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.

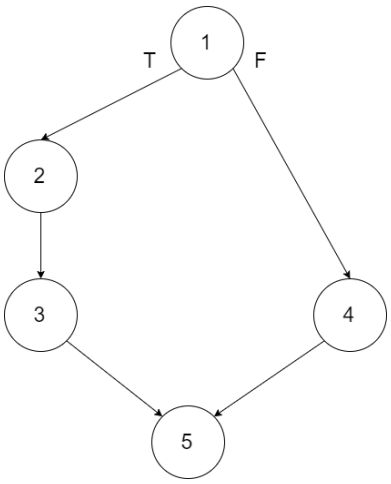
2.- DESARROLLO DEL CÓDIGO JAVA

```
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)){                                          1
        this.listaBisiestos.add(year);                             2
        System.out.println("Año añadido con éxito:");              3
        System.out.println(listaBisiestos.toString());
    }
    else
        System.out.println("El año no es bisiesto y no puede añadirse.");   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:

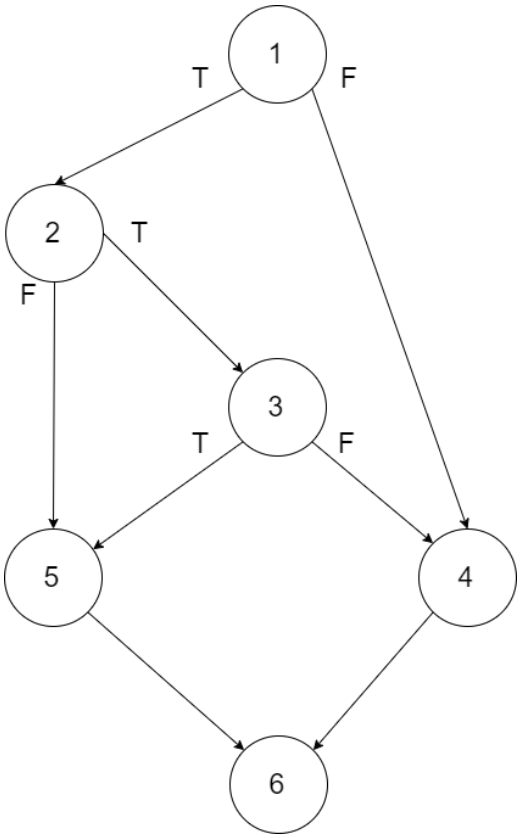
1,2,3,5

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”

```java
public boolean esBisiesto (Integer year){

    // Si el año es divisible entre 4
if (year % 4 == 0) {                                                    1

  // Si el año es un siglo
  if (year % 100 == 0) {                                                2

    // Si el año se puede dividir entre 400 entonces es bisiesto:
    if (year % 400 == 0)                                                3
      return true;                                                      5
    else
      return false;                                                     4
  }

  // Si el año no es un siglo es bisiesto:
  else                                                                  5
    return true;
}
//Si no se cumple ninguna condición, no es bisiesto:
else                                                                    4
  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

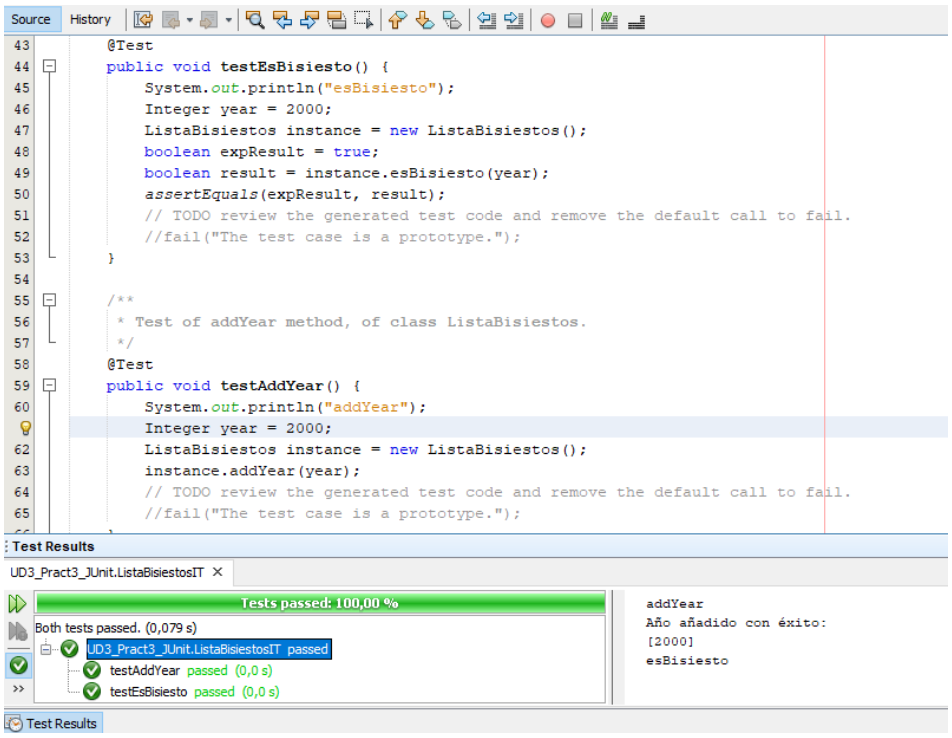
CASOS DE PRUEBA:

year	devuelve
2000	true
2020	True
1900	false
1921	false

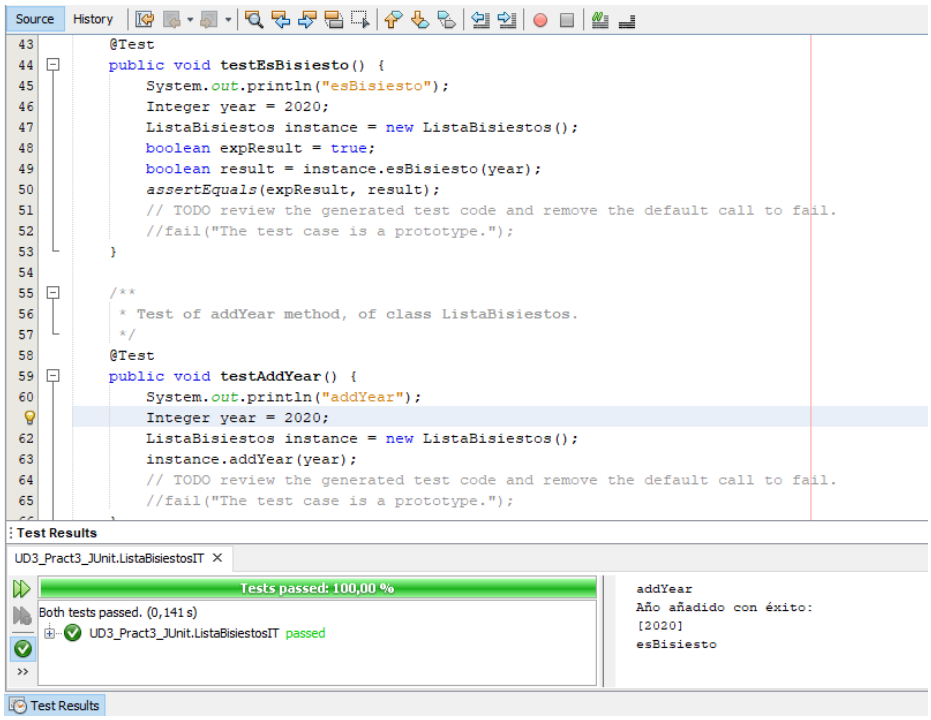
4.- PRUEBAS CON NETBEANS:

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

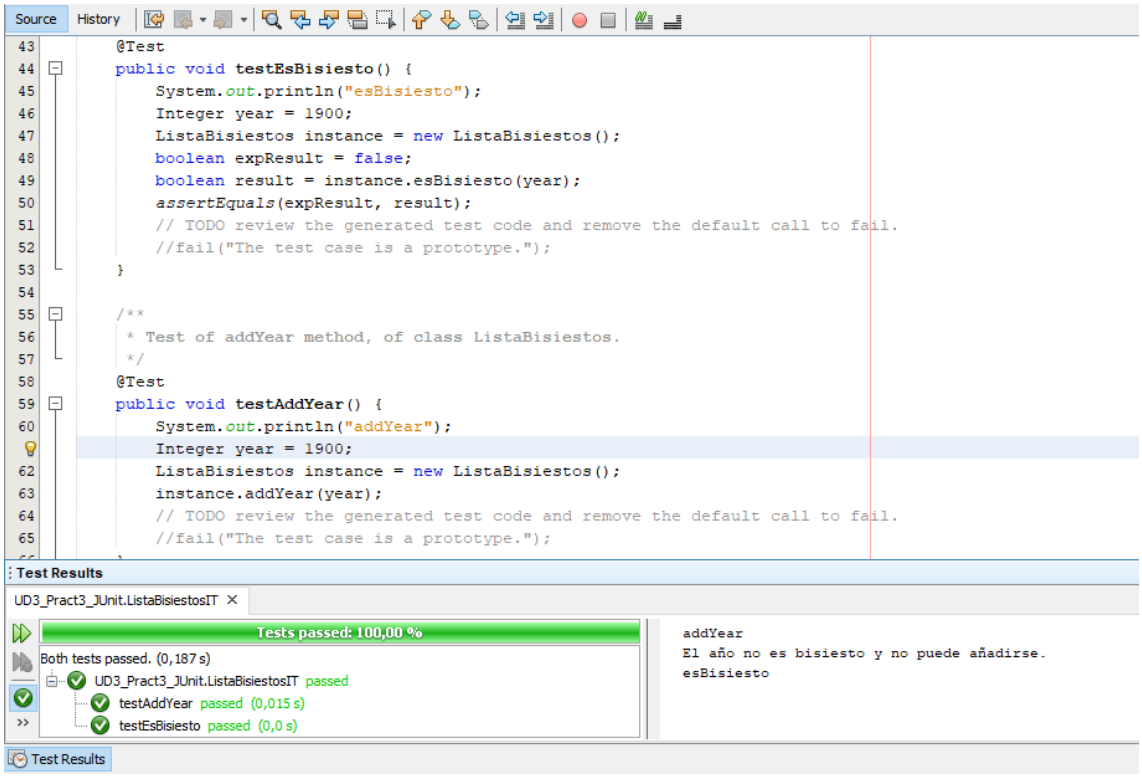
4.1.- 2000:



4.2.- 2020



4.3.- 1900



4.4.- 1921