IES Virrey Morcillo. (Nombre Ciclo Formativo)

Actividades de Ampliación 1 y 2 del Tema3 de Entornos

Módulo: Entornos de desarrollo

Nombre del Alumno: Santiago Alarcón

Fecha: 7/3/2021

Contenido

Criterios de Evaluación

- CE.x.x) ...

Material

Haremos uso de los siguientes recursos para realizar las tareas:

- Ordenador
- Eclipse IDE
- Excel
- app.diagram.es
- Procesador de Textos Pages

Objetivos

- Conocer ...
- Repasar ...

1. Introducción

Sobre la función java enunciado 1:

```
static int Contador2(int x, int y) {
  Scanner entrada = new Scanner(System.in);
  int num, c = 0;
  if (  x > 0   &&   y > 0) {
     System.out.println("Escribe un número");
     num = entrada.nextInt();
     while (num != 0) {
           if (  num >= x  &&   num <= y ) {
                 System.out.println("\tNúmero en el rango");
                 c++;
           } else
                 System.out.println("\tNúmero fuera de rango");
           System.out.println("Escribe un número");
           num = entrada.nextInt();
     }//fin while
  }
  else
     c = -1;
  entrada.close();
  return c;
}//
```

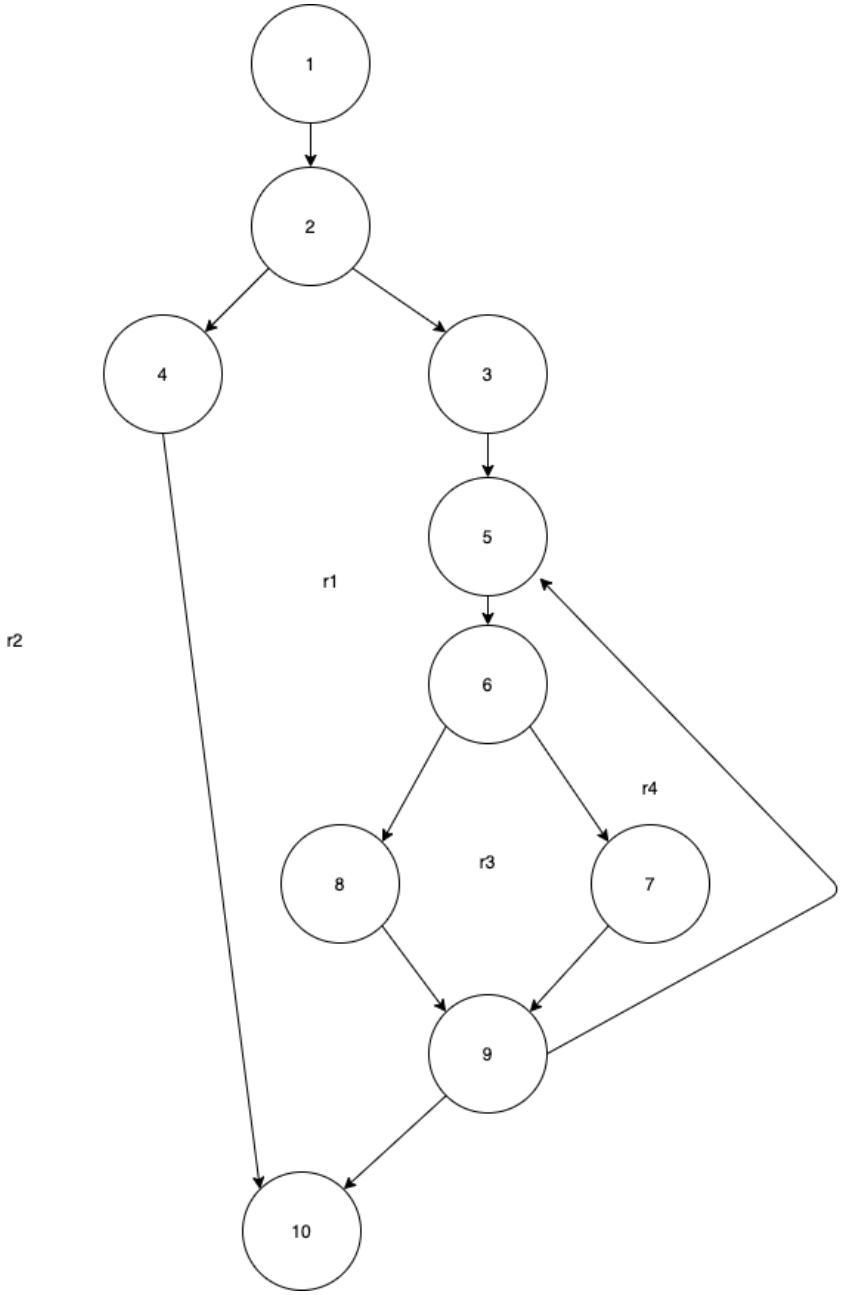
Función java enunciado 2:

```
public class TablaEnteros {
  private Integer[] tabla;

  TablaEnteros(Integer[] tabla) {
    if (tabla == null || tabla.length == 0)
      throw new IllegalArgumentException("No hay elementos");
    this.tabla = tabla;
  }

  //devuelve la suma de los elementos de la tabla
  public int sumaTabla() {
    int suma = 0;
    for (int i = 0; i < tabla.length; i++)
      suma += tabla[i];
    return suma;
  }

  //devuelve el mayor elemento de la tabla
  public int mayorTabla() {
    int max = -999;
    for (int i = 0; i < tabla.length; i++)
      if (tabla[i] > max)
          max = tabla[i];
    return max;
  }

  //devuelve la posición de un elemento cuyo valor se pasa
  public int posicionTabla(int n) {
    for (int i = 0; i < tabla.length; i++)
      if (tabla[i] == n)
          return i;
    throw new java.util.NoSuchElementException("No existe:" + n);
  }
}//
```

2. Tareas

Sobre enunciado 1) Realiza el grafo de flujo, calcula la complejidad ciclomátíca, define el conjunto básico de caminos, elabora los casos de prueba para cada camino y evalúa el riesgo para la siguiente función Java:
Solución:

Grafo de flujo:



Cálculo de la complejidad ciclomática:

NODOS	COMANDOS	
1	Scanner Entrada	
	Declaración de variables	
2	if	nodo predicado
3	System.out.println	
	num	
4	c=1	
5	while	nodo predicado
6	if	nodo predicado
7	System.out.println	
	c++	
8	System.out.println	
9	System.out.println	
	num=entrada.nextInt()	
10	entrada.close	
	return c	

Número regiones = 4
Número aristas - número nodos + 2 = 12 - 10 + 2 = 4
Número nodos predicado + 1 = 4

Riesgo para esta complejidad ciclomática: Programas o métodos sencillos sin mucho riesgo

Conjunto básico de caminos:

Camino 1—> 1-2-4-10
Camino 2 —> 1-2-3-5-6-7-9-10
Camino 3 —> 1-2-3-5-6-8-9-10
Camino 4 —> 1-2-3-5-6-7-9-5-6-8-9-10

Casos de prueba para camino:

CAMINO	CASO DE PRUEBA	RESULTADO ESPERADO
1	x<=0	Visualiza c= -1
	y<=0	
	num=	
2	x>0	Número en el rango
	y>0	Visualiza c>0
	num>x num<y	
3	x>0	número fuera de rango
	y>0	c= 0
	num<x || num>y	
4	x>0	
	y>0	
	num<x || num>y	c=0
	num>x num<y	c>1
	num = 0	
		Visualiza: c > 1

Sobre enunciado 2) Escribe una clase de pruebas para probar los métodos de la clase TablaEnteros. En esta clase de prueba crea un método con la anotación @BeforeClass en que inicialices array de enteros para usarlo en las pruebas de los métodos. El método ,vumaTuh/a() suma los elementos del array y devuelve la suma. El método ,nayorTabla() devuelve elemento mayor de la tabla. Y método posicionTabla() devuelve la posición ocupada por elemento cuyo valor se envía. En constructor se comprueba número de elementos de la tabla es nulo en este caso se lanza excepción IllegalArgumeníException con mensaje No hay elementos. método posiciónTabla también lanza excepción, NoSuchEleinentException. en caso de que no se encuentre elemento en la tabla. Hay que añadir otros dos métodos de prueba para probar estas excepciones.
Solución:

```
import static org.juunit.Assert.*;
import org.junit.Test;
import java.util.Scanner;
import java.lang.*;

public class TablaEnterosTest {
private Integer[] tabla;
```

```
private int longitud_tabla;
private int[][] comprobatio;
Scanner dime = new Scanner(System.in);

@BeforeClass
public int[] llenar(int longitud_tabla){
	integer[] tabla = new int[longitud_tabla];
	//para controlar las posiciones he creado una matriz de dos dimensiones
	int comprobatio[][]= new int[longitud_tabla][2];
	int sumatorio = 0;
	mayor = -999;

	//lleno el array con números distintos de 0
	for (int i = 0; i<tabla.length; i++){

		do{
		tabla[i] = dime.nextInt();
		i—;
		}while (tabla[i] = 0);

		sumatorio =+ tabla[i];
		mayor = Math.max(mayor, tabla[i]);

			for(int j= 0; j<2;j++){
			comprobatio[i][0]=tabla[i];
			comprobatio[i][1]=i;
			}
	return tabla;
	}

@Test
public void TablaEnterosTest() {
	try {
		longitud_tabla = 0;
		tabla = new llenar(longitud_tabla);
		int obviedad = tabla.TablaEnteros();
		fail (“FALLO, Debería haber lanzado la excepción”);
	} catch (IllegalArgumentException) {
	//PRUEBA SATISFACTORIA
	}
}

@Test
public void sumaTablaTest() {
	x = 4;
	tabla = new llenar(x);
	int resultado_suma = tabla.sumaTabla();
	assertEquals(sumatorio, resultado_suma);
}

@Test
public void MayorTablaTest(){
```

```
        x= 4;
        tabla = new llenar(x);
        int maximo = tabla.mayorTabla()
        assertEquals(mayor,maximo);
}
@Test
public int posicionTablaTest(){
        tabla = new llenar(4);
        int k = comprobatio [3][0];
        posicion_correcta = tabla.posicionTabla(k);
        assertEquals(comprobatio[3][1], posicion_correcta);
}

@Test // falla pues he obligado a que no se meta el número 0
public void posicionTablaTest(){
        try {
                tabla = new llenar(4);
                int k = 0;
                int obviedad =new posicionTabla(k)
                fail ("FALLO, debiere haber lanzado la excepción");
        } catch (java.utilNoSuchelementException){
        //PRUEBA SATISFACTORIA
        }
}
```

3. Conclusión

Aunque una aproximación es en el before de cada prueba pedir los números para rellenar el array, esto no es necesario, se podría dar un array dentro de cada test con unos valores ya prefijados (igual que se da la cantidad de casillas, y hariamos lo mismo para el test de posición tabla test, esto es, asignariamos una matriz directamente desde la prueba para ejecutar posiciónTabla() para ese array y se comprobaría el resultado.

4. Links

Indicamos los sitios web visitados para poder realizar las actividades.