
	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.


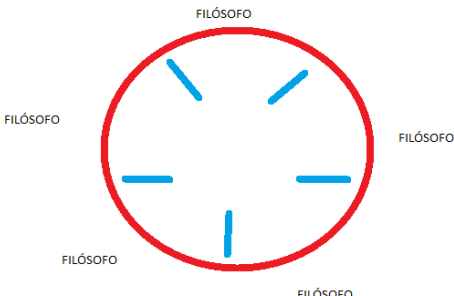
Alcance


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:			
<p>Problema del Filósofo:</p> <p>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».</p> <p>En general todos los objetos de la clase Filósofo está en un bucle infinito dedicándose a comer y a pensar.</p> <p>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.</p>			
			

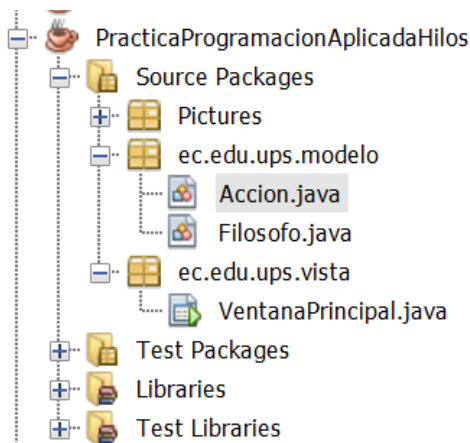
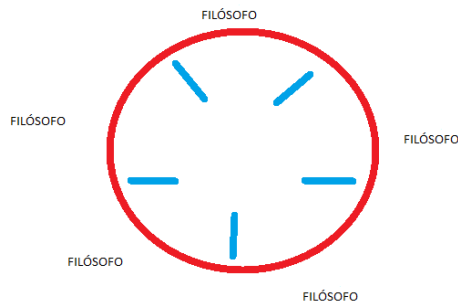
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

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
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: 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.			
ACTIVIDADES DESARROLLADAS			
1. 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.			



Paquetes y clases que conforman el proyecto de los filósofos

```
public void run() {
    boolean salir = true;
    while (salir==true) {
        pensando();
        accion.tomarTenedores(id);
        comer();
        accion.soltarTenedores(id);
    }
}
```


Método con el bucle infinito donde indica que el programa no va a parar y los filósofos van estar dedicados a comer y a pensar sin caer el problemas de sincronización porque los métodos tomar tenedor y soltar tenedor controlan esa parte.

Paquete Modelo

Clase Filosofo

```
public class Filosofo extends Thread{
    private int id;
    private Accion accion;
    //private int nComidas;
    private final Random randomico = new Random();
    VentanaPrincipal ventanaPrincipal;

    public Filosofo(int id, Accion a) {
```

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

```

this.id = id;
accion = a;
//nComidas = 0;

}

public void pensando() {
    System.out.println("Filosofo " + id + " pensando");
    ventanaPrincipal.filosofosStatus.get(id).setIcon(ventanaPrincipal.pensandoImg);
    try {
        Thread.sleep(randomico.nextInt(5000));
    } catch (Exception e) {
        System.out.println(e);
    }
}

public void comer() {
    System.out.println("Filosofo " + id + "comiendo");
    try {
        Thread.sleep(randomico.nextInt(5000));
    } catch (Exception e) {
        System.out.println(e);
    }
}

public void run() {
    boolean salir = true;
    while (salir==true) {
        pensando();
        accion.tomarTenedores(id);
        comer();
        accion.soltarTenedores(id);
    }
}
}

```


La clase filosofo hereda de la clase Thread de java la cual nos ayuda con la creación de hilos dentro de esta clase vamos a tener dos métodos el uno que va a hacer pensar y el otro comer los cuales van a tener un tiempo de espera randomico entre 1000 y 5000 milisegundos a su vez el método pensar la a cargar en la ventana principal la imagen cuando el filosofo se encuentre realizando ese procesos, finalmente como las clases hilos deben tener el método run que es donde vamos a ejecutar los procesos ubicamos que va hacer el filósofo y después gracias a la sincronización y los hilos vamos a ir controlando esas acciones con la clase Accion la cual nos va a indicar con los métodos tomar tenedor y soltar tenedor que el filósofo va a comer o a pensar.

Clase Acción

```

public class Accion {
    public static final int pensando = 0;
    public static final int hambriento = 1;
    public static final int comiendo = 2;
    private int n;
    private int estado[];
    VentanaPrincipal ventanaPrincipal;
}

```

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

```


public Accion(int n) {
    this.n = n;
    estado = new int[n];
}

public synchronized void tomarTenedores(int id) {
    estado[id] = hambriento;
    prueba(id);
    while (estado[id] != comiendo) {
        try {
            System.out.println("Filosofo " + id + " Esta esperando");
            wait();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

public synchronized void soltarTenedores(int id) {
    int izq, der;
    der = (id + 1) % n;
    izq = id - 1;
    if (izq < 0) {
        izq = n - 1;
    }
    ventanaPrincipal.tenedor(id, n);
    ventanaPrincipal.tenedor(der, n);
    estado[id] = pensando;
    ventanaPrincipal.filosofosStatus.get(id).setIcon(ventanaPrincipal.pensandoImg);
    prueba(izq);
    prueba(der);
}

public void prueba(int id) {
    int izq, der;
    der = (id + 1) % n;
    izq = id - 1;
    if (izq < 0) {
        izq = n - 1;
    }
    if (estado[id] == hambriento && estado[izq] != comiendo && estado[der] != comiendo) {
        ventanaPrincipal.tenedor.get(id).setIcon(null);
        ventanaPrincipal.tenedor.get(der).setIcon(null);
        estado[id] = comiendo;
        ventanaPrincipal.filosofosStatus.get(id).setIcon(ventanaPrincipal.comiendoImg);
        notifyAll();
    }
}
}

```

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

En la clase acción tenemos los siguientes métodos el tomar tenedor y soltar tenedor que nos va a indicar o nos da ese indicio si el filósofo va a comer o a pensar de acuerdo al estado en el que se encuentre el filósofo verificando también si el filósofo tiene a sus dos lados un tenedor para pasar a comer caso contrario seguirá pensando y para eso creamos el método prueba que nos va ayudar con esa parte de ahí cargando la imagen de filósofo cuando se encuentre comiendo y haciendo sus validaciones respectivas.

Paquete Vista

JFrame Vista Principal

```
package ec.edu.ups.vista;
import ec.edu.ups.modelo.Accion;
import ec.edu.ups.modelo.Filosofo;
import java.util.ArrayList;
import javax.swing.ImageIcon;
import javax.swing.JLabel;

/**
 *
 * @author NANCY
 */
public class VentanaPrincipal extends javax.swing.JFrame {

    static public ArrayList<JLabel> filosofosStatus = new ArrayList<JLabel>();
    static public ArrayList<JLabel> tenedor = new ArrayList<JLabel>();
    static public ArrayList<JLabel> comidas = new ArrayList<JLabel>();
    static public ArrayList<JLabel> filosofos = new ArrayList<JLabel>();
    int n;
    static public int numRand;
    static public ImageIcon pensandolmg = new ImageIcon("src/Pictures/pensando.png");
    static public ImageIcon comiendolmg = new ImageIcon("src/Pictures/comiendo.png");
    static public ImageIcon comida = new ImageIcon("src/Pictures/comida.png");
    static public ImageIcon uno = new ImageIcon("src/Pictures/1.png");
    static public ImageIcon dos = new ImageIcon("src/Pictures/2.png");
    static public ImageIcon tres = new ImageIcon("src/Pictures/3.png");
    static public ImageIcon cuatro = new ImageIcon("src/Pictures/4.png");
    static public ImageIcon cinco = new ImageIcon("src/Pictures/5.png");
    static public ImageIcon seis = new ImageIcon("src/Pictures/6.png");
    static public ImageIcon siete = new ImageIcon("src/Pictures/7.png");
    static public ImageIcon ocho = new ImageIcon("src/Pictures/8.png");
    static public ImageIcon nueve = new ImageIcon("src/Pictures/9.png");
    static public ImageIcon diez = new ImageIcon("src/Pictures/10.png");

    /**
     * Creates new form VentanaPrincipal
     */
    public VentanaPrincipal() {
        initComponents();
        this.setLocationRelativeTo(null);
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        //Se inhabilita el boton para evitar errores por el usuario
        jButton1.setEnabled(false);
        try {
            //Se crean los hilos

```



```
n = Integer.parseInt(nFilosofos.getSelectedItem().toString());
Accion c = new Accion(n);
Filosofo f[] = new Filosofo[n];
Filosofos(n);
for (int cont = 0; cont < n; cont++) {
    filosofos.get(cont).setText((cont + 1) + "");
    filosofosStatus.get(cont).setIcon(pensandoImg);
    tenedor(cont, n);
    comidas.get(cont).setIcon(comida);
    f[cont] = new Filosofo(cont, c);
    f[cont].start();
}

} catch (Exception e) {
    System.out.println("");
}

}

public void Filosofos(int n) {
    //Asina cad filosofo a un JLabel en especifico
    switch (n) {
        case 5:
            filosofosStatus.add(f1);
            filosofosStatus.add(f3);
            filosofosStatus.add(f5);
            filosofosStatus.add(f7);
            filosofosStatus.add(f9);
            tenedor.add(tenedor1);
            tenedor.add(tenedor3);
            tenedor.add(tenedor5);
            tenedor.add(tenedor7);
            tenedor.add(tenedor9);

            comidas.add(comida1);
            comidas.add(comida3);
            comidas.add(comida5);
            comidas.add(comida7);
            comidas.add(comida9);

            filosofos.add(n1);
            filosofos.add(n3);
            filosofos.add(n5);
            filosofos.add(n7);
            filosofos.add(n9);

            break;
        case 8:
            filosofosStatus.add(f1);
            filosofosStatus.add(f2);
            filosofosStatus.add(f3);

            filosofosStatus.add(f5);
            filosofosStatus.add(f6);
```

```
filosofosStatus.add(f7);  
filosofosStatus.add(f8);  
filosofosStatus.add(f9);
```

```
tenedor.add(tenedor1);  
tenedor.add(tenedor2);  
tenedor.add(tenedor3);  
tenedor.add(tenedor5);  
tenedor.add(tenedor6);  
tenedor.add(tenedor7);  
tenedor.add(tenedor8);  
tenedor.add(tenedor9);
```

```
comidas.add(comida1);  
comidas.add(comida2);  
comidas.add(comida3);  
comidas.add(comida5);  
comidas.add(comida6);  
comidas.add(comida7);  
comidas.add(comida8);  
comidas.add(comida9);
```

```
filosofos.add(n1);  
filosofos.add(n2);  
filosofos.add(n3);  
filosofos.add(n5);  
filosofos.add(n6);  
filosofos.add(n7);  
filosofos.add(n8);  
filosofos.add(n9);
```

```
break;  
case 10:  
    filosofosStatus.add(f1);  
    filosofosStatus.add(f2);  
    filosofosStatus.add(f3);  
    filosofosStatus.add(f4);  
    filosofosStatus.add(f5);  
    filosofosStatus.add(f6);  
    filosofosStatus.add(f7);  
    filosofosStatus.add(f8);  
    filosofosStatus.add(f9);  
    filosofosStatus.add(f10);
```

```
tenedor.add(tenedor1);  
tenedor.add(tenedor2);  
tenedor.add(tenedor3);  
tenedor.add(tenedor4);  
tenedor.add(tenedor5);  
tenedor.add(tenedor6);  
tenedor.add(tenedor7);  
tenedor.add(tenedor8);  
tenedor.add(tenedor9);  
tenedor.add(tenedor10);
```

```
comidas.add(comida1);
comidas.add(comida2);
comidas.add(comida3);
comidas.add(comida4);
comidas.add(comida5);
comidas.add(comida6);
comidas.add(comida7);
comidas.add(comida8);
comidas.add(comida9);
comidas.add(comida10);
```

```
filosofos.add(n1);
filosofos.add(n2);
filosofos.add(n3);
filosofos.add(n4);
filosofos.add(n5);
filosofos.add(n6);
filosofos.add(n7);
filosofos.add(n8);
filosofos.add(n9);
filosofos.add(n10);
break;
```

```
default:
    System.out.println("Ninguno de los anteriores");
```

```
}
}
```

```
public static void tenedor(int id, int numF) {
```

```
    switch (numF) {
```

```
        case 5:
```

```
            // -----
```

```
            switch (id) {
```

```
                case 0:
```

```
                    tenedor.get(0).setlcon(unos);
```

```
                    break;
```

```
                case 1:
```

```
                    tenedor.get(1).setlcon(tres);
```

```
                    break;
```

```
                case 2:
```

```
                    tenedor.get(2).setlcon(cinco);
```

```
                    break;
```

```
                case 3:
```

```
                    tenedor.get(3).setlcon(siete);
```

```
                    break;
```

```
                case 4:
```

```
                    tenedor.get(4).setlcon(nueve);
```

```
                    break;
```

```
            default:
```

```
                System.out.println("Ninguno de los anteriores");
```

```
}

//-----
break;
case 8:

switch (id) {
    case 0:
        tenedor.get(0).setlcon(unos);
        break;
    case 1:
        tenedor.get(1).setlcon(dos);
        break;
    case 2:
        tenedor.get(2).setlcon(tres);
        break;
    case 3:
        tenedor.get(3).setlcon(cinco);
        break;
    case 4:
        tenedor.get(4).setlcon(seis);
        break;
    case 5:
        tenedor.get(5).setlcon(siete);
        break;
    case 6:
        tenedor.get(6).setlcon(ocho);
        break;
    case 7:
        tenedor.get(7).setlcon(nueve);

        break;

    default:
        System.out.println("Ninguno de los anteriores");
}

break;
case 10:

switch (id) {
    case 0:
        tenedor.get(0).setlcon(unos);
        break;
    case 1:
        tenedor.get(1).setlcon(dos);
        break;
    case 2:
        tenedor.get(2).setlcon(tres);
        break;
    case 3:
        tenedor.get(3).setlcon(cuatro);
        break;
    case 4:
```

```
        tenedor.get(4).setIcon(cinco);
        break;
    case 5:
        tenedor.get(5).setIcon(seis);
        break;
    case 6:
        tenedor.get(6).setIcon(siete);
        break;
    case 7:
        tenedor.get(7).setIcon(ocho);

        break;
    case 8:
        tenedor.get(8).setIcon(nueve);

        break;
    case 9:
        tenedor.get(9).setIcon(diez);


        break;

    default:
        System.out.println("Ninguno de los anteriores");
}

break;

default:
    System.out.println("Ninguno de los anteriores");
}

}
/**
 * @param 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(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    }
}
```

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

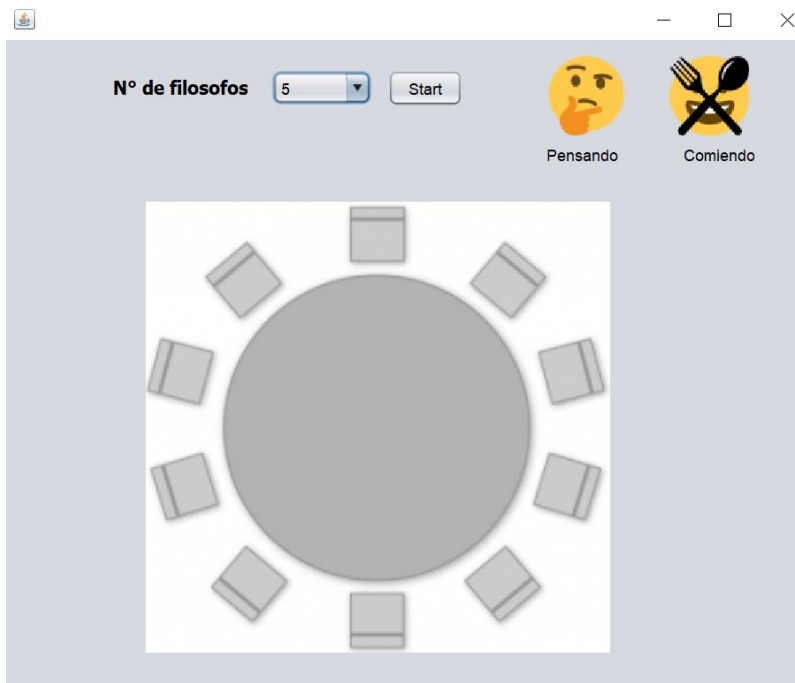
```

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

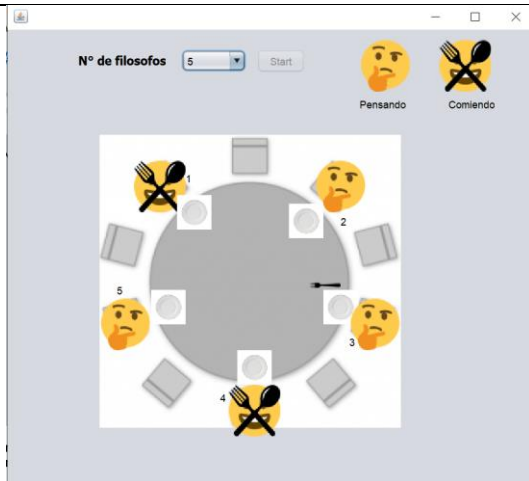
```

En la ventana principal se crean ya los hilos al momento presionar el Botón de inicio para que así pueda funcionar el programa de una manera sincronizada

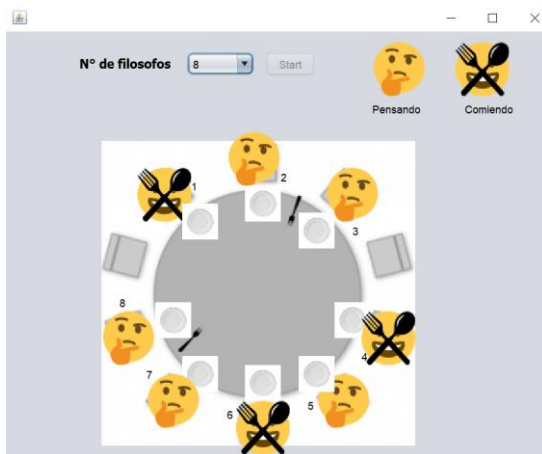


Si le damos en ejecutar el programa se nos abre la siguiente ventana en donde vamos a tener que seleccionar cuantos filósofos vamos a querer para hacer que funcione el programa por defecto ya son 5

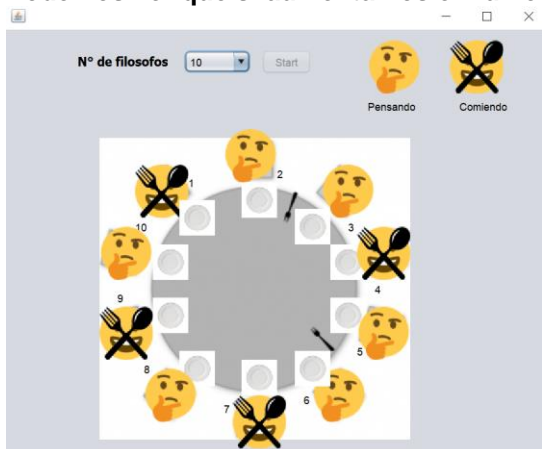
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		




Al presionar Start se bloqueará el botón y comenzará a ejecutar el programa de una manera infinita



Podemos ver que si aumentamos el número de filósofos a ocho también sirve



También funciona si aumentamos el numero a 10 que es el máximo de filósofos que se pueden aumentar

	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 número de filósofos.

En la parte de la vista en el código se puede encontrar los métodos filósofo y tenedor los cuales van a aumentar el número de filósofos y tenedores de acuerdo con lo que seleccione el usuario

```
public void Filósofos(int n) {
    //Asina cad filosofo a un JLabel en especifico
    switch (n) {
        case 5:
            filosofosStatus.add(f1);
            filosofosStatus.add(f3);
            filosofosStatus.add(f5);
            filosofosStatus.add(f7);
            filosofosStatus.add(f9);
            tenedor.add(tenedor1);
            tenedor.add(tenedor3);
            tenedor.add(tenedor5);
            tenedor.add(tenedor7);
            tenedor.add(tenedor9);

            comidas.add(comida1);
            comidas.add(comida3);
            comidas.add(comida5);
            comidas.add(comida7);
            comidas.add(comida9);

            filosofos.add(n1);
            filosofos.add(n3);
            filosofos.add(n5);
            filosofos.add(n7);
            filosofos.add(n9);

            break;
        case 8:
            filosofosStatus.add(f1);
            filosofosStatus.add(f2);
            filosofosStatus.add(f3);

            filosofosStatus.add(f5);
            filosofosStatus.add(f6);
            filosofosStatus.add(f7);
            filosofosStatus.add(f8);
            filosofosStatus.add(f9);

            tenedor.add(tenedor1);
            tenedor.add(tenedor2);
            tenedor.add(tenedor3);
            tenedor.add(tenedor5);
            tenedor.add(tenedor6);
            tenedor.add(tenedor7);
            tenedor.add(tenedor8);
            tenedor.add(tenedor9);
```



```
comidas.add(comida1);
comidas.add(comida2);
comidas.add(comida3);
comidas.add(comida5);
comidas.add(comida6);
comidas.add(comida7);
comidas.add(comida8);
comidas.add(comida9);

filosofos.add(n1);
filosofos.add(n2);
filosofos.add(n3);
filosofos.add(n5);
filosofos.add(n6);
filosofos.add(n7);
filosofos.add(n8);
filosofos.add(n9);

break;
case 10:
    filosofosStatus.add(f1);
    filosofosStatus.add(f2);
    filosofosStatus.add(f3);
    filosofosStatus.add(f4);
    filosofosStatus.add(f5);
    filosofosStatus.add(f6);
    filosofosStatus.add(f7);
    filosofosStatus.add(f8);
    filosofosStatus.add(f9);
    filosofosStatus.add(f10);

    tenedor.add(tenedor1);
    tenedor.add(tenedor2);
    tenedor.add(tenedor3);
    tenedor.add(tenedor4);
    tenedor.add(tenedor5);
    tenedor.add(tenedor6);
    tenedor.add(tenedor7);
    tenedor.add(tenedor8);
    tenedor.add(tenedor9);
    tenedor.add(tenedor10);

    comidas.add(comida1);
    comidas.add(comida2);
    comidas.add(comida3);
    comidas.add(comida4);
    comidas.add(comida5);
    comidas.add(comida6);
    comidas.add(comida7);
    comidas.add(comida8);
    comidas.add(comida9);
    comidas.add(comida10);

    filosofos.add(n1);
```

```
filosofos.add(n2);
filosofos.add(n3);
filosofos.add(n4);
filosofos.add(n5);
filosofos.add(n6);
filosofos.add(n7);
filosofos.add(n8);
filosofos.add(n9);
filosofos.add(n10);
break;

default:
    System.out.println("Ninguno de los anteriores");
}
}

public static void tenedor(int id, int numF) {

    switch (numF) {

        case 5:

            // -----
            switch (id) {
                case 0:
                    tenedor.get(0).setIcon(unos);
                    break;
                case 1:
                    tenedor.get(1).setIcon(tres);
                    break;
                case 2:
                    tenedor.get(2).setIcon(cinco);
                    break;
                case 3:
                    tenedor.get(3).setIcon(siete);
                    break;
                case 4:
                    tenedor.get(4).setIcon(nueve);
                    break;
                default:
                    System.out.println("Ninguno de los anteriores");
            }

            //-----
            break;
        case 8:

            switch (id) {
                case 0:
                    tenedor.get(0).setIcon(unos);
                    break;
                case 1:
                    tenedor.get(1).setIcon(dos);
                    break;
```

```
case 2:
    tenedor.get(2).setlcon(tres);
    break;
case 3:
    tenedor.get(3).setlcon(cinco);
    break;
case 4:
    tenedor.get(4).setlcon(seis);
    break;
case 5:
    tenedor.get(5).setlcon(siete);
    break;
case 6:
    tenedor.get(6).setlcon(ocho);
    break;
case 7:
    tenedor.get(7).setlcon(nueve);


    break;

default:
    System.out.println("Ninguno de los anteriores");
}

break;
case 10:

switch (id) {
    case 0:
        tenedor.get(0).setlcon(unos);
        break;
    case 1:
        tenedor.get(1).setlcon(dos);
        break;
    case 2:
        tenedor.get(2).setlcon(tres);
        break;
    case 3:
        tenedor.get(3).setlcon(cuatro);
        break;
    case 4:
        tenedor.get(4).setlcon(cinco);
        break;
    case 5:
        tenedor.get(5).setlcon(seis);
        break;
    case 6:
        tenedor.get(6).setlcon(siete);
        break;
    case 7:
        tenedor.get(7).setlcon(ocho);

        break;
    case 8:
```

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

```

        tenedor.get(8).setIcon(nueve);

        break;
    case 9:
        tenedor.get(9).setIcon(diez);

        break;

    default:
        System.out.println("Ninguno de los anteriores");
    }

    break;

    default:
        System.out.println("Ninguno de los anteriores");
    }

}

```

RESULTADO(S) OBTENIDO(S):

Conocemos el manejo de hilos para agilizar los procesos a la hora de ejecutar el programa lo hagan de una manera más fácil y rápida.

CONCLUSIONES:

En Conclusión, esta practica nos ayuda a entender de mejor manera los hilos y como pueden trabajar con varios procesos a la ves de una manera sincronizada y fácil

RECOMENDACIONES:

No hay recomendaciones

Estudiantes: Romel Ávila

Firma:

