

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

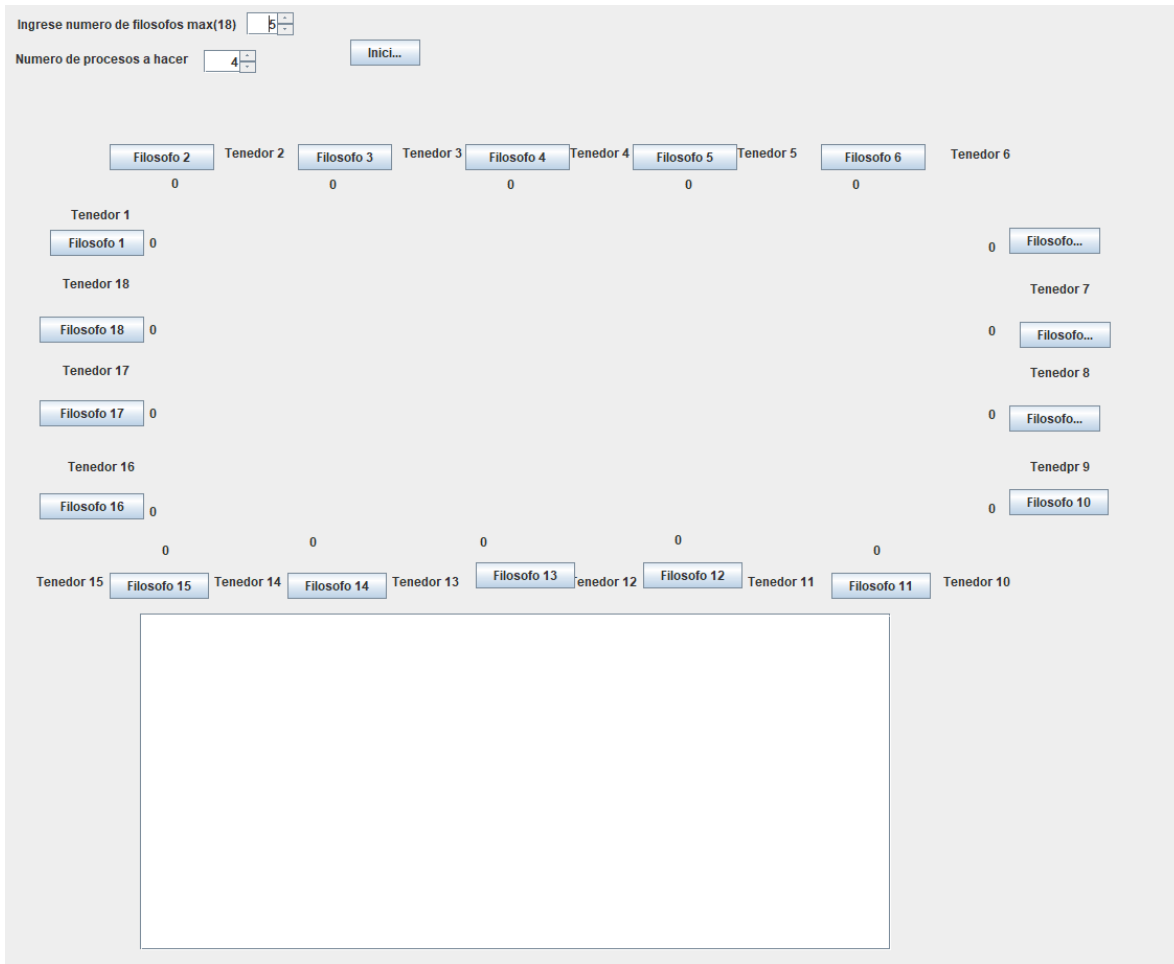
		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Hilos en Java	
OBJETIVO ALCANZADO: Identificar los cambios importantes de Java Diseñar e Implementar las nuevas técnicas de programación concurrente Entender cada una de las características de Thread en Java.			
ACTIVIDADES DESARROLLADAS			
1. Revisar los conceptos fundamentales de Thread en Java			
2. Establecer como implementar Thread en Java			
3. Implementar y diseñar los nuevos componentes de concurrencia			
4. Realizar el informe respectivo según los datos solicitados.			
5.			
6.			
N.			
RESULTADO(S) OBTENIDO(S): Realizar procesos de Hilos en Java. Entender las aplicaciones de codificación de las nuevas características de concurrencia. Entender las funcionalidades de sincronización y manejo de grupo de Thread dentro de Java.			
CONCLUSIONES: Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.			
RECOMENDACIONES:			

Nombre de estudiante: Adrian Lopez

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Firma de estudiante: _____


Interfaz




The interface displays a simulation of the Dining Philosophers problem. At the top, there are input fields for 'Ingrese numero de filosofos max(18)' (set to 5) and 'Numero de procesos a hacer' (set to 4), along with an 'Inici...' button. Below this, 18 philosophers (Filosofo 1 to 18) are arranged in a circular pattern, each with a corresponding fork (Tenedor 1 to 18). Each philosopher and fork has a counter set to 0. The philosophers are represented by blue boxes with their ID and a 'Filosofo...' button. The forks are represented by blue boxes with their ID. The interface is designed to visualize the execution of the algorithm, showing the state of each philosopher and fork as they interact.

La interfaz es una sola pantalla simple en donde vemos 18 filósofos a los que podemos utilizar para el funcionamiento del programa, así como 18 tenedores y 18 contadores de las veces que comen los filósofos.

En la parte de arriba del programa podemos elegir el número de filósofos y el numero de procesos que harán los mismos.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


 Ingrese numero de filosofos max(18)

Numero de procesos a hacer

Filosofo 2 Ocupado Comien... Ocupado

0 1 0

Ocupado

Comien... 1

Ocupado

Filosofo 18 0

Ocupado

Comiendo 1

Ocupado

Filosofo 16 0

Filosofo 4 Ocupado Comiendo Ocupado

0 1 0

Ocupado

Comiendo 1

Ocupado

Filosofo 13 Ocupado Comiendo Ocupado

0 1 0

Ocupado

Comiendo 1

Ocupado

Filosofo 15 Ocupado Comiendo Ocupado

0 1 0

Ocupado

Comiendo 1

Ocupado

Filosofo 11 Ocupado Comiendo Ocupado

0 1 0

Ocupado

Comiendo 1

Ocupado

Filosofo 6 Ocupado

0

Ocupado

Comien... 1

Ocupado

Filosofo...

0

Tenedor 8

0

Filosofo...

0


Tenedor 9


0

Filosofo 10

Fil.= 7 Comiendo usa sus tenedores
 Fil.= 14 Comiendo usa sus tenedores
 Fil.= 5 Comiendo usa sus tenedores
 Fil.= 3 Comiendo usa sus tenedores
 Fil.= 12 Comiendo usa sus tenedores
 Fil.= 1 Comiendo usa sus tenedores
 Fil.= 17 Comiendo usa sus tenedores

Para este informe he seleccionado 18 filósofos en 10 procesos por lo que ellos comenzaran a comer si tienes ambos tenedores libres hasta acabar los 10 procesos los procesos de comer y pensar tienen un tiempo de espera de 5000 milisegundos.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


 Ingrese numero de filosofos max(18)

Numero de procesos a hacer

Pensando Libre

18

Pensando Libre

18

Libre

18

Pensando Libre

18

Pensando Libre

18

Libre

Pensando

18

Libre

Pensando

18

Libre

Pensando

18

Libre

Pensando

18

Libre

Pensando

18

18

18

18

18

18

Libre

Pensando

Libre

Pensando

Libre

Pensando

Libre

Pensando

Libre

Fil = 14 Comiendo usa sus tenedores

Fil = 6 Comiendo usa sus tenedores

Fil = 4 Comiendo usa sus tenedores

Fil = 10 Comiendo usa sus tenedores

Fil = 8 Comiendo usa sus tenedores

Fil = 1 Comiendo usa sus tenedores

Fil = 16 Comiendo usa sus tenedores

Fil = 1 Deja de comer y queda pensando, libera sus tenedores

Fil = 16 Deja de comer y queda pensando, libera sus tenedores

Fil = 14 Deja de comer y queda pensando, libera sus tenedores

Fil = 8 Deja de comer y queda pensando, libera sus tenedores

Fil = 4 Deja de comer y queda pensando, libera sus tenedores

Fil = 6 Deja de comer y queda pensando, libera sus tenedores

Fil = 10 Deja de comer y queda pensando, libera sus tenedores

Fil = 18 Comiendo usa sus tenedores

Fil = 3 Comiendo usa sus tenedores

Fil = 5 Comiendo usa sus tenedores


Fil = 7 Comiendo usa sus tenedores

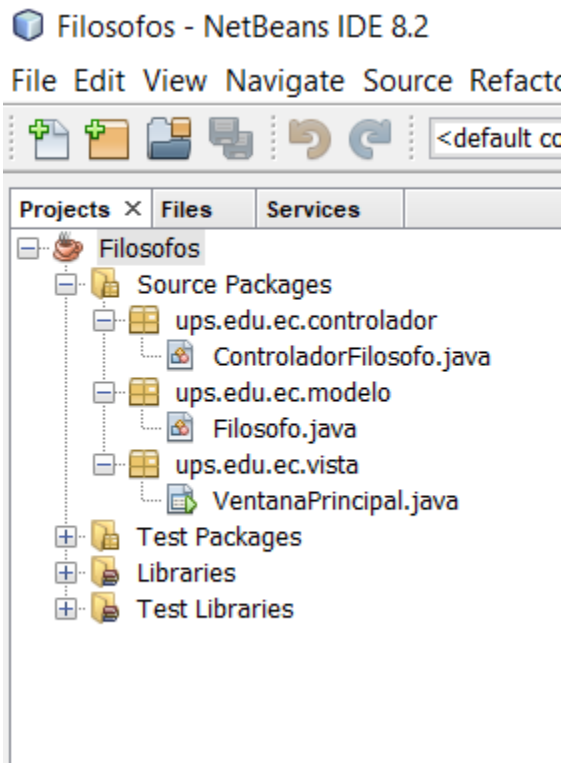
Fil = 9 Comiendo usa sus tenedores

Fil = 13 Comiendo usa sus tenedores


Una vez se acaban los procesos vemos que en este caso los 18 filósofos han comido 18 veces en el jtxArea nos va mostrando quien come y quien esta pensando.

Código

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



Se uso un patrón de diseño mvc.


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

Start Page x ControladorFilosofo.java x VentanaPrincipal.java x Filosofo.java x
Source Design History
1 package ups.edu.ec.vista;
2 import java.util.HashMap;
3 import java.util.Map;
4 import javax.swing.JButton;
5 import javax.swing.JLabel;
6 import ups.edu.ec.controlador.ControladorFilosofo;
7 import ups.edu.ec.modelo.Filosofo;
8 public class VentanaPrincipal extends javax.swing.JFrame {
9     Map<Integer, JButton> mapB = new HashMap<Integer, JButton>();
10    Map<Integer, JLabel> mapT = new HashMap<Integer, JLabel>();
11    Map<Integer, JLabel> mapR = new HashMap<Integer, JLabel>();
12    ControladorFilosofo controladorFilosofo;
13    public VentanaPrincipal() {
14        initComponents();
15        this.setExtendedState(this.MAXIMIZED_BOTH);
16
17
18        controladorFilosofo=new ControladorFilosofo();
19
20        mapB.put(1, jButton19); mapR.put(1, r1); mapT.put(1, t1);
21        mapB.put(2, jButton1); mapR.put(2, r2); mapT.put(2, t2);
22        mapB.put(3, jButton2); mapR.put(3, r3); mapT.put(3, t3);
23        mapB.put(4, jButton3); mapR.put(4, r4); mapT.put(4, t4);
24        mapB.put(5, jButton4); mapR.put(5, r5); mapT.put(5, t5);
25        mapB.put(6, jButton5); mapR.put(6, r6); mapT.put(6, t6);
26        mapB.put(7, jButton6); mapR.put(7, r7); mapT.put(7, t7);
27        mapB.put(8, jButton7); mapR.put(8, r8); mapT.put(8, t8);
28        mapB.put(9, jButton8); mapR.put(9, r9); mapT.put(9, t9);
29        mapB.put(10, jButton9); mapR.put(10, r10); mapT.put(10, t10);
30        mapB.put(11, jButton10); mapR.put(11, r11); mapT.put(11, t11);
31        mapB.put(12, jButton11); mapR.put(12, r12); mapT.put(12, t12);
32        mapB.put(13, jButton12); mapR.put(13, r13); mapT.put(13, t13);
33        mapB.put(14, jButton13); mapR.put(14, r14); mapT.put(14, t14);
34        mapB.put(15, jButton14); mapR.put(15, r15); mapT.put(15, t15);
35        mapB.put(16, jButton15); mapR.put(16, r16); mapT.put(16, t16);
36        mapB.put(17, jButton16); mapR.put(17, r17); mapT.put(17, t17);
37        mapB.put(18, jButton17); mapR.put(18, r18); mapT.put(18, t18);
38
39
40
41
42

```

Al iniciar el programa se crea un mapa con los 18 filósofos, tenedores y veces que han comido los filósofos asignándoles un valor int como llave.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

43 |
44 | }
45 | @SuppressWarnings("unchecked")
46 | Generated Code
364 |
365 | private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
366 |     Filosofo com;
367 |     JButton filosofo[];
368 |     JLabel tenedor[];
369 |     JLabel resultado[];
370 |     filosofo = new JButton(Integer.parseInt(jSpinner1.getValue().toString()));
371 |     tenedor = new JLabel(Integer.parseInt(jSpinner1.getValue().toString()));
372 |     resultado = new JLabel(Integer.parseInt(jSpinner1.getValue().toString()));
373 |
374 |     for (int i = 1; i <= Integer.parseInt(jSpinner1.getValue().toString()); i++) {
375 |
376 |         JButton boton=mapB.get(i);
377 |         JLabel tene=mapT.get(i);
378 |         JLabel resu=mapR.get(i);
379 |         filosofo[i-1]=boton;
380 |
381 |         tenedor[i-1]=tene;
382 |
383 |         resultado[i-1]=resu;
384 |     }
385 |
386 |     int i, izq, der = 0;
387 |
388 |
389 |     for (i = 0; i < Integer.parseInt(jSpinner1.getValue().toString()); i++) {
390 |
391 |         izq = i - 1;
392 |
393 |         if (izq < 0) {
394 |             izq = Integer.parseInt(jSpinner1.getValue().toString())-1;
395 |         }
396 |         der = i;
397 |
398 |         controladorFilosofo.create(i, tenedor[izq], tenedor[der], filosofo[i], resultado[i], jTextArea1,Integer.parseInt(jSpinner1.getValue().toString())
399 |     }
400 |
401 |


```

Al apastar el botón se crean el número que hallamos elegido y se va ejecutando los procesos en filosofo.

```

Start Page x ControladorFilosofo.java x VentanaPrincipal.java x Filosofo.java x
Source History
39 }
40 @Override
41 public void run() {
42     for(int i =0;i<numerosveces;i++){
43         System.out.println(i);
44         synchronized(this.izquierdo){
45             synchronized(this.derecho){
46                 comer();
47             }
48         }
49         pensar();
50     }
51 }
52 }
53 void comer () {
54     derecho.setText("Ocupado");
55     derecho.setForeground(Color.red);
56
57     izquierdo.setText("Ocupado");
58     izquierdo.setForeground(Color.red);
59
60     filosofo.setText("Comiendo");
61     filosofo.setBackground(Color.GREEN);
62
63     res=Integer.parseInt(resultado.getText());
64     res+=1;
65     resultado.setText(String.valueOf(res));
66     proceso= "Fil.- "+(id+1)+ " Comiendo usa sus tenedores\n";
67     textArea.append(proceso);
68     try{
69         Thread.sleep(5000);
70     }catch(InterruptedException e){
71     }
72     derecho.setText("Libre");
73     derecho.setForeground(Color.black);
74
75     izquierdo.setText("Libre");
76     izquierdo.setForeground(Color.black);
77
78     filosofo.setText("Pensando");
79     filosofo.setBackground(Color.DARK_GRAY);
80     proceso="Fil.- "+(id+1)+ " Deja de comer y queda pensando, libera sus tenedores\n";
81     textArea.append(proceso);
82 }
83 void pensar() {
84     derecho.setText("Libre");
85
86     izquierdo.setText("Libre");
87     izquierdo.setForeground(Color.black);
88
89     filosofo.setText("Pensando");
90     filosofo.setBackground(Color.DARK_GRAY);
91
92     try{
93
94         Thread.sleep(5000);
95     }catch(InterruptedException e){
96     }
97 }
98 }

```


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

La clase filosofo tiene 3 métodos el run que se ejecutara cuando se cree un filosofo y 2 mas que son comiendo y pensar que se sincronizaran y cambiaran los filósofos si estos tienen disponibles los tenedores.