

**ALEJANDRO SALGADO CERDEIRA**

**ENTORNOS DE DESARROLLO 2021**

**TAREA JUNIT PROPIA**

## **1.- INTRODUCCIÓN Y PLANTEAMIENTO**

Vamos a crear una aplicación para controlar años bisiestos. Para ello desarrollaremos un código JAVA que constará de una función booleana que comprobará si un año es bisiesto o no y devolverá true o false según las condiciones. Y después crearemos un método para añadir los años solo si son bisiestos a un ArrayList; además imprimirá por consola mensajes de éxito o error en función al resultado de la inserción.

## **2.- DESARROLLO DEL CÓDIGO JAVA**

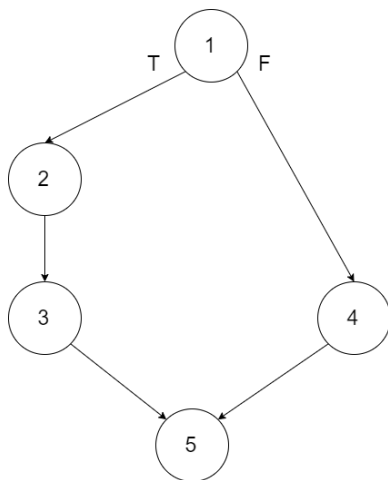
```
public class ListaBisiestos {  
    private ArrayList<Integer> listaBisiestos;  
    private Integer year;  
  
    public ListaBisiestos () {  
        this.listaBisiestos= new ArrayList<Integer>();  
    }  
  
    public void addYear (Integer year){  
        if(esBisiesto(year)){  
            this.listaBisiestos.add(year);  
            System.out.println("Año añadido con éxito:");  
            System.out.println(listaBisiestos.toString());  
        }  
        else  
            System.out.println("El año no es bisiesto y no puede añadirse.");  
        }  
  
    public boolean esBisiesto (Integer year){  
        if (year % 4 == 0) {  
            if (year % 100 == 0) {  
                if (year % 400 == 0)  
                    return true;  
                else  
                    return false;  
            }  
            else  
                return true;  
        }  
        else  
            return false;  
        }  
    }  
}
```

### 3.- PRUEBAS DE CAJA BLANCA:

Debido a que la aplicación contiene llamadas a funciones internas, he decidido dividirla en dos grafos para simplificar su comprensión y teniendo en cuenta que en ambos casos se ha añadido un nodo extra para el fin de proceso:

```
public void addYear (Integer year){  
    if(esBisiesto(year)){  
        this.listaBisiestos.add(year);  
        System.out.println("Año añadido con éxito:");  
        System.out.println(listaBisiestos.toString());  
    }  
    else  
        System.out.println("El año no es bisiesto y no puede añadirse.");  
}
```

1  
2  
3  
4



#### CÁLCULO DE COMPLEJIDAD CICLOMÁTICA:

$$V(G) = a - n + 2 = 5 - 5 + 2 = 2$$

$$V(G) = c + 1 = 1 + 1 = 2$$

$$V(G) = r = 2$$

#### CAMINOS:

1,2,3,5

1,4,5

#### CASOS DE PRUEBA:

year	esBisiesto	inserción	salida
2000	true	this.listaBisiestos.add(2000)	"Año añadido con éxito:" [2000]
2020	true	this.listaBisiestos.add(2020)	"Año añadido con éxito:" [2020]
1900	false	No ocurre	"El año no es bisiesto y no pudo añadirse"
1921	false	No ocurre	"El año no es bisiesto y no pudo añadirse"

```

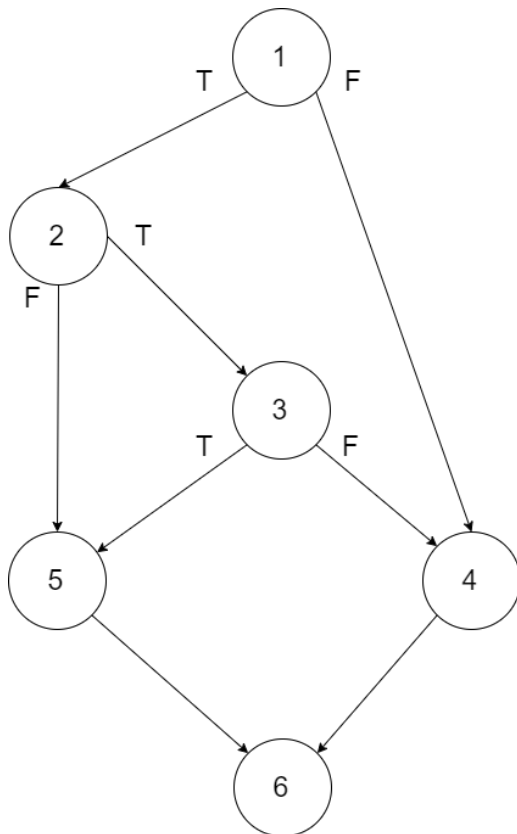
public boolean esBisiesto (Integer year){

    // Si el año es divisible entre 4
    if (year % 4 == 0) {
        // Si el año es un siglo
        if (year % 100 == 0) {
            // Si el año se puede dividir entre 400 entonces es bisiesto:
            if (year % 400 == 0)
                return true;
            else
                return false;
        }

        // Si el año no es un siglo es bisiesto:
        else
            return true;
    }

    //Si no se cumple ninguna condición, no es bisiesto:
    else
        return false;
}

```



#### CÁLCULO DE COMPLEJIDAD CICLOMÁTICA:

$$V(G) = a - n + 2 = 8 - 6 + 2 = 4$$

$$V(G) = c + 1 = 3 + 1 = 4$$

$$V(G) = r = 4$$

#### CAMINOS:

1,2,3,5,6

1,2,3,4,6

1,2,5,6

1,4,6

#### CASOS DE PRUEBA:

year	devuelve
2000	true
2020	True
1900	false
1921	false

## 4.- PRUEBAS CON NETBEANS:

Insertamos los casos que hemos planteado en JUnit y comprobamos que devuelven los resultados esperados correctamente, dando por finalizado el proceso de pruebas:

### 4.1.- 2000:

The screenshot shows the NetBeans IDE with a Java source file and a Test Results window. The source file contains two JUnit tests: `testEsBisiesto()` and `testAddYear()`. The `testEsBisiesto()` test sets `year = 2000` and checks if the year is a leap year. The `testAddYear()` test sets `year = 2000` and checks if the year is added to the list. The Test Results window shows that both tests passed successfully. The output of the `addYear` test is displayed on the right.

```
43 @Test
44 public void testEsBisiesto() {
45     System.out.println("esBisiesto");
46     Integer year = 2000;
47     ListaBisiestos instance = new ListaBisiestos();
48     boolean expectedResult = true;
49     boolean result = instance.esBisiesto(year);
50     assertEquals(expectedResult, result);
51     // TODO review the generated test code and remove the default call to fail.
52     //fail("The test case is a prototype.");
53 }
54
55 /**
56  * Test of addYear method, of class ListaBisiestos.
57  */
58 @Test
59 public void testAddYear() {
60     System.out.println("addYear");
61     Integer year = 2000;
62     ListaBisiestos instance = new ListaBisiestos();
63     instance.addYear(year);
64     // TODO review the generated test code and remove the default call to fail.
65     //fail("The test case is a prototype.");
66 }
```

Test Results

UD3\_Pract3\_JUnit.ListaBisiestosIT x

Tests passed: 100,00 %

Both tests passed. (0,079 s)

- UD3\_Pract3\_JUnit.ListaBisiestosIT passed
- testAddYear passed (0,0 s)
- testEsBisiesto passed (0,0 s)

addYear  
Año añadido con éxito:  
[2000]  
esBisiesto

### 4.2.- 2020

The screenshot shows the NetBeans IDE with a Java source file and a Test Results window. The source file contains two JUnit tests: `testEsBisiesto()` and `testAddYear()`. The `testEsBisiesto()` test sets `year = 2020` and checks if the year is a leap year. The `testAddYear()` test sets `year = 2020` and checks if the year is added to the list. The Test Results window shows that both tests passed successfully. The output of the `addYear` test is displayed on the right.

```
43 @Test
44 public void testEsBisiesto() {
45     System.out.println("esBisiesto");
46     Integer year = 2020;
47     ListaBisiestos instance = new ListaBisiestos();
48     boolean expectedResult = true;
49     boolean result = instance.esBisiesto(year);
50     assertEquals(expectedResult, result);
51     // TODO review the generated test code and remove the default call to fail.
52     //fail("The test case is a prototype.");
53 }
54
55 /**
56  * Test of addYear method, of class ListaBisiestos.
57  */
58 @Test
59 public void testAddYear() {
60     System.out.println("addYear");
61     Integer year = 2020;
62     ListaBisiestos instance = new ListaBisiestos();
63     instance.addYear(year);
64     // TODO review the generated test code and remove the default call to fail.
65     //fail("The test case is a prototype.");
66 }
```

Test Results

UD3\_Pract3\_JUnit.ListaBisiestosIT x

Tests passed: 100,00 %

Both tests passed. (0,141 s)

- UD3\_Pract3\_JUnit.ListaBisiestosIT passed

addYear  
Año añadido con éxito:  
[2020]  
esBisiesto

#### 4.3.- 1900

The screenshot shows an IDE with a Java source file. The code defines two test methods: `testEsBisiesto()` and `testAddYear()`. `testEsBisiesto()` tests the `esBisiesto` method with the year 1900, expecting it to return `false`. `testAddYear()` tests the `addYear` method with the year 1900. Both tests use `assertEquals` and `fail` methods. The Test Results window at the bottom shows that both tests passed. The `addYear` method's output is also visible: "El año no es bisiesto y no puede añadirse. esBisiesto".

```
43 @Test
44 public void testEsBisiesto() {
45     System.out.println("esBisiesto");
46     Integer year = 1900;
47     ListaBisiestos instance = new ListaBisiestos();
48     boolean expectedResult = false;
49     boolean result = instance.esBisiesto(year);
50     assertEquals(expectedResult, result);
51     // TODO review the generated test code and remove the default call to fail.
52     //fail("The test case is a prototype.");
53 }
54
55 /**
56  * Test of addYear method, of class ListaBisiestos.
57  */
58 @Test
59 public void testAddYear() {
60     System.out.println("addYear");
61     Integer year = 1900;
62     ListaBisiestos instance = new ListaBisiestos();
63     instance.addYear(year);
64     // TODO review the generated test code and remove the default call to fail.
65     //fail("The test case is a prototype.");
66 }
```

**Test Results**

UD3\_Pract3\_Unit.ListaBisiestosIT X

Tests passed: 100,00 %

Both tests passed. (0,187 s)

- UD3\_Pract3\_Unit.ListaBisiestosIT passed
- testAddYear passed (0,015 s)
- testEsBisiesto passed (0,0 s)

addYear  
El año no es bisiesto y no puede añadirse.  
esBisiesto

#### 4.4.- 1921

The screenshot shows the same IDE as before, but with the year 1921 used in the tests. `testEsBisiesto()` tests the `esBisiesto` method with the year 1921, expecting it to return `false`. `testAddYear()` tests the `addYear` method with the year 1921. Both tests use `assertEquals` and `fail` methods. The Test Results window at the bottom shows that both tests passed. The `addYear` method's output is also visible: "El año no es bisiesto y no puede añadirse. esBisiesto".

```
43 @Test
44 public void testEsBisiesto() {
45     System.out.println("esBisiesto");
46     Integer year = 1921;
47     ListaBisiestos instance = new ListaBisiestos();
48     boolean expectedResult = false;
49     boolean result = instance.esBisiesto(year);
50     assertEquals(expectedResult, result);
51     // TODO review the generated test code and remove the default call to fail.
52     //fail("The test case is a prototype.");
53 }
54
55 /**
56  * Test of addYear method, of class ListaBisiestos.
57  */
58 @Test
59 public void testAddYear() {
60     System.out.println("addYear");
61     Integer year = 1921;
62     ListaBisiestos instance = new ListaBisiestos();
63     instance.addYear(year);
64     // TODO review the generated test code and remove the default call to fail.
65     //fail("The test case is a prototype.");
66 }
```

**Test Results**

UD3\_Pract3\_Unit.ListaBisiestosIT X

Tests passed: 100,00 %

Both tests passed. (0,11 s)

- UD3\_Pract3\_Unit.ListaBisiestosIT passed
- testAddYear passed (0,016 s)
- testEsBisiesto passed (0,0 s)

addYear  
El año no es bisiesto y no puede añadirse.  
esBisiesto