**Universidad Politécnica de Tecámac**

**Programación Concurrente**

**Documentación Código 5 Filósofos**

**Nombre**: Hernández García Luis Fernando

**Profesor**: Ramírez Fuentes Benito

**Grupo**: 2722IS

**Matricula**: 1320114059

Contenido

[**Problemática** 3](#_Toc116423843)

[**Solución** 3](#_Toc116423844)

[**Visualización del Programa** 3](#_Toc116423845)

[**Código en formato txt** 4](#_Toc116423846)

[**Funciones del Código** 6](#_Toc116423847)

[**Ejecución del programa** 8](#_Toc116423848)

[**Conclusión** 10](#_Toc116423849)

[Ilustración 1 Uso de la clase de Filósofos 6](#_Toc116423879)

[Ilustración 2 Creación de Hilos 6](#_Toc116423880)

[Ilustración 3 Creación de la clase Filosofo 7](#_Toc116423881)

[Ilustración 4 Creación de método Comer 7](#_Toc116423882)

[Ilustración 5 Creación de Monitores para alternar palillos 8](#_Toc116423883)

[Ilustración 6 Capturas de ejecución del programa. 9](#_Toc116423884)

# **Problemática**

En una mesa se encuentran 5 Filósofos, donde los 5 quieren comer sus platillos que comida que están situados en la mesa, sin embargo, solamente cada uno tiene un palillo para comer y cada uno necesita de 2 palillos para poder comer bien.

La forma en la que planearon para comer todos es que compartan sus palillos, pero con el que tienen a un lado, el que no tenga palillos, debería esperar a que uno termine y deje sus palillos para que pueda comer.

# **Solución**

Para que los 5 filósofos puedan comer de la mejor manera posible, se alterno por el desarrollo de un programa (de cierta forma aleatorio) donde se estará mostrando el orden en el que los filósofos estarán tomando los palillos y comerán, seguido de eso, los dejaran para que otro entre en acción.

Todo esto con la ayuda de la programación concurrente, donde con la ayuda de Hilos, este proceso se podrá desarrollar de manera rápida. Además de ello, se implementará el uso de Monitores, los cuales ayudaran a bloquear variables para ningún otro haga uso de ellos. En este caso los palillos, si estos están en uso, ningún otro filósofo hará uso de los palillos hasta que este los suelte, es ahí donde dejara de estar protegida la variable.

# **Visualización del Programa**

Por esta ocasión, el programa esta presentado en modo de Consola, donde se podrá visualizar el orden en el que el filosofo toma sus palillos, come y los deja, para después otro filosofo hacer uso de ellos.

# **Código en formato txt**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace ExamenFilosofos

{

class Program

{

static void Main(string[] args)

{

//Asignacion de cantidad de Filosofos

const int numFilosof = 5;

//Uso de la clase Filosofo como variable

Filosofo[] filosof = new Filosofo[5];

filosof[0] = new Filosofo(0, 0, numFilosof - 1, 0);

//Arreglo para recorrer el orden de los filosofos con los palillos

for (int i = 1; i < numFilosof; i++){

filosof[i] = new Filosofo(i, i - 1, i, 0);

}

//Hilos

var H1 = new Thread(new ParameterizedThreadStart(filosof[0].comer));

var H2 = new Thread(new ParameterizedThreadStart(filosof[1].comer));

var H3 = new Thread(new ParameterizedThreadStart(filosof[2].comer));

var H4 = new Thread(new ParameterizedThreadStart(filosof[3].comer));

var H5 = new Thread(new ParameterizedThreadStart(filosof[4].comer));

H1.Start(filosof[0]);

H2.Start(filosof[1]);

H3.Start(filosof[2]);

H4.Start(filosof[3]);

H5.Start(filosof[4]);

Console.ReadKey();

}

class Filosofo{

//Declaracion de Variables

int index;

int tandpetareVan;

int tandpetareHog;

int cmelman;

//Llamado de variables para su uso

public Filosofo(int indice, int palilloIzq, int palilloDrc, int cmelmansueco)

{

this.index = indice;

this.tandpetareVan = palilloIzq;

this.tandpetareHog = palilloDrc;

this.cmelman = cmelmansueco;}

//Metodo para el proceso de cada Filosofo

public void comer(object param){

Object[] objetos = new Object[5];

//Metodo FOR para alternar el orden de los Filoso

for (int i = 0; i < objetos.Length; i++){

objetos[i] = new Object();

}

Filosofo filo = (Filosofo)param;

//Metodo FOR para hacer que cada Filosofo coma 5 veces

for (int i = 0; i < 5; i++){

//Metodo IF para asignar los palillos a cada filosofo, sea derecho o izquierdo

if (Monitor.TryEnter(objetos[filo.tandpetareVan], 1000)){

try{

Console.WriteLine("El Filosofo {0} toma el palillo Izquierdo", index);

if (Monitor.TryEnter(objetos[filo.tandpetareHog], 1000))

{

try{

Console.WriteLine("El Filosofo {0} toma el palillo Derecho", index);

Console.WriteLine("El Filosofo {0} esta comiendo", index);

cmelman++;

Console.WriteLine("El Filosofo {0} ha comido " + cmelman + " veces", index);

}

finally

{

Monitor.Exit(objetos[filo.tandpetareHog]);

Console.WriteLine("El Filosofo {0} dejo el palillo Izquierdo", index);

}

}

else{

Console.WriteLine("El Filosofo {0} tiene hambre", index);

}

}

finally{

Monitor.Exit(objetos[filo.tandpetareVan]);

Console.WriteLine("El Filosofo {0} dejo el palillo Derecho", index);

}

}

else{

Console.WriteLine("El Filosofo {0} tiene hambre", index);

}

}

}

}

}

# **Funciones del Código**

Al inicio del código se hace el llamado de la clase de Filósofos, la cual se desarrollo mas abajo del código, este para que haga la función de la alternación de palillos entre ellos. Además de declarar la cantidad de Filósofos que habrá y darle un orden a los filósofos para que interactúen.

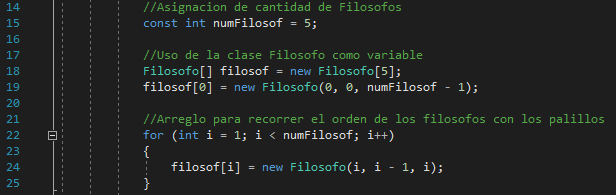


Ilustración Uso de la clase de Filósofos

La creación de Hilos es fundamental para un procedimiento más rápido y eficiente, dividiendo el proceso en varios hilos. En este caso, desarrollando un hilo para cada filosofo.

Texto

Descripción generada automáticamente

Ilustración Creación de Hilos

Al crear una clase para los Filósofos, en ella primera se declaran los palillos ya sean izquierdos y derechos, además de declarar un índice para que este actué como el orden entre cada filosofo.

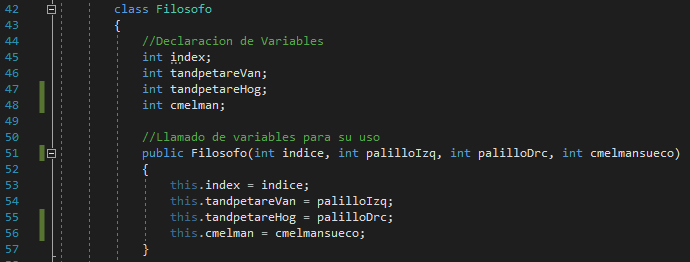


Ilustración Creación de la clase Filosofo

Crear el método de Comer, el cual estarán involucrados todo el proceso para el estar alternando los palillos con los que cada uno estará comiendo.

Texto

Descripción generada automáticamente

Ilustración Creación de método Comer

Para la alternación de palillos con ayuda de los Monitores, estos se encargan de bloquear las variables libres de cada palillo, con el cual cada filosofo deberá de esperar hasta que estos estén libres o disponibles para su uso, el método FOR que se implementa es para que cada Filósofos pueda comer 5 veces.

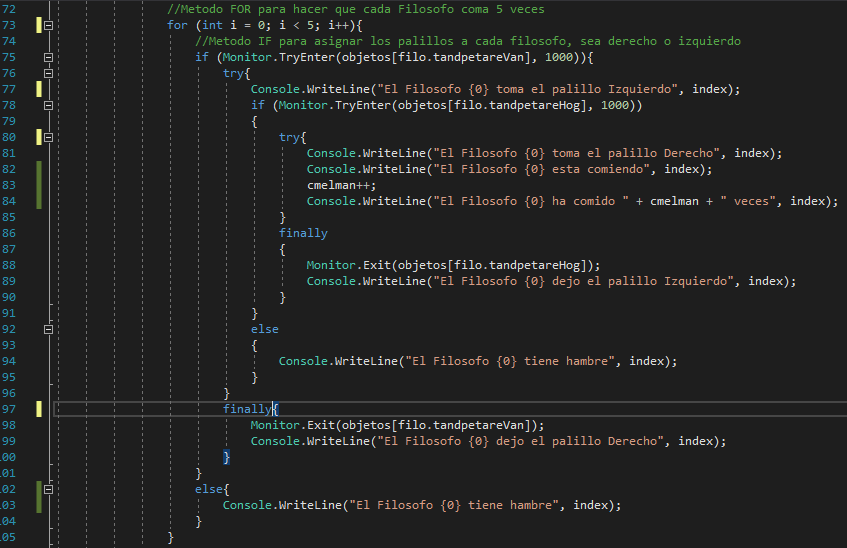


Ilustración Creación de Monitores para alternar palillos

# **Ejecución del programa**

Para comprobar que el programa presentado sirve de la manera mas optima, se presentan las siguientes capturas donde se presenta su funcionamiento.

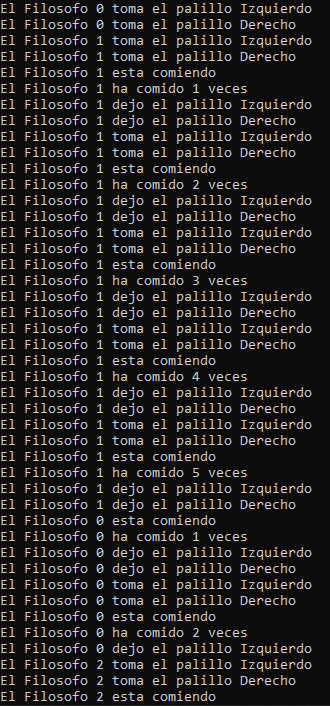
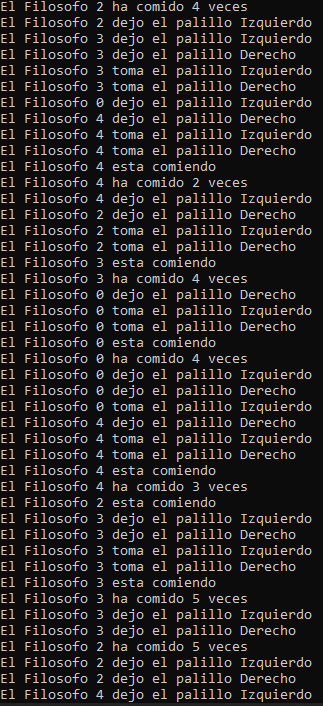
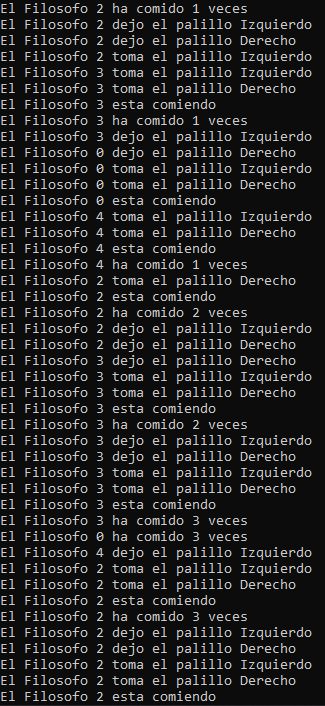
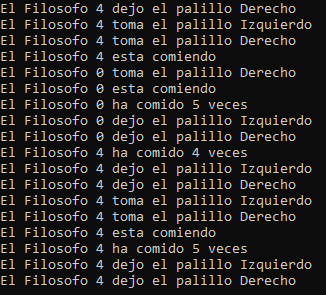


Ilustración Capturas de ejecución del programa.





# **Conclusión**

Como conclusión el hecho de que cada filosofo coma su 5 veces correspondientes fue todo un éxito, si bien el orden en el que comen es un poco disparejo, los 5 comen las 5 veces correspondientes. En lo que se puede visualizar, el código como tal necesita de lo que son los Monitores, sin estos el código simplemente no realizara nada, solamente va a abrir, pero no ocurrirá nada.

Ahora bien, sin el uso de Monitores, otras variantes que pueden ocurrir son el hecho de que ocurra corrupción de datos, un ejemplo es que 2 Filósofos utilicen el mismo palillo a la vez, en vez de que este bloqueado para los demás hasta que sea desocupado, sin embargo, ese no es el caso que se presenta.

En conclusión, el código funciona de forma mas optima y cada Filosofo como los correspondido a 5 platillos.