	<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.


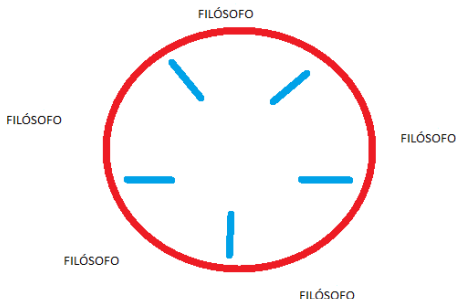
### Alcance


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 todo el objeto 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. <b>Fecha de Entrega:</b> 11 Enero del 2021 23:55
<b>RESULTADO(S) OBTENIDO(S):</b> 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.
<b>CONCLUSIONES:</b> Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.
<b>RECOMENDACIONES:</b> Realizar el trabajo dentro del tiempo establecido.

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

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

		<b>FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES</b>
<b>CARRERA:</b> Computacion.		<b>ASIGNATURA:</b> Programacion Aplicada.
<b>NRO. PRÁCTICA:</b>		<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>		
1. Revisar la teoría y conceptos de Thread en Java  Thread (hilo, tarea) es la clase base de Java para definir de ejecución concurrentes dentro de un mismo programa.  El concepto de concurrencia está asociado a los objetos: Son los objetos que actúan concurrentemente con otros.  Un objeto concurrente pertenece a una clase que extiende Thread.  Hay que redefinir el método run () que especifica la tarea concurrente.  Al implementar de la clase Runnable debemos implementar el método run obligatoriamente ya que este método es el encargo de toda la ejecución del programa.		


```
@Override
public void run() {
    while (iterar) {
        //podria ser un ciclo infinito pero lo que
        //tenga una mejor presetrnacion.
        for (int i = 0; i < 10000; i++) {
            //Si no utilizamos el synchronized no
            //cada uno se ejecutaria en diferente
            //Compiten todos los procesos, viene
            //ocupado no participaria, pero si no
            synchronized (this.izquierdo) {
                synchronized (this.derecho) {
                    comer();
                }
            }
            pensar();
        }
        if (pausar) {
            break;
        }
    }
}
```

La ejecución de la tarea concurrente se realiza mediante el método start (heredado de thread)

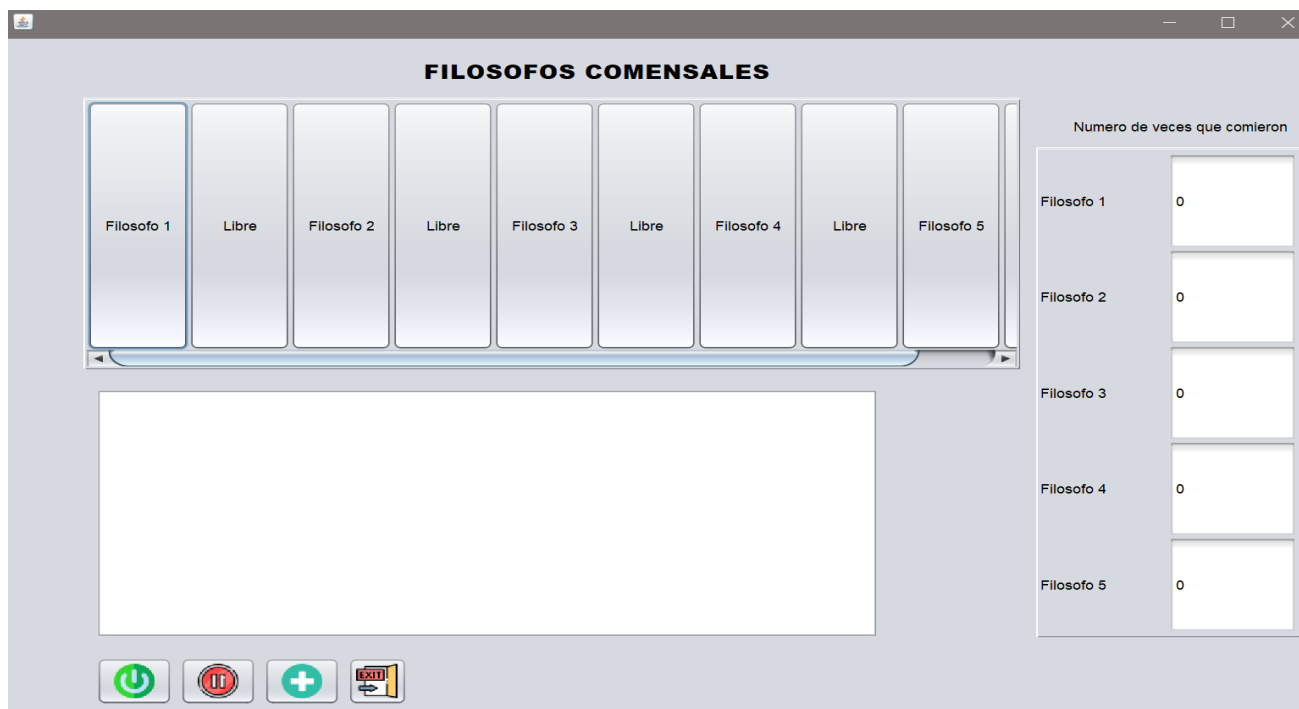
```
tread = new Thread(this);
tread.start();
```

La interfaz Runnable proporciona un método alternativo a la utilización de la clase Thread, para los casos en los que no es posible hacer que la clase definida extienda la clase Thread. Esto ocurre cuando dicha clase que se desea ejecutar en un hilo independiente deba extraer alguna otra clase

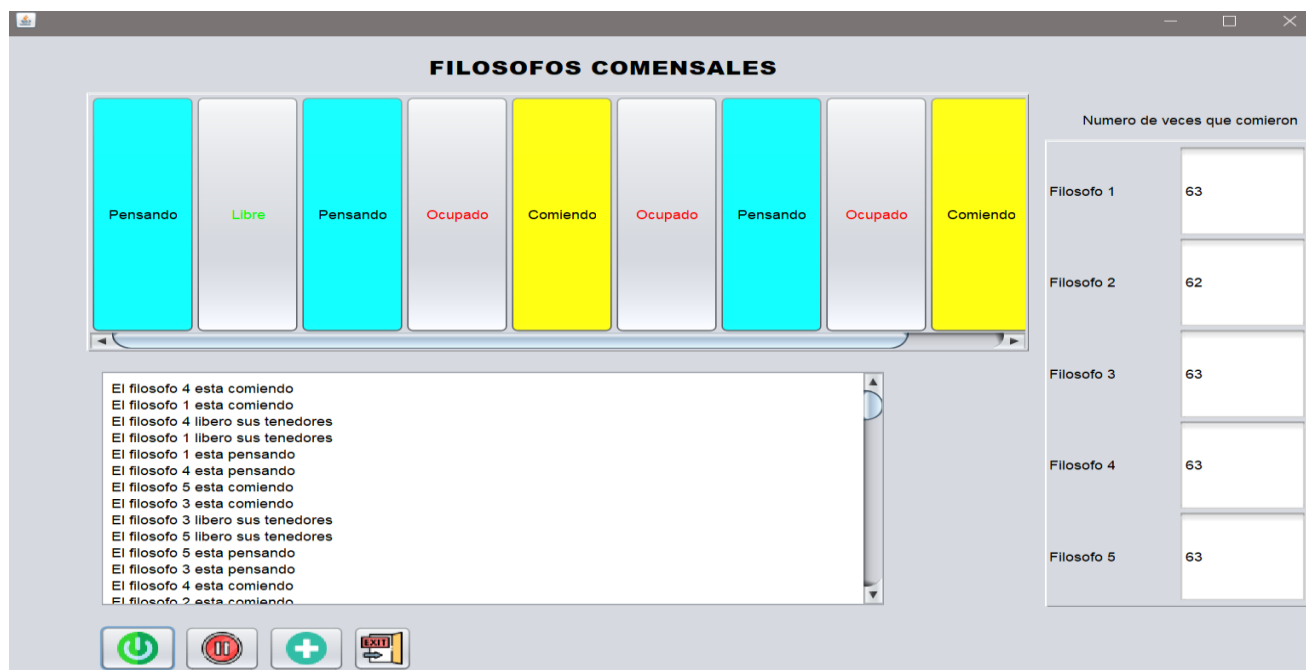
```
public class Filósofos implements Runnable {
```


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

2. Diseñar e implementar las características de Java para generar una simulación 2D del siguiente enunciado



Al presionar el botón verde iniciara el programa con cinco filosofos.



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

Como vemos nos muestra los numero de veces que los filósofos hayan comido

Numero de veces que comieron	
Filosofo 1	29
Filosofo 2	29
Filosofo 3	29
Filosofo 4	30
Filosofo 5	29

Y como vemos en el JTextArea nos muestra lo que ha realizado el filósofo.

```

El filosofo 1 esta comiendo
El filosofo 4 esta comiendo
El filosofo 1 libero sus tenedores
El filosofo 4 libero sus tenedores
El filosofo 1 esta pensando
El filosofo 4 esta pensando

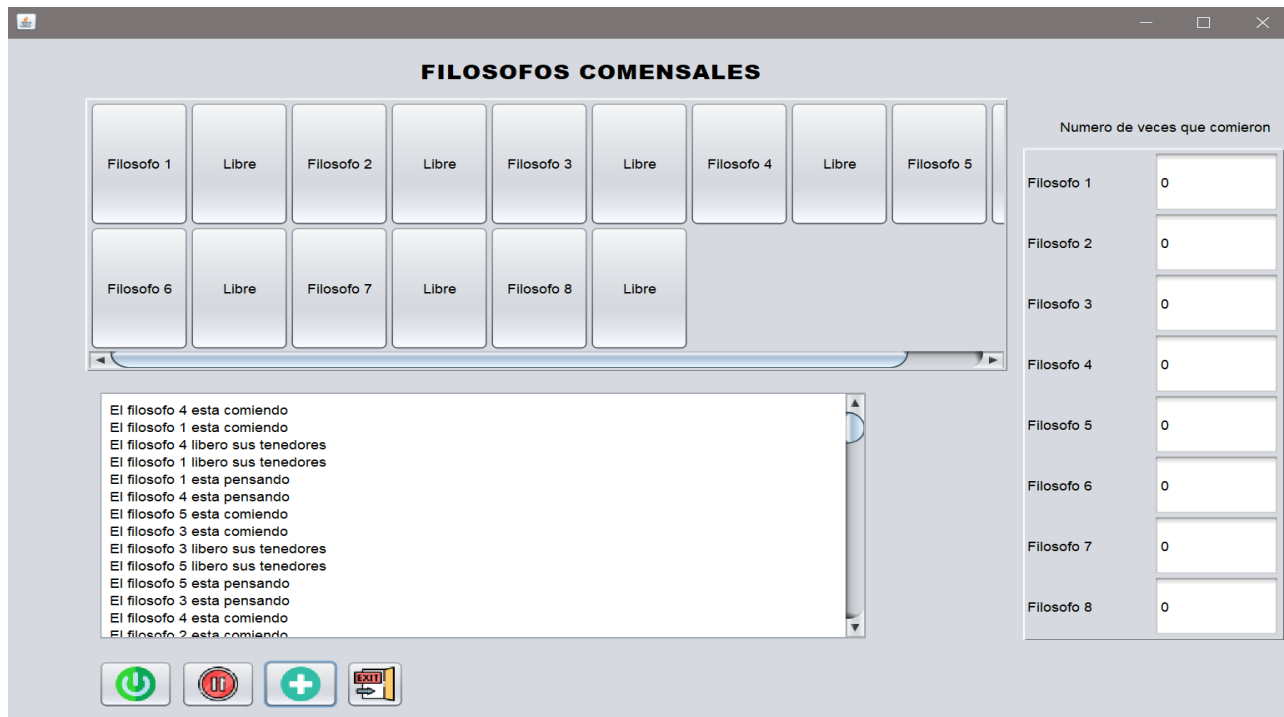
```

Tenemos un botón pausar donde si lo pulsamos este se pausará por unos segundos y un botón para salir.



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


Podremos agregar filósofos al presionar el botón con el signo más.



Ya trabajando cuando hemos añadido nuevos filósofos





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

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

- **CLASE FILOSOFO**

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.ups.edu.filosofos;

import java.awt.Color;
import javax.swing.JLabel;
import javax.swing.JTextArea;
import javax.swing.JTextField;

//Solucion del problema
//Nos dicen que cinco filosofos se pasan pensando y comiendo
//estos comen solo si los dos palillos estan desocupados
//por lo cual le vamos a asignar un par de palillos a cada uno
//Por ejemplo para el filosofo numero 1 podra ocupar el palillo numero 1
//y el palillo numero 5
/**
 *
 * @author santi
 */
//Se ha implementado el Runnable el cual nos va a ayudar a hacer uso del Thread
//con el cual vamos a poder iniciar el hilo realizar la sincronizacion de los hilos.

public class Filosofos implements Runnable {

    //variables
    int id;
    int res;
    Thread tread;
    //JButton filosofo;
    JLabel derecho;
    JLabel izquierdo;
    JTextField resultado;
    String proceso;
    JTextArea txtArea;
    JLabel filosofo;
    public static boolean pausar;
    public static boolean iterar;

    public Filosofos() {
    }

    //constructor

```

```
//Aqui tambien tenemos Thread y le decimos que
public Filósofos(int id, JLabel izquierdo, JLabel derecho, JLabel filosofo, JTextField
resultado, JTextArea txtArea) {

    this.id = id;
    this.filosofo = filosofo;
    this.derecho = derecho;
    this.izquierdo = izquierdo;
    this.resultado = resultado;
    this.txtArea = txtArea;
    this.pausar = false;
    this.iterar = true;

    //Aqui instanciamos al tread
    tread = new Thread(this);
    //El metodo star se va a ejecutar siempre y cuando exista el metodo run que ya
    //sabemos que es obligatorio crear ese metodo cuando trabajamos con hilos
    //debido a que es donde se van a ejecutar los procesos.


    tread.start();

}

@Override
public void run() {
    while (iterar) {
        //podria ser un ciclo infinito pero lo que es darle un limite de 2 para que
        //tenga una mejor presetrnacion.
        for (int i = 0; i < 10000; i++) {
            //Si no utilizamos el synchronized no se tendríamos un control
            //cada uno se ejecutaria en diferentes momentos.
            //Compiten todos los procesos, viene cualquiera de los metodos, y si el tenedor
            izquierdo esta //ocupado no participaria, pero si no esta ocupado este entraria al proceso y
            ejecutaria el metodo comer.
            synchronized (this.izquierdo) {
                synchronized (this.derecho) {
                    comer();
                }
            }
            pensar();
        }
        if (pausar) {
            break;
        }
    }
}

public void comer() {

    //Cambiamos los estados de los tenedores de libre a ocupado tanto del tenedor derecho
    como del izquierdo.
    //y le asignamos un color rojo cuando este este Ocupado.
    derecho.setText("Ocupado");
}
```

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

```

derecho.setForeground(Color.red);

izquierdo.setText("Ocupado");
izquierdo.setForeground(Color.red);

//Cambiamos los estados de los filofos caundo este comiendo.
//Y le damos un color amarillo cuando este este comiendo
filosofo.setText("Comiendo");
filosofo.setBackground(Color.YELLOW);

//Contador
//Capturamos la etiqueta resultado y le guardamos en la variable ress
//y este ress va a ir aumentado 1 a 1 cada vez que el filofo coma
ress = Integer.parseInt(resultado.getText());
ress++;

//Y se le dice que debe ser guardada en la eqiqueta de resultado que en este caso es
nuestro
//txtField.
resultado.setText(String.valueOf(ress));

//Proceso
//Vamos a determinar si el filosofo esta comiendo esto para posteriormente mostrarlo
//en el txtArea.
proceso = "El filosofo " + (id + 1) + " esta comiendo\n";
txtArea.append(proceso);

//Lo que hacemos aqui es interrumpir el proceso por una cantidad de segundos
//que le asignemos //para posteriormente liberar los palillos
try {

    //duerme por 4 segundos
    Thread.sleep(1000);

} catch (InterruptedException e) {


}

//Una vez que el filosofo ya a comido este soltara los palillos y se podra de color verde
derecho.setText("Libre");
derecho.setForeground(Color.green);

izquierdo.setText("Libre");
izquierdo.setForeground(Color.green);

//Al igual cuando el filosofo deje de comer este pasara a un estado de pensamiento
//el cual se pondra de un color celeste
//Y mostraremos un mensaje en txtArea que el filosofo con el id tal libero el tenedor
//y con el metodo append mostramos en el txtArea.
filosofo.setText("Pensando");
filosofo.setBackground(Color.CYAN);
proceso = "El filosofo " + (id + 1) + " libero sus tenedores\n";
txtArea.append(proceso);

```

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

```

}

public void pensar() {

    //Para el metodo pensar lo que hemos hecho es que si el palillo derecho esta libre y el
derecho estan libres
    //este pasara a pensar
    derecho.setText("Libre");
    derecho.setForeground(Color.GREEN);

    izquierdo.setText("Libre");
    izquierdo.setForeground(Color.GREEN);

    filosofo.setText("Pensando");
    filosofo.setBackground(Color.CYAN);

    //Y le mandaremos un mensaje de que el filosofo esta pensando
    proceso = "El filosofo " + (id + 1) + " esta pensando\n";
    txtArea.append(proceso);

    //Y interrumpiremos el proceso por el tiempo que deseemos poner.
    //le estamos diciendo que se quede por un tiempo en el estado de pensar.
    try {

        Thread.sleep(1000);

    } catch (InterruptedException e) {
    }
}

```

## • CLASE VISTA CENA

```

package ec.ups.edu.vista;


import ec.ups.edu.filosofos.Filosofos;
import static java.lang.Thread.sleep;
import javax.swing.JLabel;
import javax.swing.JTextField;

/**
 *
 * @author santi
 */
public class VistaCena extends javax.swing.JFrame {

    Filosofos filosofo;

    public VistaCena() {
        initComponents();
    }

```

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

```

inicio();

}

public void inicio() {

    txtFilosofo1.setText("0");
    txtFilosofo2.setText("0");
    txtFilosofo3.setText("0");
    txtFilosofo4.setText("0");
    txtFilosofo5.setText("0");

}

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    lbl1 = new javax.swing.JLabel();
    lbl2 = new javax.swing.JLabel();
    lbl3 = new javax.swing.JLabel();
    lbl4 = new javax.swing.JLabel();
    lbl5 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    txtArea = new javax.swing.JTextArea();
    btnIniciar = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    txtFilosofo1 = new javax.swing.JTextField();
    txtFilosofo2 = new javax.swing.JTextField();
    txtFilosofo3 = new javax.swing.JTextField();
    txtFilosofo4 = new javax.swing.JTextField();
    txtFilosofo5 = new javax.swing.JTextField();
    jLabel6 = new javax.swing.JLabel();
    btnFinalizar = new javax.swing.JButton();
    jLabel8 = new javax.swing.JLabel();
    btnSalir = new javax.swing.JButton();
    lblFilosofo1 = new javax.swing.JLabel();
    lblFilosofo2 = new javax.swing.JLabel();
    lblFilosofo3 = new javax.swing.JLabel();
    lblFilosofo4 = new javax.swing.JLabel();
    lblFilosofo5 = new javax.swing.JLabel();


    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    lbl1.setText("Tenedor 1");

    lbl2.setText("Tenedor 2");

    lbl3.setText("Tenedor 3");

```

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

```

lbl4.setText("Tenedor 4");

lbl5.setText("Tenedor 5");

txtArea.setColumns(20);
txtArea.setRows(5);
jScrollPane.setViewportView(txtArea);

btnIniciar.setForeground(new java.awt.Color(204, 204, 204));
btnIniciar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ec/ups/edu/images/poder.png"))); // NOI18N
btnIniciar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnIniciarActionPerformed(evt);
    }
});

jLabel1.setText("Filosofo 2");

jLabel2.setText("Filosofo 1");

jLabel3.setText("Filosofo 3");

jLabel4.setText("Filosofo 4");

jLabel5.setText("Filosofo 5");

txtFilosofo1.setEditable(false);

txtFilosofo2.setEditable(false);

txtFilosofo3.setEditable(false);

txtFilosofo4.setEditable(false);

txtFilosofo5.setEditable(false);

jLabel6.setFont(new java.awt.Font("Arial Black", 1, 18)); // NOI18N
jLabel6.setText("FILOSOFOS COMENSALES");

btnFinalizar.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ec/ups/edu/images/boton-de-pausa.png"))); //
NOI18N
btnFinalizar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnFinalizarActionPerformed(evt);
    }
});

jLabel8.setText("Numero de veces que comieron");

btnSalir.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/ec/ups/edu/images/salida.png"))); // NOI18N
btnSalir.addActionListener(new java.awt.event.ActionListener() {

```

```

        public void actionPerformed(java.awt.ActionEvent evt) {
            btnSalirActionPerformed(evt);
        }
    });

    lblFilosofo1.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N
    lblFilosofo1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    lblFilosofo1.setText("Filosofo 1");
    lblFilosofo1.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    lblFilosofo2.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N
    lblFilosofo2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    lblFilosofo2.setText("Filosofo 2");
    lblFilosofo2.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    lblFilosofo3.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N
    lblFilosofo3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    lblFilosofo3.setText("Filosofo 3");
    lblFilosofo3.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    lblFilosofo4.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N
    lblFilosofo4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    lblFilosofo4.setText("Filosofo 4");
    lblFilosofo4.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    lblFilosofo5.setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N
    lblFilosofo5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    lblFilosofo5.setText("Filosofo 5");
    lblFilosofo5.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(btnIniciar)
                    .addComponent(btnFinalizar))
                .addGap(40, 40, 40)
                .addComponent(btnSalir, javax.swing.GroupLayout.PREFERRED_SIZE,
47, javax.swing.GroupLayout.PREFERRED_SIZE))
    );

```

```
.addGroup(layout.createSequentialGroup())

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup())

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
    .addComponent(lbl5)
    .addGroup(layout.createSequentialGroup())
        .addComponent(lbl4)
        .addGap(14, 14, 14))
    .addComponent(lblFilosofo5,
javax.swing.GroupLayout.PREFERRED_SIZE, 87, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())
        .addComponent(lblFilosofo4,
javax.swing.GroupLayout.PREFERRED_SIZE, 83, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(2, 2, 2)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup())
        .addGap(4, 4, 4)
        .addComponent(lbl3)
        .addGap(8, 8, 8)
        .addComponent(lblFilosofo3,
javax.swing.GroupLayout.PREFERRED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createSequentialGroup())
        .addComponent(lblFilosofo1,
javax.swing.GroupLayout.PREFERRED_SIZE, 87, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(lbl2)
    .addComponent(lbl1,
javax.swing.GroupLayout.PREFERRED_SIZE, 60, javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGroup(layout.createSequentialGroup())
        .addGap(110, 110, 110)
        .addComponent(lblFilosofo2,
javax.swing.GroupLayout.PREFERRED_SIZE, 84, javax.swing.GroupLayout.PREFERRED_SIZE)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup())
        .addGap(71, 71, 71)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())
        .addComponent(jLabel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 71, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```



```

        .addComponent(txtFilosofo4,
javax.swing.GroupLayout.PREFERRED_SIZE, 51, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())
        .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 71, javax.swing.GroupLayout.PREFERRED_SIZE)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(txtFilosofo3,
javax.swing.GroupLayout.PREFERRED_SIZE, 51, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())
        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 71, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(txtFilosofo2,
javax.swing.GroupLayout.PREFERRED_SIZE, 51, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())
        .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 71, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(txtFilosofo1,
javax.swing.GroupLayout.PREFERRED_SIZE, 51, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())
        .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 71, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(txtFilosofo5,
javax.swing.GroupLayout.PREFERRED_SIZE, 51, javax.swing.GroupLayout.PREFERRED_SIZE))))
        .addGroup(layout.createSequentialGroup()
            .addGap(32, 32, 32)
            .addComponent(jLabel8))))))
        .addGroup(layout.createSequentialGroup()
            .addGap(28, 28, 28)
            .addComponent(jScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 606,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup()
            .addGap(184, 184, 184)
            .addComponent(jLabel6)))
        .addContainerGap(29, Short.MAX_VALUE))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(19, 19, 19)
            .addComponent(jLabel6)
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()

```

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

```

        .addGap(9, 9, 9)
        .addComponent(jLabel8)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel2)
        .addComponent(txtFilosofo1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtFilosofo2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel11))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtFilosofo3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel3))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(txtFilosofo4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel4))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel5)
        .addComponent(txtFilosofo5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup())

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
                .addGap(21, 21, 21)


.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(lbl5, javax.swing.GroupLayout.PREFERRED_SIZE,
34, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(lbl11, javax.swing.GroupLayout.PREFERRED_SIZE,
41, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(lblFilosofo2,
javax.swing.GroupLayout.PREFERRED_SIZE, 31, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(lblFilosofo5,
javax.swing.GroupLayout.PREFERRED_SIZE, 31, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

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

```

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
    .addComponent(lbl4, GroupLayout.PREFERRED_SIZE,
34, GroupLayout.PREFERRED_SIZE)
    .addComponent(lbl2, GroupLayout.PREFERRED_SIZE,
38, GroupLayout.PREFERRED_SIZE)))
    .addComponent(lblFilosofo1, GroupLayout.PREFERRED_SIZE,
31, GroupLayout.PREFERRED_SIZE))

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()

.addPreferredGap(GroupLayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(lblFilosofo3,
javax.swing.GroupLayout.PREFERRED_SIZE, 31, GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createSequentialGroup()

.addPreferredGap(GroupLayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(lblFilosofo4,
javax.swing.GroupLayout.PREFERRED_SIZE, 31, GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createSequentialGroup()
        .addGap(18, 18, 18)
        .addComponent(lbl3, GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE))))
    .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jScrollPane, GroupLayout.PREFERRED_SIZE, 209,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(GroupLayoutStyle.ComponentPlacement.RELATED, 28,
Short.MAX_VALUE)
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
        .addComponent(btnIniciar)
        .addComponent(btnSalir)
        .addComponent(btnFinalizar))
    .addGap(15, 15, 15))
);


pack();
} // </editor-fold>

/**
 * Recursos.
 *
 * Se ha utilizado cinco botones que son para los filosofos Se ha utilizado
 * cinco label que son para los tenedores Y cinco campos de texto en donde
 * se guardara la informacion de cuantas veces a comido cada filosofo Ademas
 * se ha añadido un txtArea en el cual se ira mostrando lo que va haciendo
 * cada filosofo ya sea pensando comiendo o si soloto los palillos.
 *
 * @param evt
 */
private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {

    Filósofos comensales;

```

```
JLabel filosofo[];  
filosofo = new JLabel[5];  
filosofo[0] = lblFilosofo1;  
filosofo[1] = lblFilosofo2;  
filosofo[2] = lblFilosofo3;  
filosofo[3] = lblFilosofo4;  
filosofo[4] = lblFilosofo5;  
  
JLabel tenedor[];  
tenedor = new JLabel[5];  
tenedor[0] = lbl1;  
tenedor[1] = lbl2;  
tenedor[2] = lbl3;  
tenedor[3] = lbl4;  
tenedor[4] = lbl5;  
  
JTextField resultado[];  
resultado = new JTextField[5];  
resultado[0] = txtFilosofo1;  
resultado[1] = txtFilosofo2;  
resultado[2] = txtFilosofo3;  
resultado[3] = txtFilosofo4;  
resultado[4] = txtFilosofo5;  
  
int i;  
int izquierda;  
int derecha;  
  
//Hemos creado un for que recorra las posiciones de los tenedores  
//diciendo que mi i sea el derecho y mi i-1 mi tenedor izquierdo.  
//y se a aniadido un condicional en el que decimos que si el tendor  
//izquierdo es menor a cero este tomara la posicion 4, hacemos esto debido a  
//que el filosofo uno debe coger el tenedor de la posicion numero 5.  
for (i = 0; i < 5; i++) {  
  
    izquierda = i - 1;  
  
    if (izquierda < 0) {  
  
        //posicion 4  
        izquierda = 4;  
  
    }  
  
    //Aqui le estamos asignando que el tenedor derecho va a ser igual a i.  
    derecha = i;  
  
    //lo de comensales es por que estamos instanciando la clase Filsofos y pasandole los  
    datos que tenemos  
    //en el constructor.  
    //Le pasamos la posicion derecha, el tenedor de la izquierda el de la derecha  
    //los filsofos y los resultados, y un txtArea donde se mostrara la informacion.  
    comensales = new Filsofos(i, tenedor[izquierda], tenedor[derecha], filosofo[i],  
resultado[i], txtArea);
```

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

```

    }

}

private void btnFinalizarActionPerformed(java.awt.event.ActionEvent evt) {

    filosofo.pausar = true;

    try {

        sleep(5000);

    } catch (InterruptedException ex) {

        System.out.println("Error ");
        ex.printStackTrace();

    }

}

private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {


    System.exit(0);

}

/**
 * @param Filosofo1 args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
feel.
     * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

        java.util.logging.Logger.getLogger(VistaCena.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (InstantiationException ex) {

```

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

```

java.util.logging.Logger.getLogger(VistaCena.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (IllegalAccessException ex) {


java.util.logging.Logger.getLogger(VistaCena.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VistaCena.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    }
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new VistaCena().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton btnFinalizar;
private javax.swing.JButton btnIniciar;
private javax.swing.JButton btnSalir;
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.JLabel jLabel6;
private javax.swing.JLabel jLabel8;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JLabel lbl1;
private javax.swing.JLabel lbl2;
private javax.swing.JLabel lbl3;
private javax.swing.JLabel lbl4;
private javax.swing.JLabel lbl5;
private javax.swing.JLabel lblFilosofo1;
private javax.swing.JLabel lblFilosofo2;
private javax.swing.JLabel lblFilosofo3;
private javax.swing.JLabel lblFilosofo4;
private javax.swing.JLabel lblFilosofo5;
private javax.swing.JTextArea txtArea;
private javax.swing.JTextField txtFilosofo1;
private javax.swing.JTextField txtFilosofo2;
private javax.swing.JTextField txtFilosofo3;
private javax.swing.JTextField txtFilosofo4;
private javax.swing.JTextField txtFilosofo5;
// End of variables declaration
}

```

	<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		

### 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:

Como conclusión se puede decir que con este practica se ha podido practicar y entender sobre los temas de concurrencia, como entender las funcionalidades de sincronización y el uso de Thread dentro de Java.

### RECOMENDACIONES:

Recomiendo que se primero entendamos que es un thread, para que nos sirve como podemos usar y también revisar el tema de sincronización.

**Nombre de estudiante:** Jorge Santiago Cabrera Arias

**Firma de estudiante:**

