PECL2. INGENIERÍA DEL SOFTWARE

EJERCICIO 2

GRADO DE INGENIERÍA INFORMÁTICA

Juan Casado Ballestero	09108762A
Miguel Ángel Losada Fernández	53824672A
Laura Pérez Medeiro	03211038P

EJERCICIO 2

Calculamos la complejidad ciclomática y obtener los caminos de los métodos de la clase Java que se adjunta al final.

GRAFO DE FLUJO. El programa es mayoritariamente secuencial, las instrucciones de la 1 a la 37 se realiza de esta forma. Durante esta primera parte se hace un setUp del lienzo y del JFrame sobre los que posteriormente se dibujará, una característica principal de esta fase son las llamadas a las clases externas que se están configurando. Esto se realiza a lo largo de cuatro funciones (contando el main). Cabe destacar la existencia de un método adicional, el constructor, el cual está vacío y de otro método que no llega a llamarse nunca (init).

En la instrucción 37 iniciamos un hilo nuevo, el cual sustituirá al main que se deja finalizar sin ejecutar más instrucciones. El nuevo hilo que toma la ejecución contiene un bucle while que encapsula a un bucle for, la misión de dichos bucles es generar nuevos puntos aleatorios en el lienzo de forma que este se actualice cada segundo. La única forma de salir de dichos bucles, sería con una interrupción del hilo cuando esté haciendo sleep, se produciría una Interrupted Exception que se capturaría por el catch externo al bucle while de forma que el hilo finalizaría su ejecución.

Es imposible salir del bucle while con un cambio en la condición que evalúa, ya que ésta será siempre true. No obstante, para generar el diagrama de flujo y el recuento de caminos se considerará que si que existe la posibilidad de salida por cambio de condición.

El correspondiente grafo de flujo asociado al código se corresponde a la Figura 1.

COMPLEJIDAD CICLOMÁTICA. La calcularemos a partir de la siguiente fórmula:

$$v(g) = aristas - nodos + 2$$

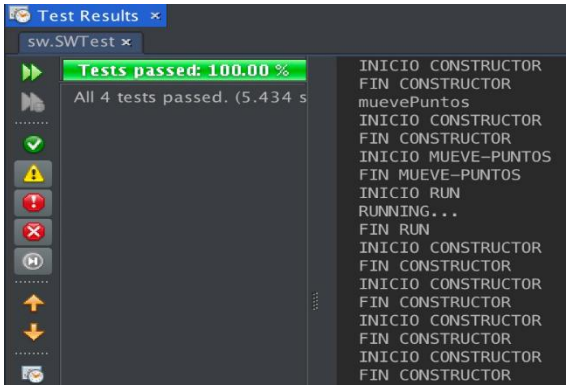
Por lo tanto, la complejidad ciclomática obtenida es de **4.**

CAMINOS INDEPENDIENTES. Tenemos dos posibles caminos:

- El primero sería el que se recorrería en caso de que se produjera alguna excepción y sería: {INICIO,1,2,3,4,5,6,7,…,37,38,39,40,39,41,42,38,39,41,42,43,FIN}
- El segundo será el camino que se ejecute cuando la condición dentro del while fuera false: {INICIO,1,2,...,37,38,FIN}

Recorridos estos dos caminos quedarían visitadas todas las aristas del grafo.

PRUEBAS REALIZADAS. Hemos realizado pruebas unitarias mediante el uso de Junit para comprobar el correcto funcionamiento de cada una de las funciones de la clase.



Además, para comprobar y facilitarnos el seguimiento del código a través de los posibles caminos hemos introducido una serie de salidas (mediante system.out.println) al inicio y al final de cada método y posible bucle. De esta manera trazamos el recorrido de ejecución del código y descubrimos que la función init no es utilizada en ningún momento.

```
INICIO MAIN
INICIO CONSTRUCTOR
FIN CONSTRUCTOR
INICIO VISUALIZA-FRAME
INICIO INICIALIZA-LIENZO
FIN INICIALIZA-LIENZO
FIN VISUALIZA-FRAME
INICIO MUEVE-PUNTOS
FIN MUEVE-PUNTOS
FIN MAIN
INICIO RUN
RUNNING...
RUNNING...
RUNNING...
RUNNING...
```

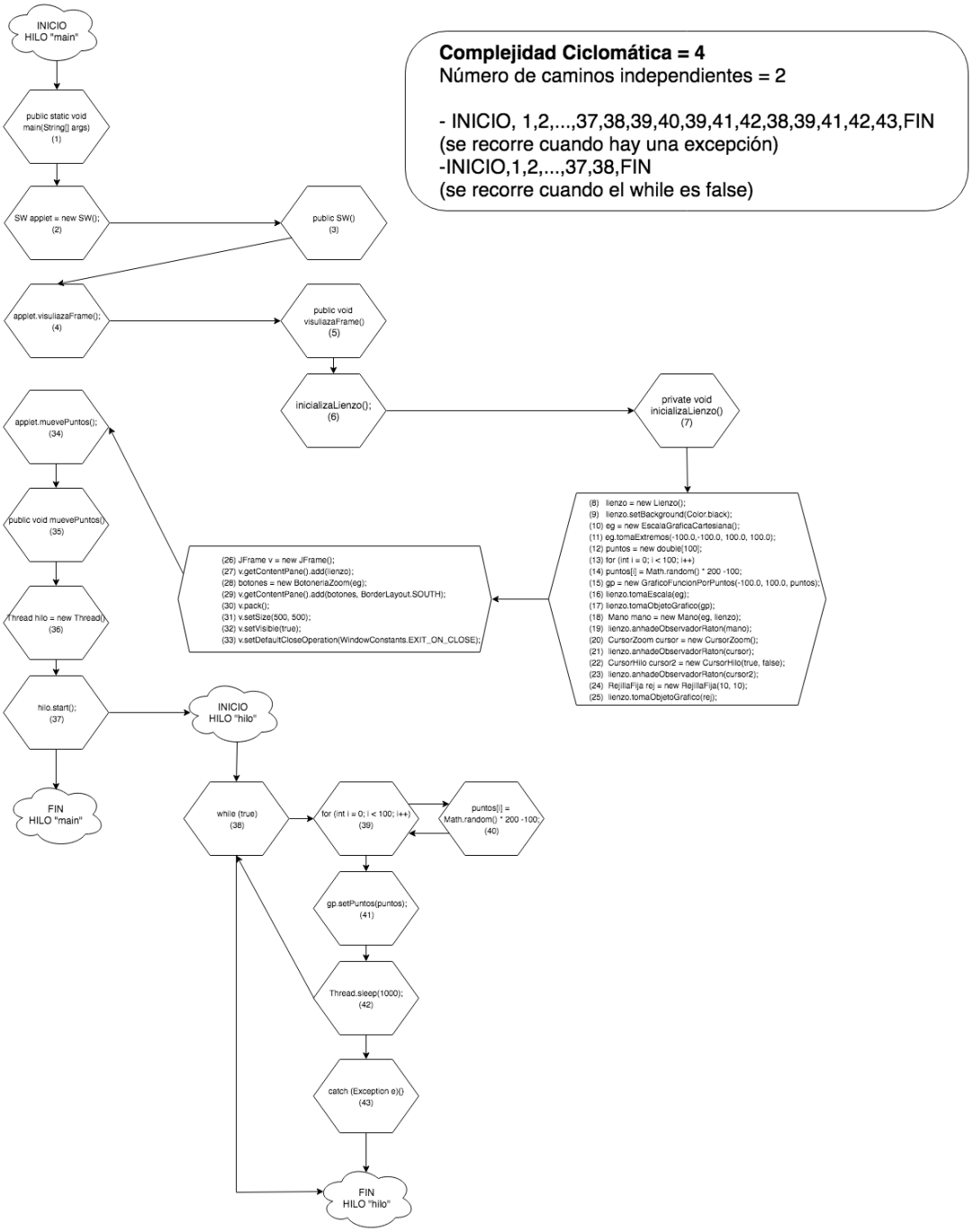

INICIO
HILO "main"
public static void
main(String[] args)
(1)
SW applet = new SW();
(2)
public SW()
(3)
applet.visualizaFrame();
(4)
public void
visualizaFrame()
(5)
inicializaLienzo();
(6)
private void
inicializaLienzo()
(7)
(8) lienzo = new Lienzo();
(9) lienzo.setBackground(Color.black);
(10) eg = new EscalaGraficaCartesiana();
(11) eg.tomaExtremos(-100.0,-100.0, 100.0, 100.0);
(12) puntos = new double[100];
(13) for (int i = 0; i < 100; i++)
(14) puntos[i] = Math.random() * 200 -100;
(15) gp = new GraficoFuncionPorPuntos(-100.0, 100.0, puntos);
(16) lienzo.tomaEscala(eg);
(17) lienzo.tomaObjetoGrafico(gp);
(18) Mano mano = new Mano(eg, lienzo);
(19) lienzo.anhadeObservadorRaton(mano);
(20) CursorZoom cursor = new CursorZoom();
(21) lienzo.anhadeObservadorRaton(cursor);
(22) CursorHilo cursor2 = new CursorHilo(true, false);
(23) lienzo.anhadeObservadorRaton(cursor2);
(24) RejillaFija rej = new RejillaFija(10, 10);
(25) lienzo.tomaObjetoGrafico(rej);
(26) JFrame v = new JFrame();
(27) v.getContentPane().add(lienzo);
(28) botones = new BotoneriaZoom(eg);
(29) v.getContentPane().add(botones, BorderLayout.SOUTH);
(30) v.pack();
(31) v.setSize(500, 500);
(32) v.setVisible(true);
(33) v.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
applet.muevePuntos();
(34)
public void muevePuntos()
(35)
Thread hilo = new Thread();
(36)
hilo.start();
(37)
FIN
HILO "main"
INICIO
HILO "hilo"
while (true)
(38)
for (int i = 0; i < 100; i++)
(39)
puntos[i] =
Math.random() * 200 -100;
(40)
gp.setPuntos(puntos);
(41)
Thread.sleep(1000);
(42)
catch (Exception e){}
(43)
FIN
HILO "hilo"
Complejidad Ciclomática = 4
Número de caminos independientes = 2
- INICIO, 1,2,...,37,38,39,40,39,41,42,38,39,41,42,43,FIN
(se recorre cuando hay una excepción)
-INICIO,1,2,...,37,38,FIN
(se recorre cuando el while es false)

Figura 1