Imagen que contiene Logotipo

Descripción generada automáticamenteTexto

Descripción generada automáticamente con confianza media

Benemérita Universidad Autónoma de Puebla

“Pensar bien, para vivir mejor”

Facultad Ciencias de la Computación

Programación Concurrente y Paralela

Práctica 4

* Profesor: Cerón Garnica Carmen

• Equipo

Rojas Luna Alexa Michelle 202074633

Ponce Lara José Abel 202051988

Urbano Nieva Raúl 202059928

Antonio Ambrosio Jesús Noé 202062848

**Objetivo de la práctica**

Implementar soluciones con monitores Hoare y variables de condición.

**Introducción**

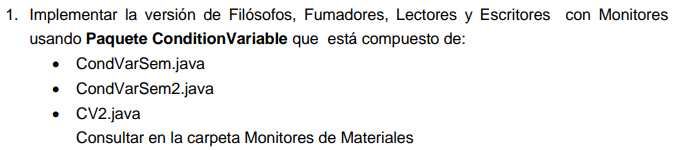
Como se ha visto, el semáforo es una primitiva de sincronización que no requiere espera ocupada y que no ha permitido solucionar con éxito los problemas habituales presentes en los sistemas con múltiples procesos en ejecución concurrente. Sin embargo, **es un mecanismo de bajo nivel**, ya que **no es estructurado**. Si deseáramos construir un gran sistema concurrente basado en semáforos, la responsabilidad de controlar los mismos seria algo difuso entre los programadores del sistema. Bastaría que alguien se olvidase de hacer una operación **signal** cuando fuera necesario, para provocar un **fallo total**, y **muy difícil de aislar** y **subastar**.

Los **monitores proporcionan una alternativa estructurada** como primitiva de programación concurrente, introducida por Hoare. No son más potentes ni útiles que los semáforos, pero si más **estructurados** y **legibles**.

Un monitor es una construcción sintáctica de un lenguaje de programación concurrente que encapsula una estructura de datos o recurso que se ha de proteger mediante exclusión mutua, junto con un conjunto de procedimientos que manejan ese recurso.

La interfaz del monitor con el exterior **es el conjunto de llamadas a procedimientos del monitor especificadas en su definición**. Los procesos requieren los servicios de los monitores mediante llamadas a sus procedimientos. Si un mismo monitor es llamado por dos procesos, la implementación del monitor asegura que las llamadas son procesadas secuencialmente para asegurar la exclusión mutua en el acceso del monitor. Si un mismo proceso llama a monitores diferentes, sus ejecuciones pueden ser entrelazadas y discurrir concurrentemente.

**Desarrollo de la práctica**

 Texto

Descripción generada automáticamente  Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja Texto

Descripción generada automáticamente

**Código**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package fumadores;

import java.util.Random;

/\*\*

\*

\* @author crish

\*/

public class Agente extends Thread{

int i,j;

Mesa mesa;

Random random;

public Agente(Mesa mesa){

this.mesa = mesa;

random = new Random();

}

public void run(){

while(true){

i=random.nextInt(3);

j=random.nextInt(3);

while(i==j){

j = random.nextInt();

}

mesa.poner(i,j);

}

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package fumadores;

//import java.util.Vector;

import java.util.ArrayList;

import java.util.List;

/\*\*

\*

\* @author crish

\*/

public class CV2 {

private List<Thread> bloqueados; // threads bloqueados en esta variable condición

private boolean condSalida; // condición para salir del DELAY

private String nombre; // nombre de la variable condición

private static List<String> condicion = new ArrayList<>();

public CV2(String \_nombre) {

nombre = \_nombre;

condSalida = true;

bloqueados = new ArrayList<>();

}

public void DELAY(Object monitor) {

synchronized (monitor) { // adquirimos el cerrojo sobre el monitor

try {

bloqueados.add(Thread.currentThread());

do {

monitor.wait();

// La condición de salida es que este thread sea el que más tiempo lleva

// esperando en la variable condición y que además esté esperando por una

// condición sobre la que se ha hecho un resume.

condSalida = bloqueados.get(0).equals(Thread.currentThread()) && condicion.contains(nombre);

// Si este thread no puede despertarse, se despierta a otro

if (!condSalida) {

monitor.notify();

}

} while (!condSalida);

condSalida = false;

bloqueados.remove(Thread.currentThread());

condicion.remove(nombre);

} catch (InterruptedException e) {

Thread.currentThread().interrupt(); // Restoring the interrupted status

e.printStackTrace();

}

}

}

public void RESUME(Object monitor) {

synchronized (monitor) {

if (!bloqueados.isEmpty()) {

if (!condicion.contains(nombre)) {

condicion.add(nombre);

}

monitor.notify(); // Solo despertamos a uno

}

}

}

public boolean EMPTY(Object monitor) {

synchronized (monitor) {

return bloqueados.isEmpty();

}

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package fumadores;

/\*\*

\*

\* @author crish

\*/

public class Fumador extends Thread{

int i;

Mesa mesa;

public Fumador(int i, Mesa mesa){

this.i=i;

this.mesa=mesa;

}

public void run(){

while(true){

mesa.coger(i);

System.out.println("F U M A N D O .. --");

System.out.println("El fumador "+i+" coloca el otro ingrediente para formar el cigarrillo");

try{

Thread.sleep(100);

}catch(Exception e){}

}

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template

\*/

package fumadores;

/\*\*

\*

\* @author crish

\*/

public class Fumadores {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

Mesa mesa = new Mesa();

for(int i=0; i<=3; i++){

//Mesa messa = new Mesa();

Agente agent = new Agente(mesa);

Fumador fumad = new Fumador(i,mesa);

//Thread agent = new Thread(agent);

//Thread fumad = new Thread(fumad);

agent.start();

fumad.start();

}

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package fumadores;

/\*\*

\*

\* @author crish

\*/

public class Mesa {

boolean ingredientes = false;

int ingrediente1=0, ingrediente2 =0;

CV2 nuevos[] = new CV2[50];

CV2 Tablavacia = new CV2("Mesa Vacia");

public Mesa(){

for(int i=0; i<3; i++){

nuevos[i] = new CV2("nuevos"+i);

}

for(int j=0; j<3; j++){

nuevos[j] = new CV2("nuevos"+j);

}

}

public synchronized void poner(int i ,int j){

if(ingredientes){

System.out.println("El agente se bloquea ... ");

Tablavacia.DELAY(this);

}

ingrediente1 = i;

ingrediente2 = j;

ingredientes = true;

System.out.println ("El agente ha dejado los ingredientes "+i+" y "+j);

int indice = (i+j) % nuevos.length; //hacemos que el indice contega los elementos totalees del ciclo

nuevos[indice].RESUME(this);

}

public synchronized void coger(int i){

if(!ingredientes || i == ingrediente1 || i == ingrediente2){

System.out.println("Se bloquea fumador.."+i);

nuevos[i].DELAY(this);

}

ingredientes = false;

Tablavacia.RESUME(this);

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package com.mycompany.golf;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

import java.util.concurrent.locks.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Club {

private Lock lock = new ReentrantLock();

private Condition pelotaLibre = lock.newCondition();

private Condition paloDispo = lock.newCondition();

private boolean pelotaDisponible = true;

private boolean paloDisponible = true;

private int[] pelotas;

private int[] palos;

private int sumtot;

private int sumtot2;

private int contador;

public Club() {

pelotas = new int[20];

palos = new int[20];

this.contador = 0;

}//le pongo tanto amor que no me queda!

public void prestarPelota(int cantidad,String nombre) {

lock.lock();

//lock.unlock();

try {

while (!pelotaDisponible) {

pelotaLibre.await();

}

System.out.println("Prestamo de pelotas...");

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

pelotas[i] = cantidad;

sumtot = pelotas[i];

contador++;

}

if(cantidad <2 || cantidad <5){ //solo pide 1 pelota el experto //el novato puede pedir 2 pelotas pero no mas de 5

while(nombre.equals("Experto")){

System.out.println("Se han prestado: " + sumtot+ " pelotas a un jugador "+ nombre);

}

}else {

if(cantidad > 2 || cantidad <= 5 ){

while(nombre.equals("Novato")){

System.out.println("Se han prestado "+ sumtot + " pelotas a un jugador "+ nombre);

}

}

}

pelotaDisponible = true;

} catch (InterruptedException ex) {

Logger.getLogger(Club.class.getName()).log(Level.SEVERE, null, ex);

} finally {

lock.unlock();

}

}

public void prestarPalo(int cantidad,String nombre){

lock.lock();

try {

while (!paloDisponible) {

paloDispo.await();

}

System.out.println("Prestamo de palos...");

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

palos[i] = cantidad;//alamacenamos el valor de 1 en i

//el contador va contar las veces necesarias hasta terminar cierto bucle de las clases anteriores

//i++;

sumtot2 = palos[i];

contador++;

}

if(cantidad <= 2){ //solo los expertos piden 2 a 5 palos

while(nombre.equals("Experto")){//mientras la variable contenga un nombre se va ejecutar

System.out.println("Se han prestado: "+ sumtot2 +" palos a un jugador"+ nombre);

}

//System.out.println("Se han prestado: "+ cantidad+" de palos a un jugador"+ nombre);

}else{

if(cantidad < 2 || cantidad <=5){

while(nombre.equals("Novato")){

System.out.println("Se han prestado " + sumtot2+ "palos a un jugador" );

}

}

}

paloDisponible = true;

} catch (InterruptedException ex) {

Logger.getLogger(Club.class.getName()).log(Level.SEVERE, null, ex);

} finally {

lock.unlock();

}

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package com.mycompany.golf;

/\*\*

\*

\* @author crish

\*/

public class Club\_M {

private boolean pelotaDisp = true;

private boolean paloDisp = true;

private int sum;

private int sum2;

private int[] pelotas;

private int[] palos;

private int contador;

public Club\_M(){

pelotas = new int[20];

palos = new int[20];

this.contador =0;//inicializacion del contador

}

public synchronized void PelotaprestarPelota(int cantidad,String nombre){

while(pelotaDisp== false ){

try{

wait();

}catch(Exception e){}

}

System.out.println("Prestamo de pelotas de golf..." );

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

pelotas[i]= cantidad;

//i++;

sum = pelotas[i];

contador++;

}

if(cantidad > 2 || cantidad > 5){

//while(nombre.equals("Experto")){

System.out.println("Se han prestado: "+sum+ "de 20 pelotas a un jugador "+ nombre);

//}

}else{

if(cantidad < 2 || cantidad <= 5){

// while(nombre.equals("Novato"))

System.out.println("Se han prestado "+sum+ " pelotas a un jugador " + nombre);

}

}

pelotaDisp = true;

}

public synchronized void PaloprestarPalo(int cantidad2,String nombre){

while(paloDisp ==false){

try{

wait();

}catch(Exception e){}

}

System.out.println("Prestamos de palos de golf ....");

for(int l=0; l<cantidad2; l++){

palos[l] = cantidad2;

//l++;

sum2 = palos[l];

contador++;

// while(nombre.equals("Experto"))

}

if(cantidad2 <= 2){ //solo los expertos piden 2 pelotas

System.out.println("Se han prestado: "+ sum2+" palos a un jugador"+ nombre);

}else{

//while( nombre.equals("Novato"))

if(cantidad2 < 2 || cantidad2 <=5){

System.out.println("Se han prestado " +sum2+ " palos a un jugador " + nombre);

}

}

paloDisp = true;

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\*/

package com.mycompany.golf;

import java.util.Scanner;

/\*\*

\*

\* @author crish

\*/

public class GOLF {

public static void main(String[] args) {

//Club clubs = new Club();

Club\_M club = new Club\_M();

Scanner leer = new Scanner(System.in);

int cant,cant2;

String socio;

System.out.println("¡..Bienvenido al Club de golf Estrella ..!");

System.out.println("Ingrese el tipo de usuario que es usted: ");

socio = leer.nextLine();

System.out.println("Ingrese la cantidad de pelotas necesarias: ");

cant= leer.nextInt();

System.out.println("Ingrese la cantidad de palos necesarios: ");

cant2=leer.nextInt();

JugadorNovato jugadorn = new JugadorNovato(club,socio,cant,cant2);

JugadorExperto jugadore = new JugadorExperto(club,socio,cant,cant2);

Thread jugn =new Thread(jugadorn);

Thread juge= new Thread(jugadore);

jugn.start();

juge.start();

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package com.mycompany.golf;

import java.util.Random;

/\*\*

\*

\* @author crish

\*/

public class JugadorExperto extends Thread {

private int cant1,cant2;

private String tipo\_jugador;

private Random random;

private int tiempojugando;

private int tiempoDescanso;

Club\_M club;

public JugadorExperto( Club\_M club,String Tipo,int cant1p,int cant2pal){

this.tipo\_jugador = Tipo;

this.random = new Random();

this.club =club;

this.cant1=cant1p;

this.cant2 =cant2pal;

}

public void run(){

for(int i=0; i<7; i++){

club.PelotaprestarPelota(cant1,tipo\_jugador);

club.PaloprestarPalo(cant2,tipo\_jugador);

}

tiempojugando = random.nextInt(6)+5;

System.out.println(tipo\_jugador+ "esta jugando durante"+ tiempojugando +"minutos");

tiempoDescanso = random.nextInt(6) + 5; // Tiempo de descanso entre 5 y 10 minutos

System.out.println(tipo\_jugador+ " está descansando durante " + tiempoDescanso + " minutos");

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package com.mycompany.golf;

import java.util.Random;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author crish

\*/

public class JugadorNovato extends Thread {

private int cant1,cant2;

private String tipo\_jugador;

private Random random;

private int tiempojugando;

private int tiempoDescanso;

Club\_M club;

public JugadorNovato( Club\_M club,String Tipo,int cant1p,int cant2pal){

this.tipo\_jugador = Tipo;

this.random = new Random();

this.club =club;

this.cant1=cant1p;

this.cant2 =cant2pal;

}

public void run(){

for(int i=0; i<7; i++){

club.PelotaprestarPelota(cant1,tipo\_jugador);

club.PaloprestarPalo(cant2, tipo\_jugador);

}

tiempojugando = random.nextInt(6)+5;

System.out.println("El "+ tipo\_jugador+"esta jugando durante"+ tiempojugando+ " minutos");

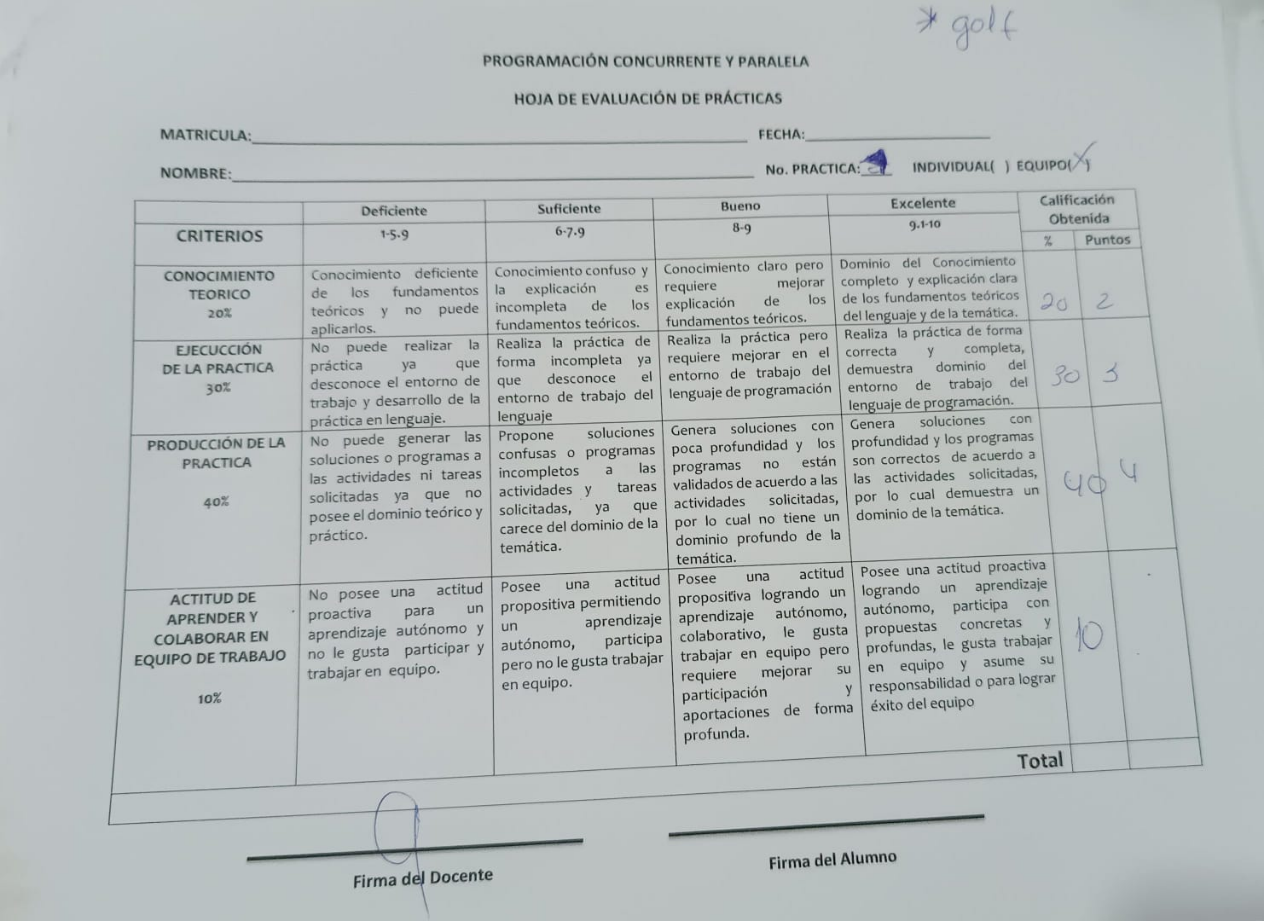
tiempoDescanso = random.nextInt(6) + 5; // Tiempo de descanso entre 5 y 10 minutos

System.out.println(tipo\_jugador+ " está descansando durante " + tiempoDescanso + " minutos");

}

}

**Hoja firmada**

****