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

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Universidad Politécnica Salesiana

Vicerrectorado Docente

Código del Formato:	GUIA-PRL-001
Versión:	VF1.0
Elaborado por:	Directores de Área del Conocimiento Integrantes Consejo Académico
Fecha de elaboración:	2016/04/01
Revisado por:	Consejo Académico
Fecha de revisión:	2016/04/06
Aprobado por:	Lauro Fernando Pesántez Avilés Vicerrector Docente
Fecha de aprobación:	2016/14/06
Nivel de confidencialidad:	Interno

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

Descripción General

Propósito


El propósito del presente documento es definir un estándar para elaborar documentación de guías de práctica de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana, con la finalidad de lograr una homogenización en la presentación de la información por parte del personal académico y técnico docente.


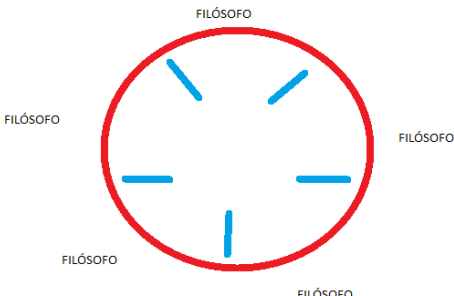
Alcance


El presente estándar será aplicado a toda la documentación referente a informes de prácticas de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana.

Formatos

- Formato de Guía de Práctica de Laboratorio / Talleres / Centros de Simulación – para Docentes
- Formato de Informe de Práctica de Laboratorio / Talleres / Centros de Simulación – para Estudiantes

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

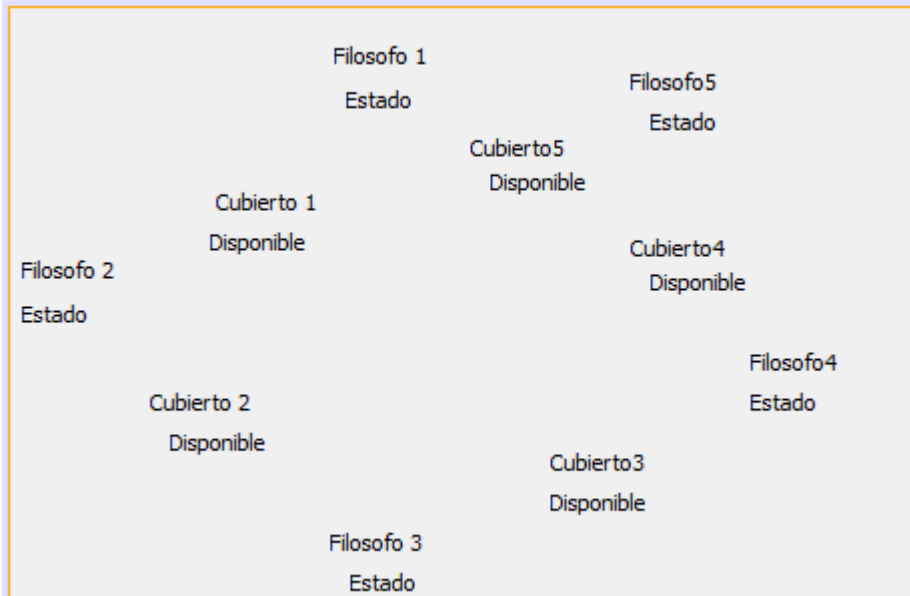
		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Hilos en Java	
OBJETIVO: Identificar los cambios importantes de Java Diseñar e Implementar las nuevas técnicas de programación concurrente Entender cada una de las características de Thread en Java.			
INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):	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.		
ACTIVIDADES POR DESARROLLAR (Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)			
1. Revisar la teoría y conceptos de Thread en Java			
2. Diseñar e implementar las características de Java para generar una simulación 2D del siguiente enunciado:			
Problema del Filósofo: En una mesa hay procesos que simulan el comportamiento de unos filósofos que intentan comer de un plato. Cada filósofo tiene un cubierto a su izquierda y uno a su derecha y para poder comer tiene que conseguir los dos. Si lo consigue, mostrará un mensaje en pantalla que indique «Filósofo 2 (numero) comiendo». Después de comer, soltará los cubiertos y esperará al azar un tiempo entre 1000 y 5000 milisegundos, indicando por pantalla «El filósofo 2 está pensando».			
En general todos los objetos de la clase Filósofo está en un bucle infinito dedicándose a comer y a pensar.			
Simular este problema en un programa Java que muestre el progreso de todos sin caer en problemas de sincronización a través de un método grafico.			
			

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

3. Probar y modificar el método para que nos permita cambiar el numero de filósofos.

4. Realizar práctica codificando con las nuevas características de Java, patrones de diseño, Thread, etc.

5. Fecha de Entrega: 11 Enero del 2021 23:55



```

6   public class Vista extends javax.swing.JFrame {
7
8       Thread Filosofo1;
9       Thread Filosofo2;
10      Thread Filosofo3;
11      Thread Filosofo4;
12      Thread Filosofo5;
13      Thread cubiertos;
14      ArrayList<cubierto> lista = new ArrayList<>();
15
16      int total = 5;

```

```

18 public Vista() {
19     initComponents();
20     for (int i = 0; i < total; i++) {
21         lista.add(new cubierto(i, "Disponible"));
22     }
23     Filosofo1 = new Thread(new Vista.hilo(1));
24     Filosofo2 = new Thread(new Vista.hilo(2));
25     Filosofo3 = new Thread(new Vista.hilo(3));
26     Filosofo4 = new Thread(new Vista.hilo(4));
27     Filosofo5 = new Thread(new Vista.hilo(5));
28     cubiertos = new Thread(new Vista.cubiertoHilo());
29     Filosofo1.start();
30     Filosofo2.start();
31     Filosofo3.start();
32     Filosofo4.start();
33     Filosofo5.start();
34     cubiertos.start();
35 }

36
37
38
39 @Override
40 public void run() {
41     while (true) {
42         for (int i = 0; i < lista.size(); i++) {
43             switch (i) {
44                 case 0:
45                     estado6.setText(lista.get(i).getEstado());
46                     if (lista.get(i).getEstado().equals("Disponible")) {
47                         estado6.setForeground(Color.red);
48                     } else {
49                         estado6.setForeground(Color.black);
50                     }
51                     break;
52                 case 1:
53                     estado7.setText(lista.get(i).getEstado());
54                     if (lista.get(i).getEstado().equals("Disponible")) {
55                         estado7.setForeground(Color.red);
56                     } else {
57                         estado7.setForeground(Color.black);
58                     }
59                     break;

```

```

60         case 2:
61             estado8.setText(lista.get(i).getEstado());
62             if (lista.get(i).getEstado().equals("Disponible")) {
63                 estado8.setForeground(Color.red);
64             } else {
65                 estado8.setForeground(Color.black);
66             }
67             break;
68         case 3:
69             estado9.setText(lista.get(i).getEstado());
70             if (lista.get(i).getEstado().equals("Disponible")) {
71                 estado9.setForeground(Color.red);
72             } else {
73                 estado9.setForeground(Color.black);
74             }
75             break;
76         case 4:
77             estado10.setText(lista.get(i).getEstado());
78             if (lista.get(i).getEstado().equals("Disponible")) {
79                 estado10.setForeground(Color.red);
80             } else {
81                 estado10.setForeground(Color.black);
82             }
83             break;
84     }
85 }
86

```

```

91 public class cubierto {
92
93     private int numero;
94     private String estado;
95
96     public cubierto(int numero, String estado) {
97         this.numero = numero;
98         this.estado = estado;
99     }
100
101     public int getNumero() {
102         return numero;
103     }
104
105     public void setNumero(int numero) {
106         this.numero = numero;
107     }
108
109     public String getEstado() {
110         return estado;
111     }
112
113     public void setEstado(String estado) {
114         this.estado = estado;
115     }
116
117 }


```

```

119 public class hilo implements Runnable {
120     int numero;
121     private hilo(int i) {
122         this.numero = i;
123     }
124
125     public void run() {
126         try {
127             while (true) {
128                 int wait = (int) (Math.random() * (5000 - 1000) + 1000);
129                 cambiar(numero, ("Pensando Filosofo " + numero), true);
130                 Thread.sleep(wait);
131                 if (numero > 1) {
132                     if (lista.get(numero - 2).estado.equals("Disponible") &&
133                         lista.get(numero-1).estado.equals("Disponible")) {
134                         lista.get(numero - 2).setEstado("Ocupado");
135                         lista.get(numero-1).setEstado("Ocupado");
136                         cambiar(numero, ("Comiendo Filosofo " + numero), false);
137                         Thread.sleep(2000);
138                         lista.get(numero - 2).setEstado("Disponible");
139                         lista.get(numero-1).setEstado("Disponible");
140                     }
141                 } else {
142                     if (lista.get(numero-1).estado.equals("Disponible") &&
143                         lista.get(total-1).estado.equals("Disponible")) {
144                         lista.get(numero-1).setEstado("Ocupado");
145                         lista.get(total-1).setEstado("Ocupado");
146                         cambiar(numero, ("Comiendo Filosofo " + numero), false);
147                         Thread.sleep(2000);
148                         lista.get(numero - 1).setEstado("Disponible");
149                         lista.get(total - 1).setEstado("Disponible");
150                     }
151                 }
152             }
153         } catch (Exception e) {
154         }
155     }
156 }
157
158

```

```
159 public void cambiar(int numero, String mensaje, boolean x) {
160     switch (numero) {
161         case 1:
162             estado1.setText(mensaje);
163             if (x) {
164                 estado1.setForeground(Color.red);
165             } else {
166                 estado1.setForeground(Color.black);
167             }
168             break;
169         case 2:
170             estado2.setText(mensaje);
171             if (x) {
172                 estado2.setForeground(Color.red);
173             } else {
174                 estado2.setForeground(Color.black);
175             }
176             break;
177         case 3:
178             estado3.setText(mensaje);
179             if (x) {
180                 estado3.setForeground(Color.red);
181             } else {
182                 estado3.setForeground(Color.black);
183             }
184             break;
185         case 4:
186             estado4.setText(mensaje);
187             if (x) {
188                 estado4.setForeground(Color.red);
189             } else {
190                 estado4.setForeground(Color.black);
191             }
192             break;
193         case 5:
194             estado5.setText(mensaje);
195             if (x) {
196                 estado5.setForeground(Color.red);
197             } else {
198                 estado5.setForeground(Color.black);
199             }
200             break;
201     }
202 }
203 }
```


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



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:

Realizar el trabajo dentro del tiempo establecido.

Docente / Técnico Docente: _____

Firma: _____

		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA:		ASIGNATURA:	
NRO. PRÁCTICA:		TÍTULO PRÁCTICA:	

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

OBJETIVO ALCANZADO:
ACTIVIDADES DESARROLLADAS
1.
2.
3.
4.
5.
6.
N.
RESULTADO(S) OBTENIDO(S):
CONCLUSIONES:
RECOMENDACIONES:

Nombre de estudiante: _____

Firma de estudiante: _____