

ACTIVIDAD PRÁCTICA – DISEÑO DE PRUEBAS

Por Pedro Berrueco

Trabajo propuesto como
cumplimiento de los requisitos para
la actividad práctica de la unidad 4.

Módulo de Entornos de Desarrollo.

Curso en Desarrollo de Aplicaciones
Multiplataforma Online.

Febrero 2023

Índice

ACTIVIDAD PRÁCTICA – DISEÑO DE PRUEBAS.....	- 1 -
Descripción de la actividad Práctica	- 3 -
Introducción.....	- 3 -
Ejercicio 1.....	- 4 -
Ejercicio 2.....	- 5 -
Ejercicio 3.....	- 6 -
Ejercicio 4.....	- 6 -
Ejercicio 5.....	- 7 -
Ejercicio 6.....	- 8 -
Ejercicio 7.....	- 13 -
Ejercicio 8.....	- 15 -
Ejercicio 9 y 10	- 15 -
Conclusiones.....	- 19 -

Descripción de la actividad Práctica

En esta actividad se debe crear un documento profesional tal y como se describe en el entregable. Se recomienda seguir los pasos en orden, aunque pueden saltarse algunos si no se sabe cómo hacerlo. En cada apartado se debe incluir capturas de pantalla de los pasos que se han seguido, junto con la explicación de los pasos que se han dado. Se deberá crear un Diagrama de Flujo a partir de un pseudocódigo facilitado, se debe crear un grafo a partir de este, calcular la complejidad ciclomática, los caminos independientes y exportar este código a JAVA para poder trabajarlo en Eclipse. Se debe ejecutar y explicar el Run Coverage de este código, determinar las diferentes clases de equivalencia e indicar los valores límites. Para terminar se deben realizar y probar los test que prueben el código, asegurándose de que se ejecuten todas las instrucciones y que se prueban todos los valores límites.

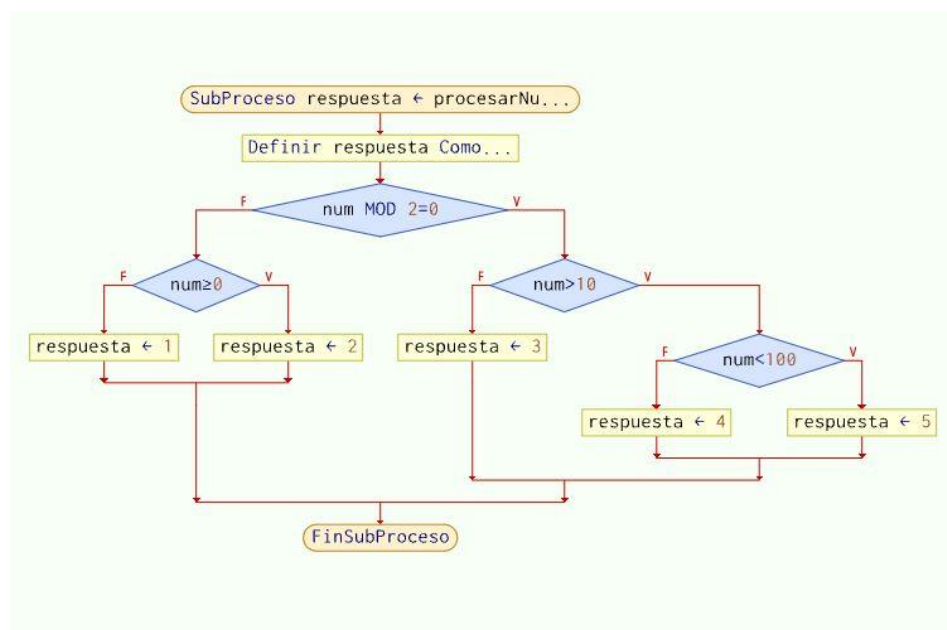
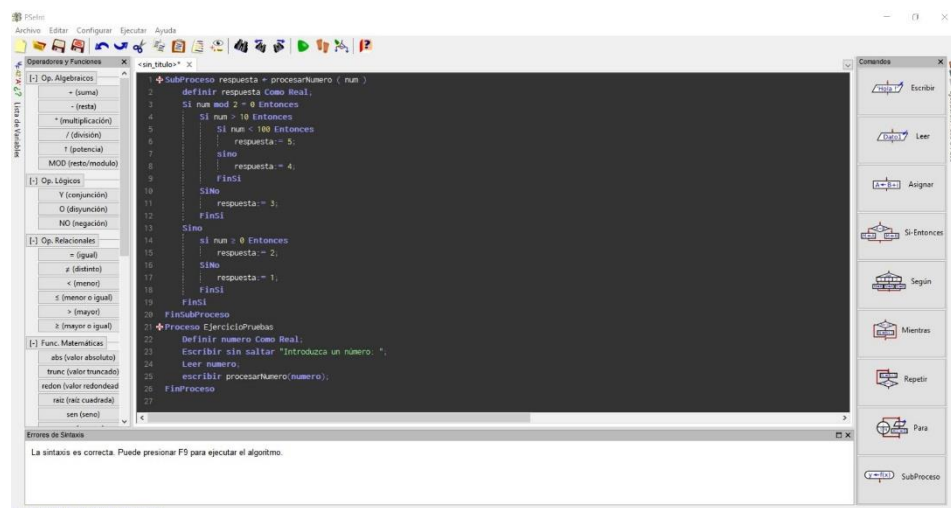
Introducción

El objetivo de esta práctica es conocer si el alumno ha adquirido los conocimientos necesarios para definir y realizar unos tests de pruebas a partir de un código dado, con la resolución de los ejercicios se van trabajando los distintos apartados de la Unidad, y se realiza todo el camino que finaliza en la creación de unos test de pruebas efectivos para el código propuesto, estos test de pruebas analizarán cada nodo y arista del grafo, así como los valores límite. Esta actividad sirve como recordatorio de la Unidad 2 y el uso de PSeInt y la transformación del pseudocódigo a JAVA, y se refuerzan conceptos de la asignatura de programación. También se trabaja con Junit que será el programa tester que utilizaremos para probar el código.

Ejercicio 1

Creación de un diagrama de flujo:

He replicado el Código en PSeInt y he seleccionado la opción “Dibujar Diagrama de Flujo” sobre el subproceso, esto me ha generado automáticamente el diagrama de flujo que vemos a continuación, en el se pueden apreciar lo que serán los nodos predicado y las posibles respuestas de Verdadero o Falso que genera cada uno.



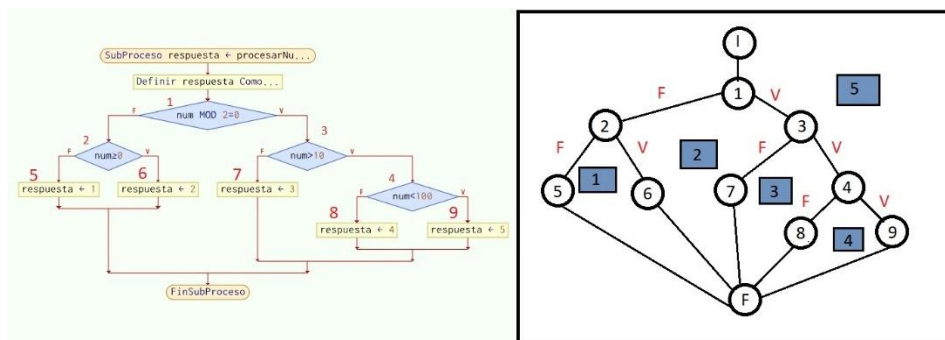
Ejercicio 2

Creación de un grafo:

Sobre el diagrama anterior he resaltado los nodos que tendrá nuestro grafo asignando un número a cada uno.

Si contamos los nodos de Inicio y Fin, los nodos predicado que son las compuertas que generan dos caminos y los resultados, me salen 11 nodos distintos; al trazar las aristas que unen dicho nodos me aparecen 14 aristas distintas.

A continuación, se muestra el grafo que ha resultado:



Ejercicio 3

Complejidad ciclomática:

La complejidad ciclomática $V(G)$ puede calcularse de varias formas:

1. $V(G)$: Número predado + 1.

Los nodos predados son CUATRO, el 1,2, 3 Y 4. Por lo que $4+1 = 5$.

2. $V(G)$: Aristas - Nodos + 2.

Tenemos 14 aristas y 11 nodos. Por lo que $14-11+2 = 5$.

3. $V(G)$: Número de regiones.

Según el grafo tenemos 5 regiones incluyendo la exterior.

En todos los casos la complejidad ciclomática es de 5.

Ejercicio 4

Caminos independientes:

A partir de la complejidad ciclomática, He creado una tabla con los cinco caminos independientes tiene este código y que deberán probarse.

Camino	Entrada	Salida
I -> 1 -> 2 -> 5 -> F	num es -1	Respuesta es 1
I -> 1 -> 2 -> 6 -> F	num es 1	Respuesta es 2
I -> 1 -> 3 -> 7 -> F	num es 10	Respuesta es 3
I -> 1 -> 3 -> 4 -> 8 -> F	num es 100	Respuesta es 4
I -> 1 -> 3 -> 4 -> 9 -> F	num es 12	Respuesta es 5

Ejercicio 5

Código Java:

Desde PSeInt se ha exportado el pseudocódigo como código Java con la opción “Archivo> Exportar> Convertir a código Java.”. sobre el código que nos ha creado PSeInt se ha intervenido para que utilice números enteros en lugar de doble, se han retirado las referencias estáticas y se ha creado un objeto de la clase principal. A continuación copio el código resultante:

```
import java.io.*;

public class ejerciciopruebas {

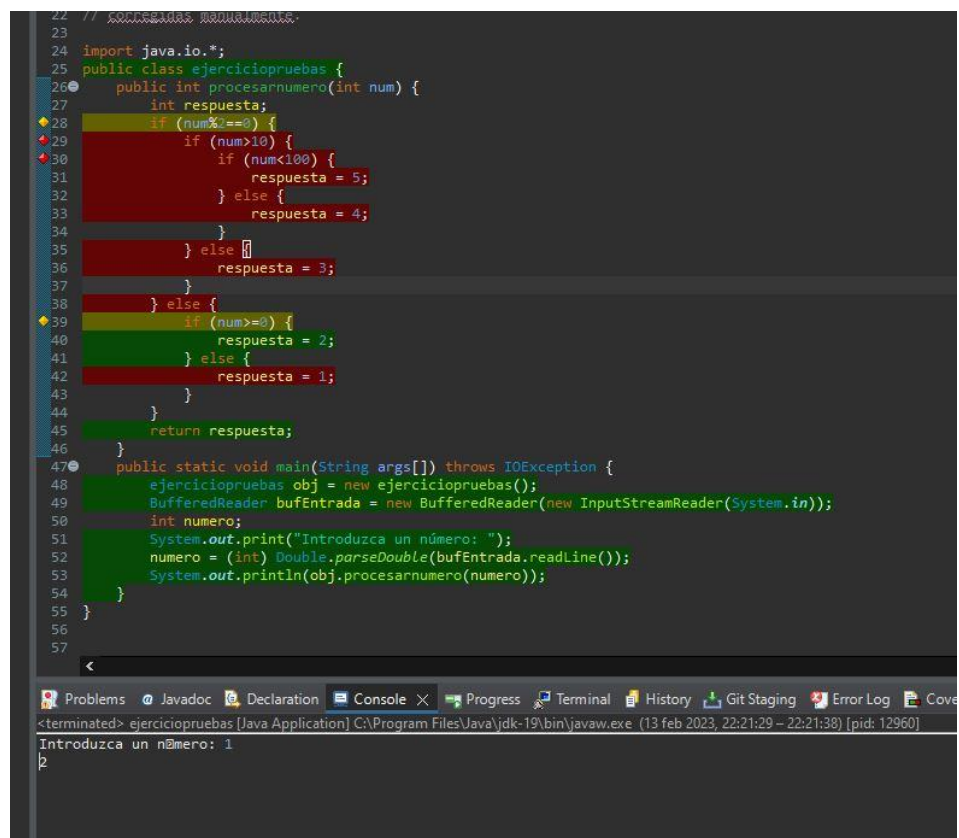
    public int procesarnumero(int num) {
        int respuesta;
        if (num%2==0) {
            if (num>10) {
                if (num<100) {
                    respuesta = 5;
                } else {
                    respuesta = 4;
                }
            } else {
                respuesta = 3;
            }
        } else {
            if (num>=0) {
                respuesta = 2;
            } else {
                respuesta = 1;
            }
        }
        return respuesta;
    }

    public static void main(String args[]) throws IOException {
        ejerciciopruebas obj = new ejerciciopruebas();
        BufferedReader bufEntrada = new BufferedReader(new
InputStreamReader(System.in));
        int numero;
        System.out.print("Introduzca un número: ");
        numero = (int) Double.parseDouble(bufEntrada.readLine());
        System.out.println(obj.procesarnumero(numero));
    }
}
```

Ejercicio 6

Coverage:

En este ejercicio se ha ejecutado el código con la opción Coverage para poder analizar cada uno de los caminos que tiene este código, a continuación procedo a mostrar los pantallazos con los resultados obtenidos y una breve descripción de cada uno:



```
22 // Comentarios adicionales.
23
24 import java.io.*;
25 public class ejerciciopruebas {
26     public int procesarnumero(int num) {
27         int respuesta;
28         if (num%2==0) {
29             if (num>10) {
30                 if (num<100) {
31                     respuesta = 5;
32                 } else {
33                     respuesta = 4;
34                 }
35             } else {
36                 respuesta = 3;
37             }
38         } else {
39             if (num>=0) {
40                 respuesta = 2;
41             } else {
42                 respuesta = 1;
43             }
44         }
45         return respuesta;
46     }
47     public static void main(String args[]) throws IOException {
48         ejerciciopruebas obj = new ejerciciopruebas();
49         BufferedReader bufEntrada = new BufferedReader(new InputStreamReader(System.in));
50         int numero;
51         System.out.print("Introduzca un número: ");
52         numero = (int) Double.parseDouble(bufEntrada.readLine());
53         System.out.println(obj.procesarnumero(numero));
54     }
55 }
56
57
```

Problems Javadoc Declaration Console X Progress Terminal History Git Staging Error Log Coverage

<terminated> ejerciciopruebas [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (13 feb 2023, 22:21:29 - 22:21:38) [pid: 12960]

Introduzca un número: 1

2

Probando con el número 1, nos muestra como resultado un 2, esto quiere decir que el programa ha tomado el segundo camino posible de nuestra lista de caminos independientes en el ejercicio 4. La primera condición (if) comprueba si el número es par, al no cumplirse pasaríamos al denominado como el nodo 2 de nuestro grafo del ejercicio 2. Que comprueba si el número es mayor o igual a 0, en este caso la respuesta es cierta, así que sale por el que hemos determinado nodo 6, Respuesta = 2.


```
23
24 import java.io.*;
25 public class ejerciciopruebas {
26     public int procesarnumero(int num) {
27         int respuesta;
28         if (num%2==0) {
29             if (num>10) {
30                 if (num<100) {
31                     respuesta = 5;
32                 } else {
33                     respuesta = 4;
34                 }
35             } else {
36                 respuesta = 3;
37             }
38         } else {
39             if (num>=0) {
40                 respuesta = 2;
41             } else {
42                 respuesta = 1;
43             }
44         }
45         return respuesta;
46     }
47     public static void main(String args[]) throws IOException {
48         ejerciciopruebas obj = new ejerciciopruebas();
49         BufferedReader bufEntrada = new BufferedReader(new InputStreamReader(System.in));
50         int numero;
51         System.out.print("Introduzca un número: ");
52         numero = (int) Double.parseDouble(bufEntrada.readLine());
53         System.out.println(obj.procesarnumero(numero));
54     }
55 }
56
57
```

Problems Javadoc Declaration Console X Progress Terminal History Git Staging Error Log C

<terminated> ejerciciopruebas [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (13 feb 2023, 22:20:59 – 22:21:03) [pid: 10252]

Introduzca un número: -1

1

Probando con el número -1, nos muestra como resultado un 1, esto quiere decir que el programa ha tomado el primer camino posible de nuestra lista de caminos independientes en el ejercicio 4. La primera condición (if) comprueba si el número es par, al no cumplirse pasaríamos al denominado como el nodo 2 de nuestro grafo del ejercicio 2. Que comprueba si el número es mayor o igual a 0, en este caso la respuesta es falsa, así que sale por el que hemos determinado nodo 5, Respuesta = 1.

```
23
24 import java.io.*;
25 public class ejerciciopruebas {
26     public int procesarnumero(int num) {
27         int respuesta;
28         if (num%2==0) {
29             if (num>10) {
30                 if (num<100) {
31                     respuesta = 5;
32                 } else {
33                     respuesta = 4;
34                 }
35             } else {
36                 respuesta = 3;
37             }
38         } else {
39             if (num>=0) {
40                 respuesta = 2;
41             } else {
42                 respuesta = 1;
43             }
44         }
45         return respuesta;
46     }
47     public static void main(String args[]) throws IOException {
48         ejerciciopruebas obj = new ejerciciopruebas();
49         BufferedReader bufEntrada = new BufferedReader(new InputStreamReader(System.in));
50         int numero;
51         System.out.print("Introduzca un número: ");
52         numero = (int) Double.parseDouble(bufEntrada.readLine());
53         System.out.println(obj.procesarnumero(numero));
54     }
55 }
56
57
```

Problems Javadoc Declaration Console X Progress Terminal History Git Staging Error Log

<terminated> ejerciciopruebas [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (13 feb 2023, 22:22:00 – 22:22:02) [pid: 18652]

Introduzca un número: 10

3

Probando con el número 10, nos muestra como resultado un 3, esto quiere decir que el programa ha tomado el tercer camino posible de nuestra lista de caminos independientes en el ejercicio 4. La primera condición (if) comprueba si el número es par, al sí cumplirse esta vez pasaríamos al denominado como el nodo 3 de nuestro grafo del ejercicio 2. Que comprueba si el número es mayor que 10, en este caso la respuesta es falsa, así que sale por el que hemos determinado nodo 7, Respuesta = 3.

```
23
24 import java.io.*;
25 public class ejerciciopruebas {
26     public int procesarnumero(int num) {
27         int respuesta;
28         if (num%2==0) {
29             if (num>10) {
30                 if (num<100) {
31                     respuesta = 5;
32                 } else {
33                     respuesta = 4;
34                 }
35             } else {
36                 respuesta = 3;
37             }
38         } else {
39             if (num>=0) {
40                 respuesta = 2;
41             } else {
42                 respuesta = 1;
43             }
44         }
45         return respuesta;
46     }
47     public static void main(String args[]) throws IOException {
48         ejerciciopruebas obj = new ejerciciopruebas();
49         BufferedReader bufEntrada = new BufferedReader(new InputStreamReader(System.in));
50         int numero;
51         System.out.print("Introduzca un número: ");
52         numero = (int) Double.parseDouble(bufEntrada.readLine());
53         System.out.println(obj.procesarnumero(numero));
54     }
55 }
56
57
```

Problems Javadoc Declaration Console X Progress Terminal History Git Staging Error Log

<terminated> ejerciciopruebas [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (13 feb 2023, 22:22:56 – 22:23:00) [pid: 18864]

Introduzca un número: 12

5

Probando con el número 12, nos muestra como resultado un 5, esto quiere decir que el programa ha tomado el quinto camino posible de nuestra lista de caminos independientes en el ejercicio 4. La primera condición (if) comprueba si el número es par, al cumplirse la condición, pasaríamos al denominado como el nodo 3 de nuestro grafo del ejercicio 2. Que comprueba si el número es mayor que 10, en este caso la respuesta es cierta, así que pasamos al denominado nodo 4 que comprueba si el número es menor que 100. Al considerarse esta afirmación como cierta, el flujo sale por el que hemos determinado nodo 9, Respuesta = 5.

```
23
24 import java.io.*;
25 public class ejerciciopruebas {
26     public int procesarnumero(int num) {
27         int respuesta;
28         if (num%2==0) {
29             if (num>10) {
30                 if (num<100) {
31                     respuesta = 5;
32                 } else {
33                     respuesta = 4;
34                 }
35             } else {
36                 respuesta = 3;
37             }
38         } else {
39             if (num>=0) {
40                 respuesta = 2;
41             } else {
42                 respuesta = 1;
43             }
44         }
45         return respuesta;
46     }
47     public static void main(String args[]) throws IOException {
48         ejerciciopruebas obj = new ejerciciopruebas();
49         BufferedReader bufEntrada = new BufferedReader(new InputStreamReader(System.in));
50         int numero;
51         System.out.print("Introduzca un número: ");
52         numero = (int) Double.parseDouble(bufEntrada.readLine());
53         System.out.println(obj.procesarnumero(numero));
54     }
55 }
56
57
```

Problems Javadoc Declaration Console X Progress Terminal History Git Staging Error Log

<terminated> ejerciciopruebas [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (13 feb 2023, 22:22:30 – 22:22:33) [pid: 70]

Introduzca un número: 100

4

Probando con el número 100, nos muestra como resultado un 4, esto quiere decir que el programa ha tomado el quinto camino posible de nuestra lista de caminos independientes en el ejercicio 4. La primera condición (if) comprueba si el número es par, al cumplirse la condición, pasaríamos al denominado como el nodo 3 de nuestro grafo del ejercicio 2. Que comprueba si el número es mayor que 10, en este caso la respuesta es cierta, así que pasamos al denominado nodo 4 que comprueba si el número es menor que 100. Al considerarse esta afirmación como falsa, el flujo sale por el que hemos determinado nodo 8, Respuesta = 4.

Ejercicio 7

Clases de equivalencia:

A partir de este ejercicio se me ha ido haciendo más difícil entender exactamente qué datos se estaban solicitando, he visto varias tablas de equivalencia en los textos de la unidad. Pero nada tenían que ver con las tablas expuestas en el video de ejemplo de la misma así que he optado por hacer un mix y crear unas tablas que tuvieran las clases de equivalencia que he entendido que debían aparecer. A continuación muestro las tablas y adjunto una breve descripción:

Atributo respuesta -NODO1				
Condicion de Entrada	Clases de equivalencia	Clases Válidas	Clases No Validas	Paso a Nodo
Numeros enteros	VALOR PAR	Cualque valor entero par	Número Impar Cadena	Nodo 3
	VALOR IMPAR	Cualque valor entero impar	Número Par Cadena	Nodo 2

Atributo respuesta - NODO 2					
Condicion de Entrada	Tipo	Clase	Limite inferior	Limite superior	Paso a Nodo
Numeros enteros	VALIDA	Valores Impares \geq a 0	1	MAXVALUE	FIN
	NO VALIDA	Valores Impares $<$ que 0	MINVALUE -1	-1	FIN

Atributo: Respuesta - NODO 3					
Condicion de Entrada	Tipo	Clase	Limite inferior	Limite superior	Paso a Nodo
Numeros enteros	VALIDA	Valores Pares > a 10	12	MAXVALUE - 1	Nodo 4
	NO VALIDA	Valores Pares <= que 10	MIN VALUE	10	FIN

Atributo: Respuesta - NODO 4					
Condicion de Entrada	Tipo	Clase	Limite inferior	Limite superior	Paso a Nodo
Numeros enteros	VALIDA	Valores Pares > 10 y < 100	12	98	FIN
	NO VALIDA	Valores Pares > = a 100	100	MAX VALUE -1	FIN

Lo que acabamos de ver son las cuatro tablas de equivalencia de cada uno de los nodos predicados dónde se ve la clase de equivalencia, con clases válidas e inválidas, los limites superior e inferior de cada nodo y el paso que desencadena en el flujo dicha clase.

Ejercicio 8

Análisis de los valores límite:

	Condiciones de Entrada y salida	Casos de Prueba
Nodo 1	Numero par o impar	1,2
Nodo 2	Valores impares ≥ 0	1,maxvalue, minvalue-1, -1
Nodo 3	Valores Pares >10	12, Maxvalue-1, min value, 10
Nodo 4	Valores Pares < 100	12,98,100, MAXVALUE-1

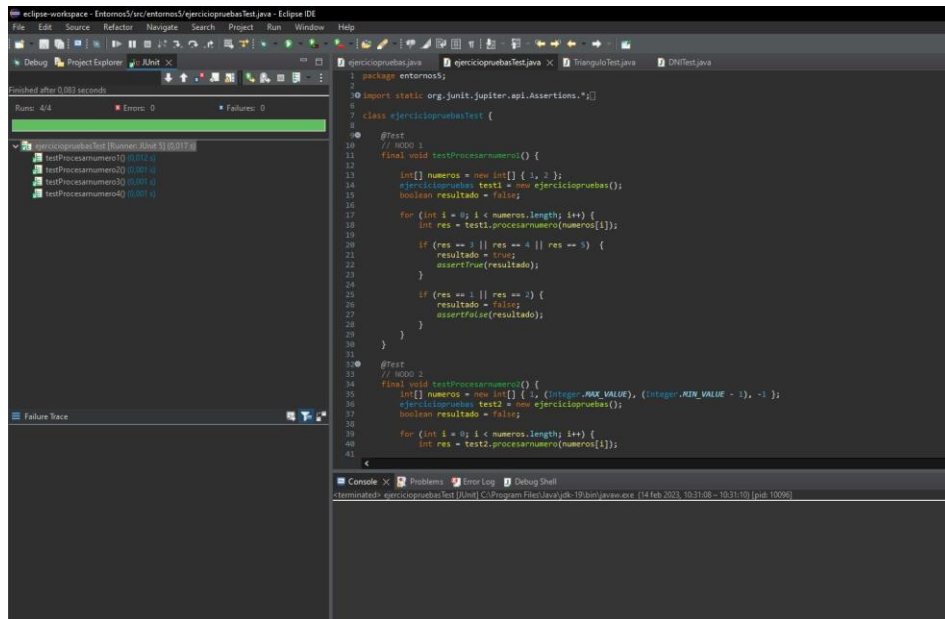
Esto que se muestra serían los valores límites que pienso que se deben probar para asegurarnos de que las condiciones del código están bien definidas y los operandos son correctos. Cómo se puede apreciar, distingo entre los distintos nodos predicados del grafo en todo momento ya que los valores límites cambian en función de este dato.

Ejercicio 9 y 10

JUnit Test Case y Pruebas parametrizadas:

Para estos ejercicios no tenía muy claro la diferencia así que he creado una clase de Test de pruebas utilizando la herramientas JUnit, estas pruebas ejecutan todas las instrucciones y prueban todos los valores límites.

A continuación, se incluye el código y un pantallazo de las pruebas realizadas.



```
package entornos5;
import static org.junit.jupiter.api.Assertions.*;
class ejerciciopruebasTest {
    @Test
    // NODO 1

    final void testProcesarnumero1() {

        int[] numeros = new int[] { 1, 2 };
        ejerciciopruebas test1 = new ejerciciopruebas();
        boolean resultado = false;

        for (int i = 0; i < numeros.length; i++) {
            int res = test1.procesarnumero(numeros[i]);

            if (res == 3 || res == 4 || res == 5) {
                resultado = true;
                assertTrue(resultado);
            }

            if (res == 1 || res == 2) {
                resultado = true;
                assertTrue(resultado);
            }
        }
    }
}
```



```

@Test // NOD02
    final void testProcesarnumero2() {
        int[] numeros = new int[] { 1, (Integer.MAX_VALUE),
(Integer.MIN_VALUE - 1), -1 };
        ejerciciopruebas test2 = new ejerciciopruebas();
        boolean resultado = false;

        for (int i = 0; i < numeros.length; i++) {
            int res = test2.procesarnumero(numeros[i]);

            if (res == 2) {
                resultado = true;
                assertTrue(resultado);
            }

            if (res == 1) {
                resultado = true;
                assertTrue(resultado);
            }

            if (res == 3 || res == 4 || res == 5) {
                resultado = false;
                assertTrue(resultado);
            }
        }
    }

@Test
    // NOD03
    final void testProcesarnumero3() {
        int[] numeros = new int[] { 12, (Integer.MAX_VALUE
- 1), (Integer.MIN_VALUE), 10 };
        ejerciciopruebas test3 = new ejerciciopruebas();
        boolean resultado = false;

        for (int i = 0; i < numeros.length; i++) {
            int res = test3.procesarnumero(numeros[i]);

            if (res == 4 || res == 5) {
                resultado = true;
                assertTrue(resultado);
            }

            if (res == 3) {
                resultado = false;
                assertFalse(resultado);
            }

            if (res == 1 || res == 2) {
                resultado = false;
                assertTrue(resultado);
            }
        }
    }

```

```

@Test
    // NODO 4
    final void testProcesarnumero4() {
        int[] numeros = new int[] { 12, (Integer.MAX_VALUE-
1), 98, 100 };
        ejerciciopruebas test4 = new ejerciciopruebas();
        boolean resultado = false;

        for (int i = 0; i < numeros.length; i++) {
            int res = test4.procesarnumero(numeros[i]);

            if (res == 5) {
                resultado = true;
                assertTrue(resultado);
            }

            if (res == 4) {
                resultado = false;
                assertFalse(resultado);
            }

            if (res == 1 || res == 2 || res == 3) {
                resultado = false;
                assertTrue(resultado);
            }
        }
    }
}

```

Conclusiones

Esta práctica ha servido para aprender a diseñar pruebas en JAVA y JUnit. El objetivo principal era comprender todos los pasos necesarios para llevar a cabo un diseño de pruebas eficiente. Comenzamos con la creación de un diagrama de flujo que sirvió como referencia para entender el alcance de la práctica. A continuación, se creó un grafo para entender la relación entre los diferentes puntos del diagrama. También se calculó la complejidad ciclomática para evaluar la complejidad del sistema. Se encontraron caminos independientes que se utilizaron para testear el sistema. Posteriormente, se escribió código Java para implementar el diseño de pruebas. Se calculó el coverage para determinar el grado de cobertura del código. Se crearon clases de equivalencia para asegurar que el código funcionara de forma correcta. Se realizó un análisis de los valores límite para comprobar el comportamiento del sistema para los límites superiores e inferiores. Se crearon JUnit Test Case para probar el código. Y finalmente, se realizaron pruebas parametrizadas para asegurar que el código se comportara de forma correcta para los diferentes conjuntos de datos.

En conclusión, esta práctica ha sido una excelente herramienta educativa para comprender los conceptos de diseño de pruebas con JAVA y JUnit. Hemos aprendido a crear diagramas de flujo, grafos, a calcular la complejidad ciclomática, a encontrar caminos independientes, a escribir código Java, a calcular el coverage, a crear clases de equivalencia, a realizar análisis de los valores límite, a crear JUnit Test Case y a realizar pruebas parametrizadas. Esto nos ha permitido adentrarnos en el mundo del diseño de pruebas en JAVA y JUnit, conociendo todos los pasos que hay que seguir para realizarlas de forma correcta.