

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

		PRÁCTICA DE LABORATORIO	
CARRERA: COMPUTACION		ASIGNATURA: PROGRAMACION APLICADA	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Hilos en Java	
OBJETIVO ALCANZADO: Identificar los cambios importantes de Java Diseñar e Implementar las nuevas técnicas de programación concurrente Entender cada una de las características de Thread en Java.			
ACTIVIDADES DESARROLLADAS			
<p>1. Revisar la teoría y conceptos de Thread en Java.</p> <p>La Máquina Virtual Java (JVM) es un sistema multihilo. Es decir, es capaz de ejecutar varios hilos de ejecución simultáneamente. La JVM gestiona todos los detalles, asignación de tiempos de ejecución, prioridades, etc., de forma similar a como gestiona un Sistema Operativo múltiples procesos. La diferencia básica entre un proceso de Sistema Operativo y un Thread Java es que los hilos corren dentro de la JVM, que es un proceso del Sistema Operativo y por tanto comparten todos los recursos, incluida la memoria y las variables y objetos allí definidos. A este tipo de procesos donde se comparte los recursos se les llama a veces procesos ligeros (lightweight process).</p> <p>Java da soporte al concepto de Thread desde el propio lenguaje, con algunas clases e interfaces definidas en el paquete java.lang y con métodos específicos para la manipulación de Threads en la clase Object.</p> <p>Desde el punto de vista de las aplicaciones los hilos son útiles porque permiten que el flujo del programa sea dividido en dos o más partes, cada una ocupándose de alguna tarea de forma independiente. Por ejemplo, un hilo puede encargarse de la comunicación con el usuario, mientras que otros actúan en segundo plano, realizando la transmisión de un fichero, accediendo a recursos del sistema (cargar sonidos, leer ficheros ...), etc. De hecho, todos los programas con interface gráfico (AWT o Swing) son multihilo porque los eventos y las rutinas de dibujo de las ventanas corren en un hilo distinto al principal.</p>			

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

La forma más directa para hacer un programa multihilo es extender la clase Thread, y redefinir el método run(). Este método es invocado cuando se inicia el hilo (mediante una llamada al método start() de la clase Thread). El hilo se inicia con la llamada al método run() y termina cuando termina éste.

La interface 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 extender alguna otra clase. Dado que no existe herencia múltiple, la citada clase no puede extender a la vez la clase Thread y otra más. En este caso, la clase debe implantar la interface Runnable, variando ligeramente la forma en que se crean e inician los nuevos hilos.

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 están 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.

Solución:

Para la solución de este problema se procedió a implementar tanto la interfaz Runnable y la clase Thread. Estos nos ayudan para secuencia de hilos en el programa y que se produzcan asincrónicamente.

Para la parte grafica se utilizaron distintas clases como: Graphics2d que nos permite realizar dibujos y gráficos en 2d en un panel; AffineTransform que nos permite realizar gráficos rectos y paralelos de líneas, es decir que en el caso de dibujar una mesa no salga pixelada en las esquinas, RenderingHints nos permite renderizar imágenes para que sean de mejor calidad y se adapten al frame, y también nos permite manipular gráficos 2d.

Se necesita crear una clase filósofo la cual hereda de la clase Thread para usar su método run(). En esta clase se definen sus características que nos van a ayudar a crear un programa que tenga concordancia. Luego se creo una clase la cual tiene los tiempos mínimos y máximo que el hilo va a pensar, comer, hambre, y dormir. En esta

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

misma clase se guardaron los colores en formato RGB. Luego nos creamos dos enums: el primer enum es para el estado del filósofo y el siguiente enum es para saber el estado del tenedor.

Para la parte gráfica, se creó una clase la cual es encargada de crear los filósofos, tenedores y la mesa, las cuales son añadidas a un panel. Luego se creó una Interfaz para poder controlar cuando un filósofo puede comer y otros van a tener que saltarse alguna comida. Luego se creó la clase Ventana la cual hereda de JFrame y nos sirve para que ahí se guarden todos los componentes. Y por último, la clase principal nos sirve para que la clase main y poder ejecutar el programa.

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

A continuación, se presenta el segmento del código de la clase controlador donde se asigna el número de filósofos y tenedores que tiene el programa:


```
public void inicializador(int numeroFilosofos) {
    this.numeroFilosofos = numeroFilosofos;
}

public void comenzarSimulacion() {
    if (this.isComenzar) {
        this.pararSimulacion();
    }

    // inicializamos los arrays
    this.tenedores = new boolean[numeroFilosofos];
    this.filosofos = new Filosofo[numeroFilosofos];

    // inicializamos los tenedores
    for (int i = 0; i < numeroFilosofos; i++) {
        this.tenedores[i] = true;
    }

    // creamos los filosofos
    Filosofo filosofo;
```

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

```

for (int i = 0; i < numeroFilosofos; i++) {
    filosofo = new Filosofo(this, i);
    filosofo.comenzar();

    this.filosofos[i] = filosofo;
}

this.isComenzar = true;
}

```

Ahora se presenta el método utilizado en la GraficarComedor donde se grafican los filósofos:

```


public void disenioFilosofos(int i, Graphics2D grafico2d) {
    // color
    if (this.estados[i] != null) {
        grafico2d.setPaint(ConstantesColorTiempo.ESTADOS_COLOR[this.estados[i].getId()]);
    } else {
        grafico2d.setPaint(getBackground());
    }

    // tamaño
    double size = Math.min(6 / Math.pow(numeroFilosofos, .8), 1.5);
    Ellipse2D.Double formaEliptica = new Ellipse2D.Double(3, -size / 2, size, size);

    // color base
    if (hilo == null || !hilo.isAlive()) {
        grafico2d.setColor(new Color(55, 55, 5));
    }

    // dibujar filosofos

```

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

```
grafico2d.fill(formaEliptica);
grafico2d.setPaint(Color.black);
grafico2d.draw(formaEliptica);
//para que el grafico vaya de lado
grafico2d.rotate(2 * Math.PI / numeroFilosofos);
}
```

Ahora se presenta el método utilizado en la GraficarComedor donde se grafican los tenedores:

```
public void disenioTenedor(int i, Graphics2D grafico2d) {
    // determinar posicion
    double posicion = (2 * i - 1) * Math.PI / numeroFilosofos;


    EstadoTenedor estado = this.getEstadoTenedor(i);
    float redondoFilo = 0;

    if (estado == EstadoTenedor.USANDO_IZQUIERDA) {
        redondoFilo = -.8f;
    } else if (estado == EstadoTenedor.USANDO_DERECHA) {
        redondoFilo = .8f;
    }

    posicion = posicion + redondoFilo * (Math.PI / (2 * numeroFilosofos));

    //longitud de tenedores
    double longitudTenedor = (2.0 * 3) / numeroFilosofos;

    // grafico tenedores
    grafico2d.setPaint(ConstantesColorTiempo.COLOR_TENEDOR);
    //para que los tenedores vayan de lado
```

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

```

grafico2d.rotate(posicion);


//afinar la grafica
grafico2d.setStroke(new BasicStroke((float) (.3 / Math.pow(numeroFilosofos, .55))));

if (estado == EstadoTenedor.USANDO_IZQUIERDA || estado == EstadoTenedor.USANDO_DERECHA) {
    //mover los tenedores al lado del filosofo
    grafico2d.draw(new Line2D.Double(3, 0, 3 + longitudTenedor, 0));
} else {
    //poner tenedores en la mesa
    grafico2d.draw(new Line2D.Double(2.3 - longitudTenedor, 0, 2.3, 0));
}

grafico2d.rotate(-posicion);
}

```

A continuación, se presenta la vista gráfica cuando se inicia la aplicación:

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

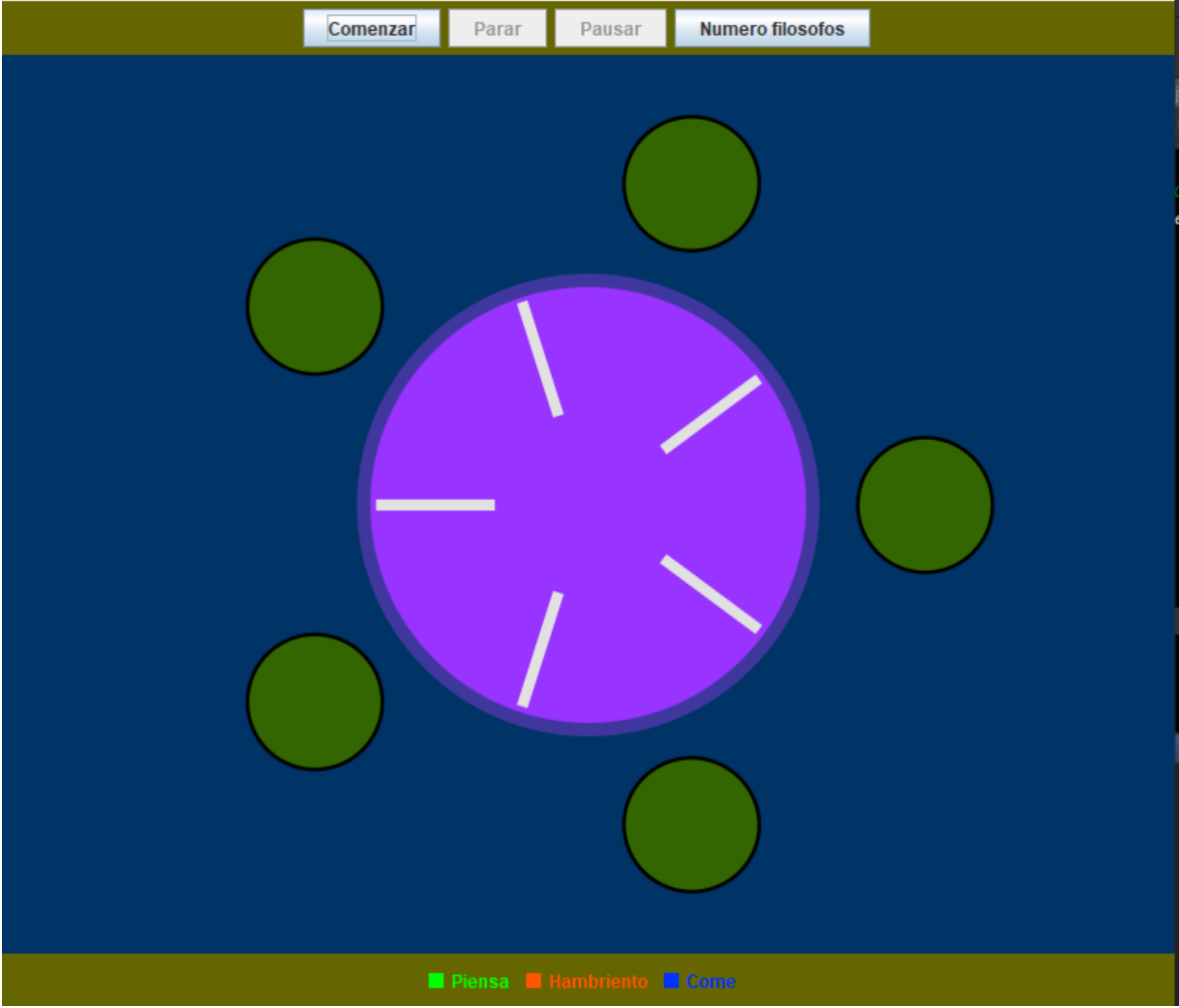
Filósofos

Comenzar

Parar

Pausar

Numero filosofos




Piensa

Hambriento

Come

Ahora le damos click al botón de comenzar:

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

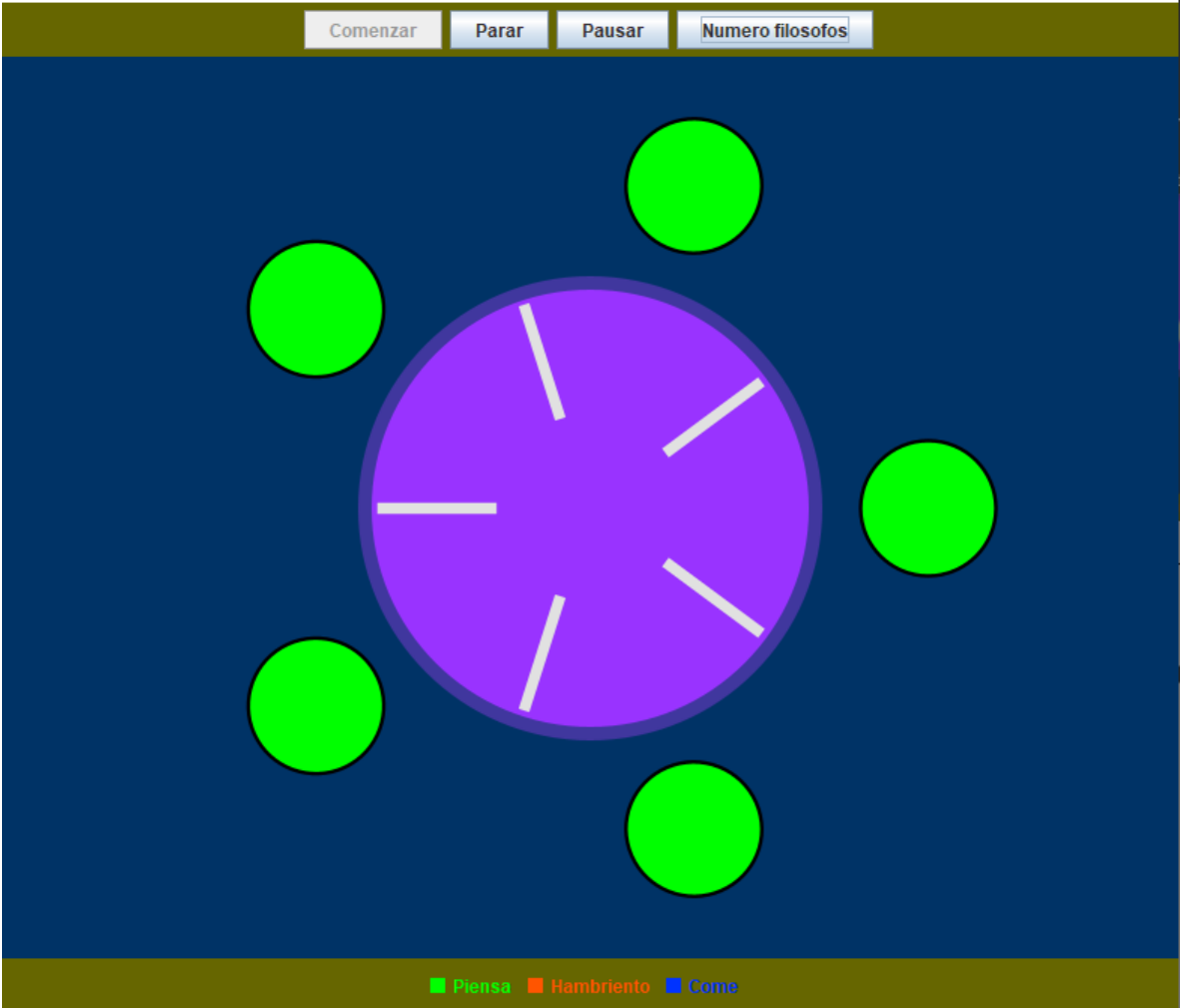
Filósofos

Comenzar

Parar

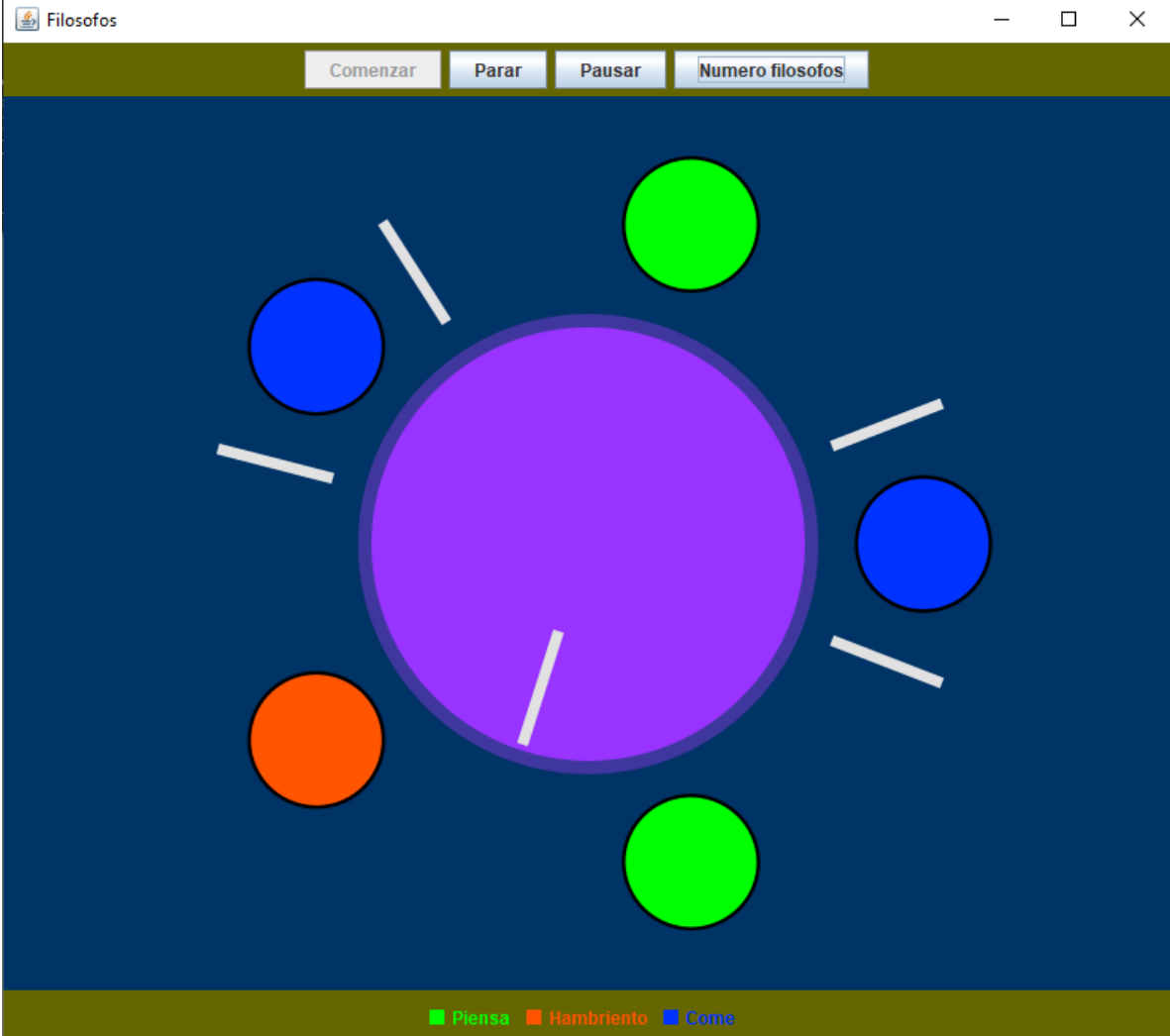
Pausar

Numero filosofos




■ Piensa
■ Hambriento
■ Come

Como observamos todos los filósofos comienzan en modo” Piensa”. Ahora una foto pasado unos segundos:



Ahora cambiaremos el número de filósofos que existen. Cabe recalcar que el máximo de filósofos que acepta el programa es de 30, y el mínimo de 3. Pondremos ahora 15 filósofos:

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

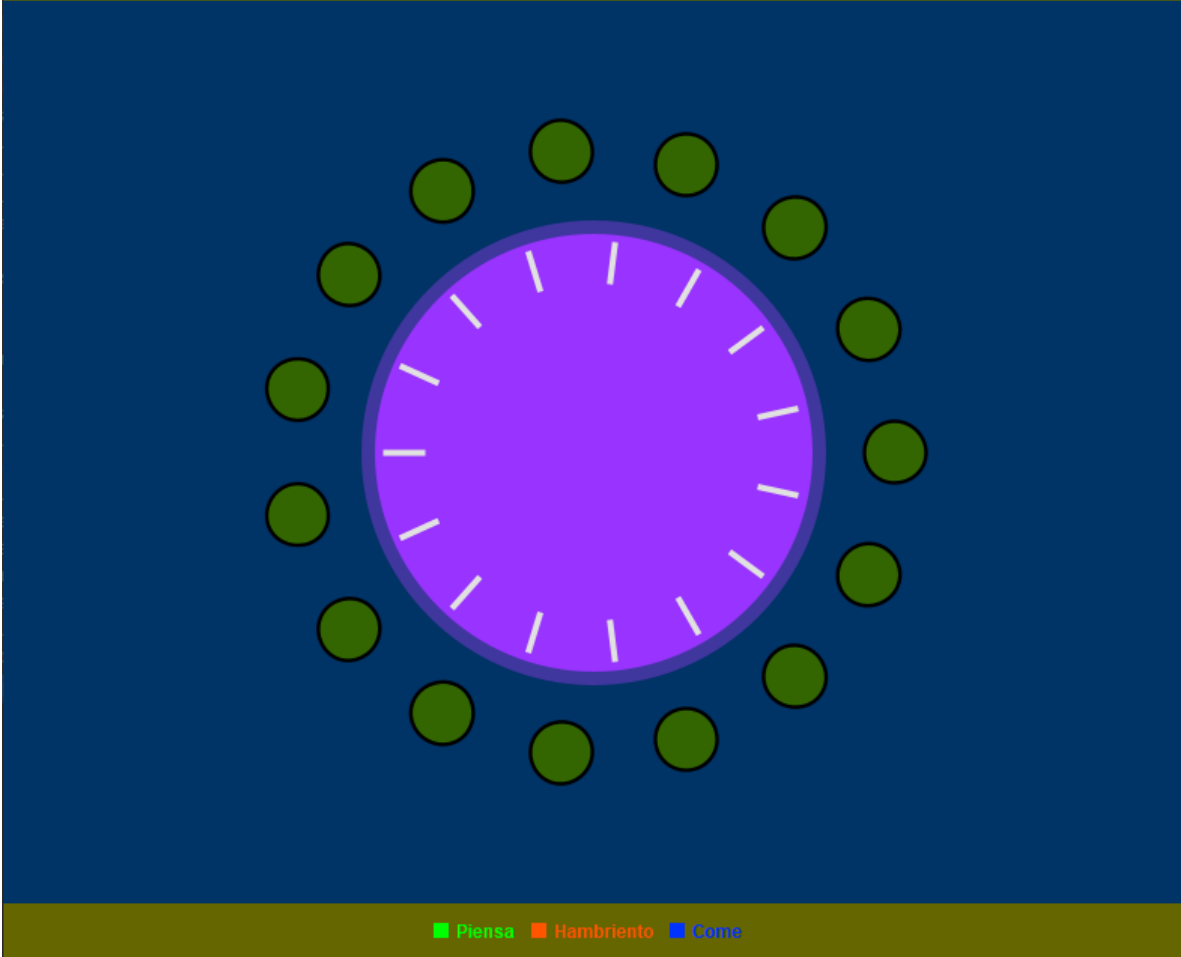
Filósofos

Comenzar

Parar

Pausar

Numero filosofos




Piensa

Hambriento

Come

Si iniciamos la simulación todos los filósofos comenzaran con el estado de pensar, por lo que la imagen que se presentara a continuación mostrara la simulación después de unos segundos:

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

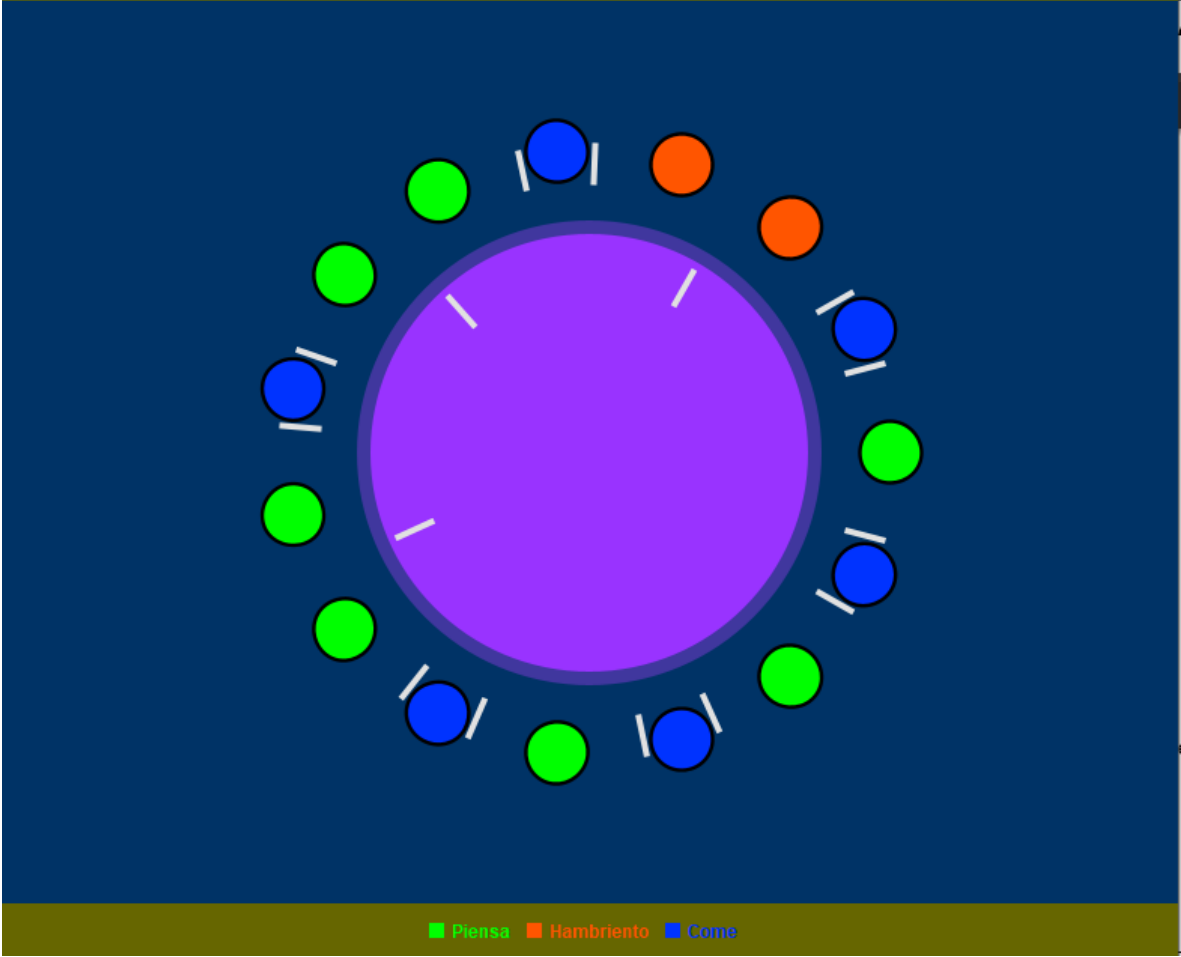
Filósofos

Comenzar

Parar

Reanudar

Numero filosofos

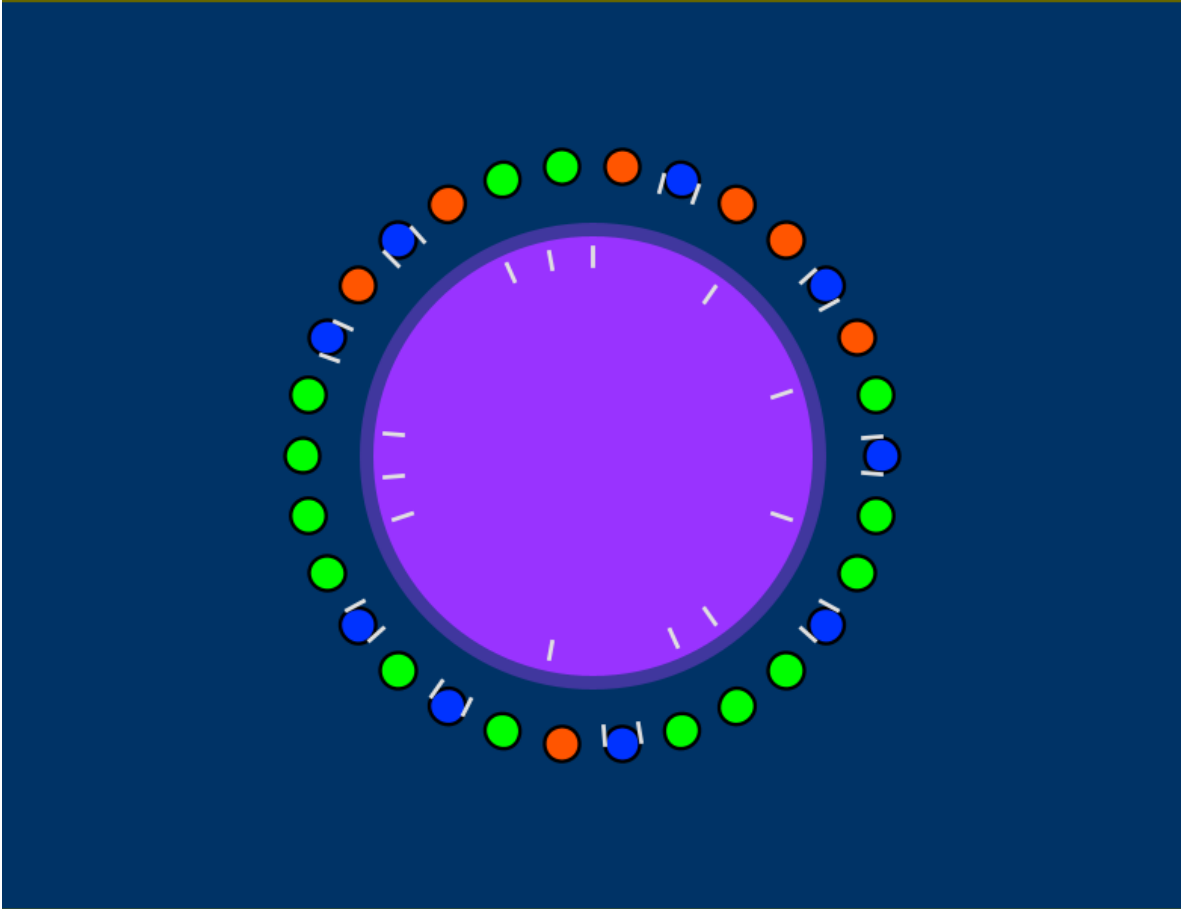


■ Piensa
■ Hambriento
■ Come

Como podemos observar hay varios filósofos que piensan y otros que comen y por último otros que están hambrientos. A continuación, mostrare una imagen con la simulación de 30 filósofos:

Filosofos

Comenzar Parar Reanudar Numero filosofos

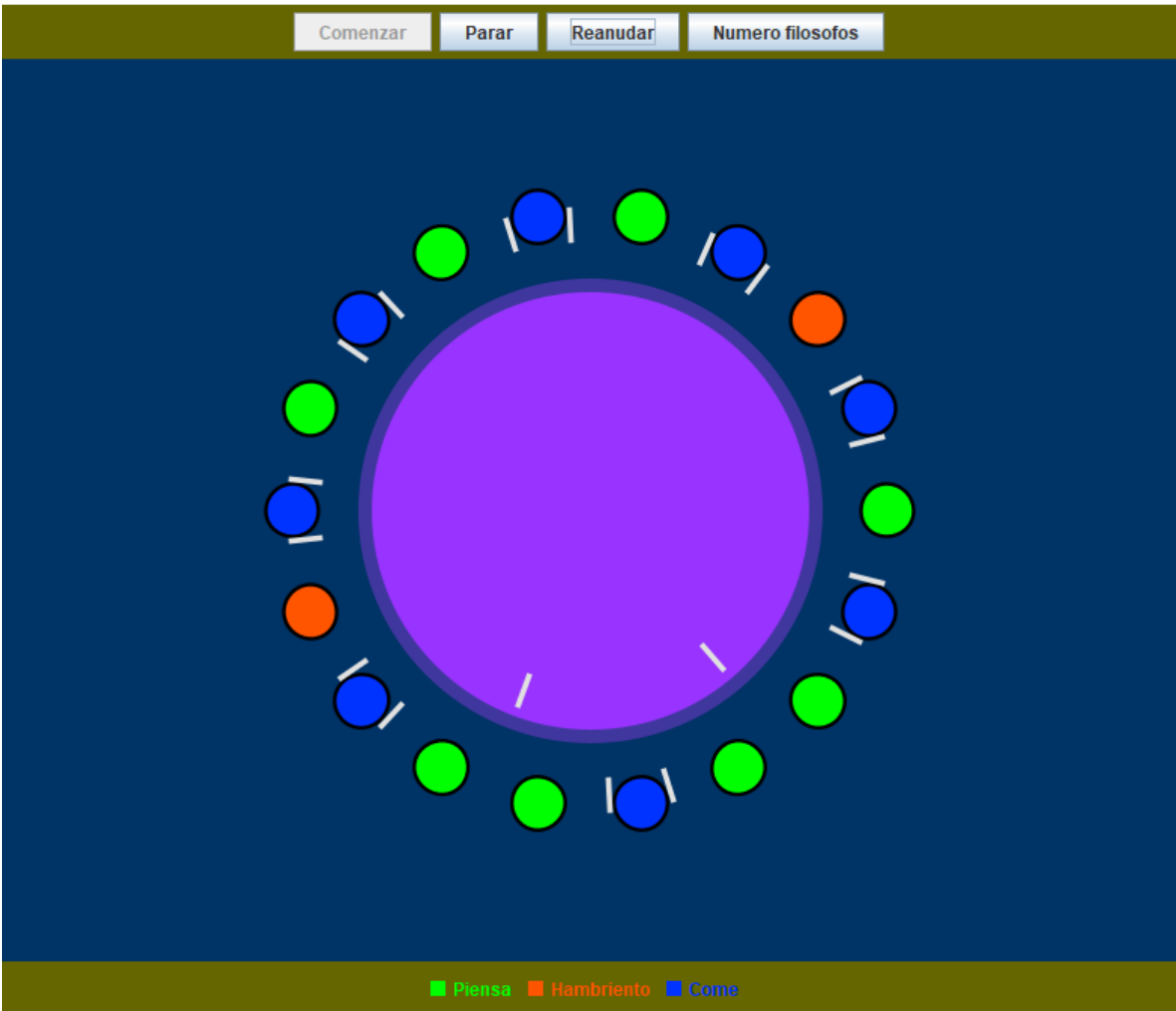


■ Piensa ■ Hambriento ■ Come

Simulación de 18 filósofos:


Filosofos

Comenzar Parar Reanudar Numero filosofos



■ Piensa ■ Hambriento ■ Come

Por último, se presentará un video de cómo funciona la aplicación:

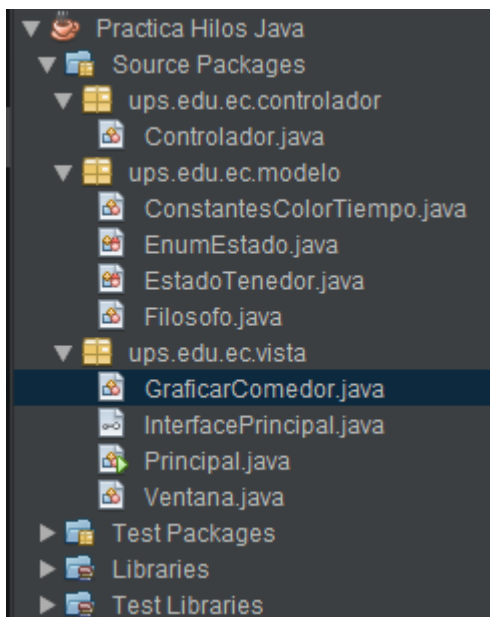
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



video demostracion

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

Paquetes creados:




clase controlador:

/*

* 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.

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

```

*/
package ups.edu.ec.controlador;

import java.util.Random;
import ups.edu.ec.modelo.Filosofo;
import ups.edu.ec.vista.Principal;

/**
 *
 * @author User
 */
public class Controlador {

    public final static Random RANDOMICO = new Random();

    private Principal principal;
    private boolean isComenzar;
    private boolean isPausar;
    private boolean[] tenedores;
    private Filosofo[] filosofos;
    private int numeroFilosofos;

    public Controlador(Principal principal) {
        this.principal = principal;
        tenedores = null;
        filosofos = null;
    }

    public Principal getPrincipal() {
        return this.principal;
    }

```

```
}

public int getNumeroFilosofos() {
    return this.numeroFilosofos;
}

public boolean getTenedores(int numeroTenedores) {
    return this.tenedores[numeroTenedores];
}

public Filosofo[] getVecinos(Filosofo filo) {
    int izquierda = (filo.getId() - 1) % this.getNumeroFilosofos();
    int derecha = (filo.getId() + 1) % this.getNumeroFilosofos();

    if (izquierda < 0) {
        izquierda += this.getNumeroFilosofos();
    }

    return new Filosofo[]{
        this.filosofos[izquierda],
        this.filosofos[derecha]
    };
}

public void setTenedor(int numeroTenedore, boolean valor) {
    this.tenedores[numeroTenedore] = valor;
}

public void inicializador(int numeroFilosofos) {
```



```
this.numeroFilosofos = numeroFilosofos;

}

public void comenzarSimulacion() {
    if (this.isComenzar) {
        this.pararSimulacion();
    }

    // inicializamos los arrays
    this.tenedores = new boolean[numeroFilosofos];
    this.filosofos = new Filosofo[numeroFilosofos];


    // inicializamos los tenedores
    for (int i = 0; i < numeroFilosofos; i++) {
        this.tenedores[i] = true;
    }

    // creamos los filosofos
    Filosofo filosofo;
    for (int i = 0; i < numeroFilosofos; i++) {
        filosofo = new Filosofo(this, i);
        filosofo.comenzar();

        this.filosofos[i] = filosofo;
    }

    this.isComenzar = true;
}
```

```
public void pararSimulacion() {  
    if (this.isComenzar) {  
  
        for (int i = 0; i < numeroFilosofos; i++) {  
            this.filosofos[i].parar();  
        }  
  
        this.isComenzar = false;  
        this.isPausar = false;  
    }  
}  
  
public boolean isIsPausar() {  
    return this.isPausar;  
}  
  
public void pausarSimulacion() {  
    this.isPausar = true;  
  
    for (Filosofo filosofo : this.filosofos) {  
        filosofo.pausar();  
    }  
}  
  
public void reanudarSimulacion() {  
    this.isPausar = false;  
  
    for (Filosofo filo : this.filosofos) {  
        filo.reanudar();  
    }  
}
```

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

```
}
```

```
}
```

```
}
```

Clase ConstanteColorTiempo:

```
/*
```

```
* 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 ups.edu.ec.modelo;
```

```
import java.awt.Color;
```

```
/**
```

```
*
```

```
* @author User
```

```
*/
```

```
public class ConstantesColorTiempo {
```

```
    public ConstantesColorTiempo() {
```

```
    }
```


```
    public static final int DEMORA_PENSANDO_MIN = 1000;
```

```
    public static final int DEMORA_PENSANDO_MAX = 5000;
```

```
    public static final int DEMORA_COMIENDO_MIN = 1000;
```

```
    public static final int DEMORA_COMIENDO_MAX = 5000;
```

```
    public static final int TIEMPO_DORMIR_HILO = 5000;
```

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

```

public static final Color COLOR_PANEL = new Color(0, 51, 102);
public static final Color COLOR_PANEL2 = new Color(102, 102, 0);
public static final Color COLOR_MESA = new Color(153, 51, 255);
public static final Color COLOR_FILO_MESA = new Color(64, 55, 158);
public static final Color COLOR_TENEDOR = new Color(225, 225, 225);

public static final Color[] ESTADOS_COLOR = {
    new Color(0, 255, 0), //pensar
    new Color(255, 85, 0), //hambre
    new Color(0, 51, 255) // comer
};

}

```


Enum EnumEstado:

```

/*
 * 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 ups.edu.ec.modelo;

/**
 *
 * @author User
 */
public enum EnumEstado {

```

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

```

PIENSA(0, "Piensa"),
HAMBRIENTO(1, "Hambriento"),
COME(2, "Come");

//id del estado
private int id;

//texto del estado
private String texto;


EnumEstado(int id, String texto) {
    this.id = id;
    this.texto = texto;
}

public int getId() {
    return id;
}

//para leer un estado a partir del id
public static EnumEstado getEstado(int id) {
    for (EnumEstado estado : EnumEstado.values()) {
        if (estado.getId() == id) {
            return estado;
        }
    }

    return null;
}

```

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

@Override

```
public String toString() {
    return this.texto;
}
```

```
}
```

Enum EnumTenedor:

```
/*
 * 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 ups.edu.ec.modelo;
```


```
/**
 *
 * @author User
 */
```

```
public enum EstadoTenedor {

    DESOCUPADO,
    USANDO_IZQUIERDA,
    USANDO_DERECHA
}
```

Clase Filosofo:

```
/*
 * To change this license header, choose License Headers in Project Properties.
```

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

```

* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package ups.edu.ec.modelo;

import java.util.concurrent.Semaphore;
import ups.edu.ec.controlador.Controlador;

/**
 *
 * @author User
 */
public class Filosofo extends Semaphore implements Runnable {


    private Controlador controlador;
    private Thread hilo;
    private EnumEstado estado;
    private int id;

    public Filosofo(Controlador controlador, int id) {
        super(1, true);

        this.controlador = controlador;
        this.estado = EnumEstado.PIENSA;
        this.id = id;

        this.hilo = new Thread(this);
    }

```

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

```

public int getId() {
    return this.id;
}

public EnumEstado getEstado() {
    return this.estado;
}

public Controlador getControlador() {
    return controlador;
}

public void setControlador(Controlador controlador) {
    this.controlador = controlador;
}

public Thread getHilo() {
    return hilo;
}

public void setHilo(Thread hilo) {
    this.hilo = hilo;
}

@Override
public int hashCode() {
    int hash = 7;
    hash = 97 * hash + this.id;
    return hash;
}

```




```
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Filosofo other = (Filosofo) obj;
    if (this.id != other.id) {
        return false;
    }
    return true;
}

public void comenzar() {
    this.hilo.start();
}

public void parar() {
    this.hilo.stop();
}

public void pausar() {
```

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

```

this.hilo.suspend();
}

public void reanudar() {
    this.hilo.resume();
}

private void saltarseComida() {
    this.controlador.getPrincipal().saltarseComidaInterfaz(this);
}

public void pensar() throws InterruptedException {
    if (this.estado != EnumEstado.PIENSA) {
        return;
    }


    this.saltarseComida();

    Thread.sleep(ConstantesColorTiempo.DEMORA_PENSANDO_MIN +
Controlador.RANDOMICO.nextInt(ConstantesColorTiempo.DEMORA_PENSANDO_MAX -
ConstantesColorTiempo.DEMORA_PENSANDO_MIN));
}

public void comer() throws InterruptedException {
    if (this.estado != EnumEstado.COME) {
        return;
    }

    this.saltarseComida();

```

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

```

Thread.sleep(ConstantsColorTiempo.DEMORA_COMIENDO_MIN +
Controlador.RANDOMICO.nextInt(ConstantsColorTiempo.DEMORA_COMIENDO_MAX -
ConstantsColorTiempo.DEMORA_COMIENDO_MIN));

}

public void preguntarPuedeComer() {
    Filosofo[] vecinos = this.controlador.getVecinos(this);

    if (this.estado == EnumEstado.HAMBRIENTO && vecinos[0].estado != EnumEstado.COME &&
vecinos[1].estado != EnumEstado.COME) {
        this.estado = EnumEstado.COME;

        this.controlador.setTenedor(this.id, false);
        this.controlador.setTenedor((this.id + 1) % this.controlador.getNumeroFilosofos(), false);

        this.saltarseComida();


        this.release();
    }
}

public void tomarTenedores() throws InterruptedException {
    synchronized (this) {
        this.estado = EnumEstado.HAMBRIENTO;
        this.saltarseComida();

        this.preguntarPuedeComer();
    }

    this.acquire();
}

```

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

```

}

public synchronized void dejarTenedores() {
    this.estado = EnumEstado.PIENSA;

    this.controlador.setTenedor(this.id, true);
    this.controlador.setTenedor((this.id + 1) % this.controlador.getNumeroFilosofos(), true);

    this.saltarseComida();


    for (Filosofo filoVecino : this.controlador.getVecinos(this)) {
        filoVecino.preguntarPuedeComer();
    }
}

@Override
public void run() {
    while (!this.controlador.isIsPausar()) {
        try {

            this.pensar();
            this.tomarTenedores();
            this.comer();
            this.dejarTenedores();

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

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

```

}

@Override
public String toString() {
    return "Filosofo{" + "controlador=" + controlador + ", hilo=" + hilo
        + ", estado=" + estado + ", id=" + id + '}';
}

}

```


Clase GraficarComedor:

```

/*
 * 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 ups.edu.ec.vista;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.AffineTransform;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;
import javax.swing.JPanel;

```

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

```

import ups.edu.ec.modelo.ConstantesColorTiempo;
import ups.edu.ec.modelo.EnumEstado;
import ups.edu.ec.modelo.EstadoTenedor;


/**
 *
 * @author User
 */
public class GraficarComedor extends JPanel implements Runnable {

    private final Ellipse2D.Double formaMesa = new Ellipse2D.Double(-2.5, -2.5, 5, 5);
    private Thread hilo;
    private int numeroFilosofos;
    private EnumEstado[] estados;
    private Dimension dimensionComedor;
    //este objeto nos sirve para poner la tabla al medio de la ventana
    private AffineTransform affineTransform;

    public GraficarComedor(int n) {
        this.numeroFilosofos = n;
        this.setBackground(ConstantesColorTiempo.COLOR_PANEL);
        this.estados = new EnumEstado[n];
        dimensionComedor = new Dimension(0, 0);
    }

    public EnumEstado[] getEstados() {
        return this.estados;
    }
}

```

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

```

public void disenioFilosofos(int i, Graphics2D grafico2d) {
    // color
    if (this.estados[i] != null) {
        grafico2d.setPaint(ConstantesColorTiempo.ESTADOS_COLOR[this.estados[i].getId()]);
    } else {
        grafico2d.setPaint(getBackground());
    }

    // tamaño
    double size = Math.min(6 / Math.pow(numeroFilosofos, .8), 1.5);
    Ellipse2D.Double formaEliptica = new Ellipse2D.Double(3, -size / 2, size, size);

    // color base
    if (hilo == null || !hilo.isAlive()) {
        grafico2d.setColor(new Color(51, 102, 0));
    }

    // dibujar filosofos
    grafico2d.fill(formaEliptica);
    grafico2d.setPaint(Color.black);

    /*String text = i + " ";

    grafico2d.drawString(text, 0, 0);*/

    grafico2d.draw(formaEliptica);
    //para que el grafico vaya de lado
    grafico2d.rotate(2 * Math.PI / numeroFilosofos);

```

```
}

public EstadoTenedor getEstadoTenedor(int id) {
    if (this.estados[(id - 1 + numeroFilosofos) % numeroFilosofos] == EnumEstado.COME) {
        return EstadoTenedor.USANDO_IZQUIERDA;
    } else if (this.estados[id] == EnumEstado.COME) {
        return EstadoTenedor.USANDO_DERECHA;
    } else {
        return EstadoTenedor.DESOCUPADO;
    }
}


public void disenioTenedor(int i, Graphics2D grafico2d) {
    // determinar posicion
    double posicion = (2 * i - 1) * Math.PI / numeroFilosofos;

    EstadoTenedor estado = this.getEstadoTenedor(i);
    float redondoFilo = 0;

    if (estado == EstadoTenedor.USANDO_IZQUIERDA) {
        redondoFilo = -.8f;
    } else if (estado == EstadoTenedor.USANDO_DERECHA) {
        redondoFilo = .8f;
    }

    posicion = posicion + redondoFilo * (Math.PI / (2 * numeroFilosofos));

    //longitud de tenedores
    double longitudTenedor = (2.0 * 3) / numeroFilosofos;
```


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

```

// grafico tenedores
grafico2d.setPaint(ConstantesColorTiempo.COLOR_TENEDOR);

//para que los tenedores vayan de lado
grafico2d.rotate(posicion);

//afinar la grafica
grafico2d.setStroke(new BasicStroke((float) (.3 / Math.pow(numeroFilosofos, .55))));

if (estado == EstadoTenedor.USANDO_IZQUIERDA || estado == EstadoTenedor.USANDO_DERECHA) {
    //mover los tenedores al lado del filosofo
    grafico2d.draw(new Line2D.Double(3, 0, 3 + longitudTenedor, 0));
} else {
    //poner tenedores en la mesa
    grafico2d.draw(new Line2D.Double(2.3 - longitudTenedor, 0, 2.3, 0));
}


grafico2d.rotate(-posicion);
}

public void saltarseComidaGraficoMesa() {
    // formar la dimension de la tabla
    if (!dimensionComedor.equals(this.getSize())) {
        this.dimensionComedor = this.getSize();

        // el tamaño de la tabla
        double escalar = Math.min(dimensionComedor.width, dimensionComedor.height) / 10.0;

        // poner en el medio de la tabla
        this.affineTransform = new AffineTransform();

```

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

```

this.affineTransform.translate(dimensionComedor.width / 2.0, dimensionComedor.height / 2.0);
this.affineTransform.scale(escalar, escalar);

}

}

public void comenzar() {
    this.hilo = new Thread(this);
    this.hilo.start();
}


public void parar() {
    if (this.hilo != null) {
        this.hilo.interrupt();
    }
}

public void pausar() {
    this.hilo.suspend();
}

public void reanudar() {
    this.hilo.resume();
}

@Override
public void run() {
    while (true) {
        try {
            Thread.sleep(ConstantesColorTiempo.TIEMPO_DORMIR_HILO);

```

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

```

    } catch (InterruptedException e) {
        break;
    }

    repaint();
}

for (int i = 0; i < numeroFilosofos; i++) {
    this.estados[i] = null;
}


repaint();
}

@Override
public void paint(Graphics g) {
    super.paint(g);
    Graphics2D grafico2d = (Graphics2D) g;
    AffineTransform afinarMesa = grafico2d.getTransform();

    // diseñar formas
    RenderingHints lineasFilo = new RenderingHints(
        RenderingHints.KEY_TEXT_ANTIALIASING,
        RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
    lineasFilo.add(new RenderingHints(RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON));
    grafico2d.setRenderingHints(lineasFilo);

    this.saltarseComidaGraficoMesa();

```

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

```

// mesa en el medio
grafico2d.transform(affineTransform);

// dibujo mesa
grafico2d.setStroke(new BasicStroke(.15f));
grafico2d.setPaint(ConstantesColorTiempo.COLOR_MESA);
grafico2d.fill(this.formaMesa);
grafico2d.setPaint(ConstantesColorTiempo.COLOR_FILO_MESA);
grafico2d.draw(this.formaMesa);

// diseño filosofos
grafico2d.setStroke(new BasicStroke(.04f));


for (int i = 0; i < numeroFilosofos; i++) {
    this.disenioFilosofos(i, grafico2d);
}

// diseño tenedores
for (int i = 0; i < numeroFilosofos; i++) {
    this.disenioTenedor(i, grafico2d);
}

grafico2d.setTransform(afinarMesa);
}

@Override
public Dimension getPreferredSize() {
    return new Dimension(600, 600);
}

```

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

```
}
```

```
}
```

Interfaz InterfacePrincipal:

```
/*
```

```
* 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 ups.edu.ec.vista;
```

```
import ups.edu.ec.modelo.Filosofo;
```

```
/**
```

```
*
```

```
* @author User
```

```
*/
```

```
public interface InterfacePrincipal {
```

```
    void comenzar(int numeroFilosofos);
```

```
    void saltarseComida(Filosofo filosofo);
```


```
}
```

Clase Principal:

```
/*
```

```
* To change this license header, choose License Headers in Project Properties.
```

```
* To change this template file, choose Tools | Templates
```

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

```

* and open the template in the editor.
*/
package ups.edu.ec.vista;

import ups.edu.ec.controlador.Controlador;
import ups.edu.ec.modelo.Filosofo;

/**
 *
 * @author User
 */
public class Principal {


    private InterfacePrincipal interfaz;
    private Controlador controlador;

    private Principal() {
        this.interfaz = new Ventana(this);
        this.controlador = new Controlador(this);
    }

    public void comenzarSimulacion() {
        this.controlador.comenzarSimulacion();
    }

    public void pararSimulacion() {
        this.controlador.pararSimulacion();
    }
}

```

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

```

public boolean isSimulacionPausar() {
    return this.controlador.isIsPausar();
}

public void pausarSimulacion() {
    this.controlador.pausarSimulacion();
}

public void reanudarSimulacion() {
    this.controlador.reanudarSimulacion();
}


public void saltarseComidaInterfaz(Filosofo filosofo) {
    this.interfaz.saltarseComida(filosofo);
}

public void inicializarFilosofos(int numeroFilosofos) {
    this.controlador.inicializador(numeroFilosofos);
    this.interfaz.comenzar(numeroFilosofos);
}

public static void main(String[] args) {
    int numeroFilosofos;

    try {
        numeroFilosofos = Integer.parseInt(args[0]);
    } catch (Exception e) {
        numeroFilosofos = 5;
    }
}

```

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

```

    new Principal().inicializarFilosofos(numeroFilosofos);
}
}

```


Clase Ventana:

```

/*
 * 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 ups.edu.ec.vista;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BorderFactory;
import javax.swing.Icon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import ups.edu.ec.modelo.ConstantesColorTiempo;

```


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

```

import ups.edu.ec.modelo.EnumEstado;
import ups.edu.ec.modelo.Filosofo;

/**
 *
 * @author User
 */
public class Ventana extends JFrame implements InterfacePrincipal, ActionListener {

    private JButton comenzar, parar, pausar, numeroFilosofos;

    private GraficarComedor comedor;

    private Principal principal;


    public Ventana(Principal principal) {
        this.principal = principal;

        this.setTitle("Filosofos");
        this.setMinimumSize(new Dimension(800, 500));
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Inicializar botones
        this.comenzar = new JButton("Comenzar");
        this.parar = new JButton("Parar");
        this.pausar = new JButton("Pausar");
        this.numeroFilosofos = new JButton("Numero filosofos");

        this.parar.setEnabled(false);

```

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

```

this.pausar.setEnabled(false);

/*

this.comenzar.setBackground(Color.BLUE);

this.parar.setBackground(Color.red);

this.pausar.setBackground(Color.ORANGE);

*/

this.comenzar.addActionListener(this);

this.parar.addActionListener(this);

this.pausar.addActionListener(this);

this.numeroFilosofos.addActionListener(this);

JPanel pieDePagina = new JPanel(new FlowLayout(FlowLayout.RIGHT));

//para poner los colores en el pie de pagina
for (int i = 0; i < ConstantesColorTiempo.ESTADOS_COLOR.length; i++) {
    this.hacerPieDePagina(pieDePagina, EnumEstado.getEstado(i).toString(),
ConstantesColorTiempo.ESTADOS_COLOR[i]);
}

pieDePagina.setOpaque(false);

// botones superiores
Container containerSuperior = new Container();

containerSuperior.setLayout(new FlowLayout());


containerSuperior.add(this.comenzar);

containerSuperior.add(this.parar);

containerSuperior.add(this.pausar);

containerSuperior.add(this.numeroFilosofos);

```

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

```

Container containerInferior = new Container();
containerInferior.setLayout(new FlowLayout());
containerInferior.add(pieDePagina);

this.add(containerSuperior, BorderLayout.NORTH);
this.add(containerInferior, BorderLayout.SOUTH);

this.getContentPane().setBackground(ConstantsColorTiempo.COLOR_PANEL2);

this.setExtendedState(Ventana.MAXIMIZED_BOTH);
}

@Override
public void comenzar(int numero) {
    //para borrar el comedor
    if (this.comedor != null) {
        this.comedor.parar();
        this.remove(this.comedor);
    }

    //nuevo comedor
    this.comedor = new GraficarComedor(numero);
    this.add(this.comedor);

    this.pack();
    this.setVisible(true);
}

```

@Override

```
public void saltarseComida(Filosofo filo) {  
    // se cambio el estado del filosofo  
    this.comedor.getEstados()[filo.getId()] = filo.getEstado();  
  
    repaint();  
}
```

/**

* acciones de los botones


* @param evento

*/

@Override

```
public void actionPerformed(ActionEvent evento) {  
    //boton comenzar  
    if (evento.getSource() == this.comenzar) {  
        this.comenzar.setEnabled(false);  
        this.parar.setEnabled(true);  
        this.pausar.setEnabled(true);  
  
        this.comedor.comenzar();  
        this.principal.comenzarSimulacion();  
    }  
    //boton pausar  
    } else if (evento.getSource() == this.pausar) {  
        boolean pausado = this.principal.isSimulacionPausar();  
  
        if (pausado) {  
            this.principal.reanudarSimulacion();  
            this.comedor.reanudar();  
        }  
    }  
}
```

```
this.pausar.setText("Pausar");  
} else {  
    this.principal.pausarSimulacion();  
    this.comedor.pausar();  
  
    this.pausar.setText("Reanudar");  
}  
  
//boton numero filosofos  
} else if (evento.getSource() == this.numeroFilosofos) {  
    String mensaje = "Escribir el número de filosofos:";  
    String numeroJOption = JOptionPane.showInputDialog(this, mensaje);  
    int numeroFilosofos;  
  
    try {  
        numeroFilosofos = Integer.parseInt(numeroJOption);  
    } catch (Exception ex) {  
        return;  
    }  
  
    //para poner un maximo y minimo de numero de filosofos  
    numeroFilosofos = Math.max(Math.min(numeroFilosofos, 30), 3);  
  
    // reiniciar los botones  
    this.comenzar.setEnabled(true);  
    this.parar.setEnabled(false);  
    this.pausar.setEnabled(false);
```

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

```
// se reinicia el numero de filosofos
```

```
this.principal.pararSimulacion();
```

```
this.principal.inicializarFilosofos(numeroFilosofos);
```

```
//boton parar
```

```
} else if (evento.getSource() == this.parar) {
```

```
    this.comenzar.setEnabled(true);
```

```
    this.parar.setEnabled(false);
```

```
    this.pausar.setEnabled(false);
```

```
    this.comedor.parar();
```

```
    this.principal.pararSimulacion();
```

```
}
```

```
}
```

```
public void hacerPieDePagina(JPanel panelPieDePagina, String texto, Color color) {
```

```
    JPanel panel = new JPanel();
```

```
    JLabel label = new JLabel(texto + " ");
```

```
    panel.setBackground(color);
```


```
    label.setForeground(color);
```

```
    panelPieDePagina.add(panel);
```

```
    panelPieDePagina.add(label);
```

```
}
```

```
}
```

	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):

- Afianzar los conocimientos sobre la clase Thread, y la interfaz Runnable.
- Investigar sobre la Graphics para poder dibujar a los filósofos y la mesa.
- Ampliar nuestros conocimientos mediante el uso de las distintas fuentes existentes en internet.

CONCLUSIONES:

En conclusión, esta practica me ha servido de mucho para aplicar lo aprendido en clase. Además, para completar esta tarea he tenido que leer muchos documentos sobre distintas cosas que necesitaba para completar la tarea. Cabe recalcar que para las graficas he aprendido muchas cosas que son útiles para este tipo de proyectos que requieren una mayor aplicabilidad de interfaz.

RECOMENDACIONES:

No existe ninguna recomendación.

Nombre de estudiante: Adolfo Sebastián Jara Gavilanes.

Firma de estudiante:

