	<b>VICERRECTORADO DOCENTE</b>	<b>Código:</b> GUIA-PRL-001
	CONSEJO ACADÉMICO	<b>Aprobación:</b> 2016/04/06
<b>Formato:</b> 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

	<b>VICERRECTORADO DOCENTE</b>	<b>Código:</b> GUIA-PRL-001
	CONSEJO ACADÉMICO	<b>Aprobación:</b> 2016/04/06
<b>Formato:</b> 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.


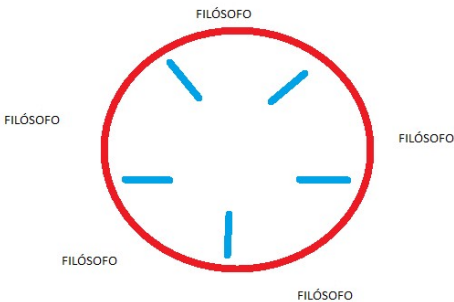
### Alcance


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		

		<b>FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES</b>	
<b>CARRERA:</b> COMPUTACIÓN		<b>ASIGNATURA:</b> Programación Aplicada	
<b>NRO. PRÁCTICA:</b>	1	<b>TÍTULO PRÁCTICA:</b> Hilos en Java	
<b>OBJETIVO:</b> 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.			
<b>INSTRUCCIONES</b> (Detallar las instrucciones que se dará al estudiante):	1. Revisar los conceptos fundamentales de Thread en Java		
	2. Establecer cómo implementar Thread en Java		
	3. Implementar y diseñar los nuevos componentes de concurrencia		
	4. Realizar el informe respectivo según los datos solicitados.		
<b>ACTIVIDADES POR DESARROLLAR</b> (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:  <b>Problema del Filósofo:</b>  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 gráfico.			
			

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

**3.** Probar y modificar el método para que nos permita cambiar el número 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

**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.


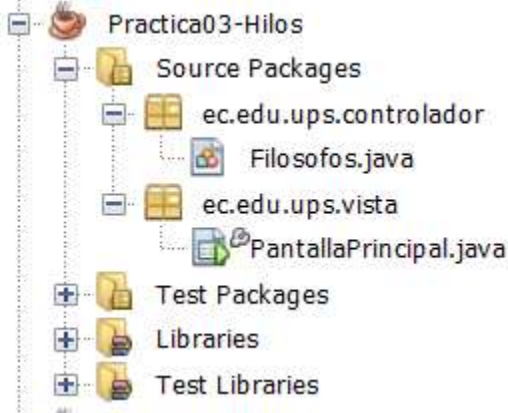
**RECOMENDACIONES:**

Realizar el trabajo dentro del tiempo establecido.

**Docente / Técnico Docente:** \_\_\_\_\_

**Firma:** \_\_\_\_\_

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

		<b>FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES</b>	
<b>CARRERA:</b> COMPUTACION		<b>ASIGNATURA:</b> Programación Aplicada	
<b>NRO. PRÁCTICA:</b>	4	<b>TÍTULO PRÁCTICA:</b> Hilos en Java	
<b>OBJETIVO ALCANZADO:</b> 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.			
<b>ACTIVIDADES DESARROLLADAS</b>			
<b>1. Revisar la teoría y conceptos de Thread en Java.</b>  Thread: es la clase base de Java para definir hilos de ejecución concurrentes dentro de un mismo programa,			
<b>2. Diseñar e implementar las características de Java para generar una simulación 2D del siguiente enunciado:</b>  Se creó una aplicación en el IDE Netbeans para resolver el problema planteado, el cual tiene 2 paquetes la cuales contiene un controlador y una vista.			
			
Ya que para resolver el problema se necesita los threads, en la clase filósofo se extiende la clase para crear los hilos.			
<pre> /**  *  * @author Sebastian Uyaguari  */ public class Filosofos extends Thread{ </pre>			

Cuando se extiende la clase thread necesariamente se debe implementar el método run en el cual se debe sincronizar los palillos derechos e izquierdos:

```
@Override
public void run() {
    for (int i = 0; i < 100; i++) {
        synchronized(this.izquierdo){
            synchronized(this.derecho){
                comer();
            }
        }
        pensar();
    }
}
```


En el cual tiene los dos métodos para controlar la vista y también los palillos para quien deba comer o no.

En la vista se utilizó unos dos paneles en los cuales se agregan de forma dinámica desde los 5 filósofos generados automáticamente hasta un N número de filósofos.

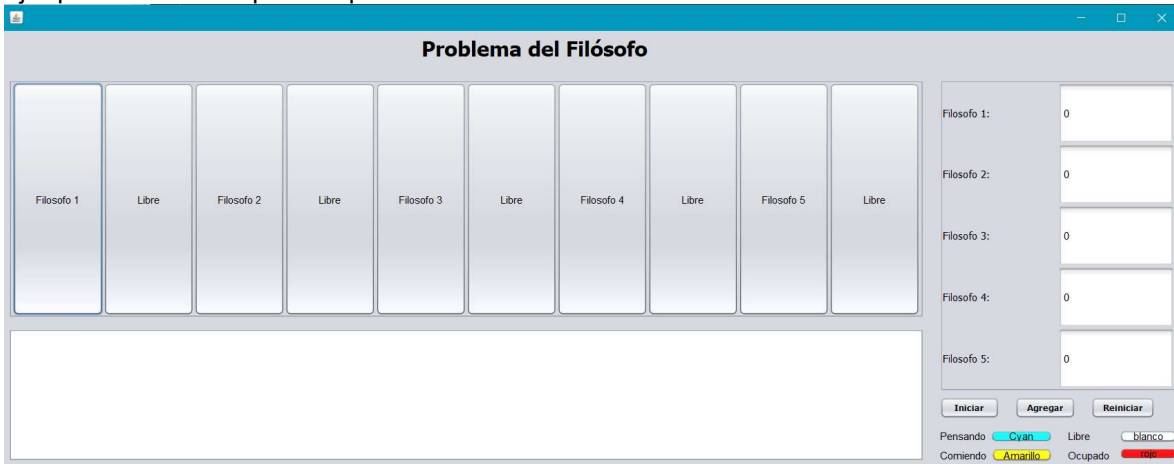
```
public void crearFilosofos() {
    for (int i = 0; i < creacion; i++) {
        JButton boton = new JButton("Filosofo "+(i+1));
        pnlBotones.add(boton);
        botones[i] = boton;
        JButton tenedo = new JButton("Libre");
        pnlBotones.add(tenedo);
        tenedor[i] = tenedo;
    }
    pnlBotones.updateUI();
    for (int i = 0; i < creacion; i++) {
        JLabel[i] = new JLabel();
        JLabel[i].setFont(new Font("Tahoma", Font.PLAIN, 12));
        JLabel[i].setText("Filosofo "+(i+1)+":");

        resultado[i] = new JTextField("0");

        pnlInformacion.add(JLabel[i]);
        pnlInformacion.add(resultado[i]);
    }
    pnlInformacion.updateUI();
}
```

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

Para mejor presentación se puede apreciar de forma visual mediante la interfaz.



**Problema del Filósofo**

Filosofo 1	Libre	Filosofo 2	Libre	Filosofo 3	Libre	Filosofo 4	Libre	Filosofo 5	Libre
------------	-------	------------	-------	------------	-------	------------	-------	------------	-------

Filosofo 1: 0  
Filosofo 2: 0  
Filosofo 3: 0  
Filosofo 4: 0  
Filosofo 5: 0

Iniciar Agregar Reiniciar

Pensando   Libre    
Comiendo   Ocupado  

En el caso de ponerlo en ejecución:



**Problema del Filósofo**

Filosofo2	Libre	Filosofo3	Libre	Filosofo4	Ocupado	Filosofo5	Ocupado	Filosofo1	Libre
-----------	-------	-----------	-------	-----------	---------	-----------	---------	-----------	-------

Filosofo 1: 2  
Filosofo 2: 2  
Filosofo 3: 1  
Filosofo 4: 2  
Filosofo 5: 1

Iniciar Agregar Reiniciar

Pensando   Libre    
Comiendo   Ocupado  

El filosofo 1 esta comiendo  
El filosofo 3 esta comiendo  
El filosofo 1 libero sus tenedores  
El filosofo 3 libero sus tenedores  
El filosofo 1 esta pensando  
El filosofo 3 esta pensando  
El filosofo 2 esta comiendo  
El filosofo 5 esta comiendo  
El filosofo 2 libero sus tenedores

Y por último agregando los componentes:



**Problema del Filósofo**


Filosofo2	Libre	Filosofo3	Ocupado	Filosofo4	Ocupado	Filosofo5	Ocupado	Filosofo6	Ocupado
Filosofo7	Libre	Filosofo8	Libre	Filosofo9	Ocupado	Filosofo10	Libre	Filosofo11	Libre
Filosofo12	Libre	Filosofo13	Libre	Filosofo14	Libre	Filosofo15	Libre	Filosofo1	Libre

Filosofo 1: 8  
Filosofo 2: 7  
Filosofo 3: 8  
Filosofo 4: 7  
Filosofo 5: 8  
Filosofo 6: 7  
Filosofo 7: 8  
Filosofo 8: 6  
Filosofo 9: 6  
Filosofo 10: 7  
Filosofo 11: 7  
Filosofo 12: 9  
Filosofo 13: 7

Iniciar Agregar Reiniciar

Pensando   Libre    
Comiendo   Ocupado  

El filosofo 8 esta comiendo  
El filosofo 4 libero sus tenedores  
El filosofo 4 esta pensando  
El filosofo 2 libero sus tenedores  
El filosofo 2 esta pensando  
El filosofo 1 esta comiendo  
El filosofo 12 esta comiendo  
El filosofo 14 esta comiendo  
El filosofo 3 esta comiendo

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

### Código:

#### Clase Filosofo:

```
package ec.edu.ups.controlador;

import java.awt.Color;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JTextArea;
import javax.swing.JTextField;

/**
 *
 * @author Sebastian Uyaguari
 */
public class Filsofos extends Thread{

    private Thread thread;
    private JButton filosofo;
    private int id;
    private int res;
    private JButton derecho;
    private JButton izquierdo;
    private JTextField resultado;
    private JTextArea txtArea;

    private Random randomico = new Random();

    public Filsofos(int id, JButton filosofo, JButton derecho, JButton izquierdo,
JTextField resultado, JTextArea txtArea) {
        this.id = id;
        this.filosofo = filosofo;
        this.derecho = derecho;
        this.izquierdo = izquierdo;
        this.resultado = resultado;
        this.txtArea = txtArea;

        thread = new Thread(this);

        thread.start();
    }

    @Override
    public void run(){
        for (int i = 0; i < 100; i++) {
            synchronized(this.izquierdo){
                synchronized(this.derecho){
                    comer();
                }
            }
            pensar();
        }
    }
}
```



```
public void comer() {
    String frase;

    derecho.setText("Ocupado");
    derecho.setBackground(Color.RED);

    izquierdo.setText("Ocupado");
    izquierdo.setBackground(Color.RED);

    filosofo.setText("Filosofo" + (id + 1) + "");
    filosofo.setBackground(Color.YELLOW);

    res = Integer.parseInt(resultado.getText());
    res++;
    resultado.setText(res + "");

    frase = "El filosofo " + (id + 1) + " esta comiendo \n";

    txtArea.append(frase);

    try {
        Thread.sleep(1000);
    } catch (InterruptedException ex) {

    }

    filosofo.setText("Filosofo" + (id + 1) + "");
    filosofo.setBackground(Color.CYAN);
    frase = "El filosofo " + (id + 1) + " libero sus tenedores\n";
    txtArea.append(frase);
}

public void pensar() {
    String frase;

    derecho.setText("Libre");
    derecho.setBackground(Color.WHITE);

    izquierdo.setText("Libre");
    izquierdo.setBackground(Color.WHITE);

    filosofo.setText("Filosofo" + (id + 1) + "");
    filosofo.setBackground(Color.CYAN);

    frase = "El filosofo " + (id + 1) + " esta pensando\n";
    txtArea.append(frase);

    try {

        Thread.sleep(randomico.nextInt(4000) + 1000);

    } catch (InterruptedException ex) {
```

```
    }  
  
    }  
  
}
```

**Clase Pantalla Principal:**

```
package ec.edu.ups.vista;  
  
import ec.edu.ups.controlador.Filosofos;  
import java.awt.Font;  
import javax.swing.JButton;  
import javax.swing.JLabel;  
import javax.swing.JTextField;  
  
/**  
 *  
 * @author Sebastian Uyaguari  
 */  
public class PantallaPrincipal extends javax.swing.JFrame {  
  
    private int creacion = 5;  
    JButton[] botones = new JButton[creacion];  
    JButton[] tenedor = new JButton[creacion];  
    JTextField[] resultado = new JTextField[creacion];  
    JLabel[] label = new JLabel[creacion];  
  
    Filosofos comensales;  
  
    public PantallaPrincipal() {  
        initComponents();  
        this.setLocationRelativeTo(null);  
        crearFilosofos();  
    }  
  
    private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {  
  
        int i;  
        int izquierda;  
        int derecha;  
  
        for (i = 0; i < creacion; i++) {  
  
            izquierda = i - 1;  
  
            if (izquierda < 0) {  
  
                izquierda = creacion-1;
```

```
    }

    derecha = i;

    comensales = new Filósofos(i, botones[izquierda], tenedor[derecha],
        tenedor[izquierda], resultado[i], txtInformacion);
}

private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    removerComponentes();
    creacion++;
    botones = new JButton[creacion];
    tenedor = new JButton[creacion];
    resultado = new JTextField[creacion];
    label = new JLabel[creacion];
    crearFilosofos();
}


private void btnReiniciarActionPerformed(java.awt.event.ActionEvent evt) {
    removerComponentes();
    creacion=5;
    crearFilosofos();
}

public void crearFilosofos(){

    for (int i = 0; i < creacion; i++) {
        JButton boton = new JButton("Filósofo "+(i+1));
        pnlBotones.add(boton);
        botones[i] = boton;
        JButton tenedo = new JButton("Libre");
        pnlBotones.add(tenedo);
        tenedor[i] = tenedo;
    }
    pnlBotones.updateUI();
    for (int i = 0; i < creacion; i++) {
        label[i] = new JLabel();
        label[i].setFont(new Font("Tahoma", Font.PLAIN, 12));
        label[i].setText("Filósofo "+(i+1)+":");

        resultado[i] = new JTextField("0");

        pnlInformacion.add(label[i]);
        pnlInformacion.add(resultado[i]);
    }
    pnlInformacion.updateUI();
}
```

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

```

public void removerComponentes () {
    for (int i = 0; i < creacion; i++) {
        pnlBotones.remove(botones[i]);
        pnlBotones.remove(tenedor[i]);
        pnlInformacion.remove(resultado[i]);
        pnlInformacion.remove(label[i]);
    }
    txtInformacion.setText("");
}

// Variables declaration - do not modify
private javax.swing.JButton btnAgregar;
private javax.swing.JButton btnIniciar;
private javax.swing.JButton btnReiniciar;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JPanel pnlBotones;
private javax.swing.JPanel pnlInformacion;
private javax.swing.JTextArea txtInformacion;
// End of variables declaration
}

```

#### 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.

**Nombre de estudiante:** Sebastián Roberto Uyaguari Ramon

	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:**