adn);
48
49          adn=ADN.mutarInv("ACGTAACGA", 3, 7);
50          assertEquals("ACGCAATGA", adn);
51
52          adn=ADN.mutarInv("ACGKKAHSFA", 4, 6);
53          assertEquals("Cadena no válida", adn);
54      }
55  }
```

References

1 https://elvex.ugr.es/decsai/java/pdf/8C-JUnit.pdf

2 https://junit.org/junit5/docs/current/user-guide/ *[recommended visit]*

Thanks for your attention