\*\*\*COMPONENTES DEL GRUPO\*\*\*

Joaquín Gálvez Díaz (Portavoz)

Jorge Urbelz Alonso-Cortés

\*\*\*MONITOR MAQUINA\*\*\*

l: ReentrantLock privado

colaMaquina: Condition privado

maquinas[]: array de enteros privado

l <- ReentrantLock(true);

colaMaquina <- l.newCondition();

maquinas[] <- entero[3];

Constructor MonitorMaquina()

begin

for i in 0 … maquinas.length hacer

maquinas[i] <- -1;

fin\_for

end

privado indexOf(n: entero): entero

begin

index: entero

index <- -1;

for i in 0 … maquinas.length hacer

si maquinas[i] == n entonces

index <- i;

devolver index;

fin\_si

fin\_for

devolver index;

end

publico solicitarMaquina(id: entero, tiempoMaquina: entero, tiempoMesa: entero, monitorMesa: MonitorMesa): entero provoca InterruptedException

begin

l.bloquear();

mesaAsignada, numeroMesa, numeroMaquina, maquina: int

mesaAsignada <- -1;

intentar

maquina <- indexOf(id);

si maquina >= 0 hacer

mesaAsignada <- monitorMesa.mesaMenorTiempoEspera();

numeroMaquina <- maquina + 1;

imprimir("Cliente " + id + " ha solicitado su servicio en la máquina: " + numeroMaquina + "\n");

imprimir("Tiempo en solicitar el servicio: " + tiempoMaquina + "\n");

numeroMesa <- mesaAsignada + 1;

imprimir("Será atendido en la mesa: " + numeroMesa + "\n");

imprimir("Tiempo en la mesa: " + tiempoMesa);

numeroMesa <- 1;

imprimir("Tiempo de espera en la mesa" + numeroMesa + " = " + monitorMesa.getTiempoColaMesa()[0]);

numeroMesa <- numeroMesa + 1;

for i in 1 … hacer

monitorMesa.getTiempoColaMesa().length hacer

imprimir(", mesa" + numeroMesa + " = " + monitorMesa.getTiempoColaMesa()[i]);

numeroMesa <- numeroMesa + 1;

fin\_for

imprimir("\n\n"); monitorMesa.introducirClienteColaMesa(mesaAsignada, tiempoMesa);

maquinas[maquina] <- -1;

colaMaquina.signal();

fin\_si

sino

mientras(indexOf(-1) == -1) hacer

colaMaquina.await();

fin\_mientras

maquinas[indexOf(-1)] <- id;

fin\_sino

finalmente

l.desbloquear();

devolver mesaAsignada;

end

\*\*\*MONITOR MESA\*\*\*

l: ReentrantLock privado

colaMesa[]: array de Condition

tiempoColaMesa[], clienteColaMesa[]: array de enteros

l <- ReentrantLock(true);

colaMesa[] <- Condition[4];

tiempoColaMesa[] <- entero[4];

clienteColaMesa[] <- entero[4];

Constructor MonitorMesa()

begin

for i in 0 … colaMesa.length hacer

tiempoColaMesa[i] <- 0;

clienteColaMesa[i] <- -1;

colaMesa[i] <- l.newCondition();

fin\_for

end

publico getTiempoColaMesa(): entero[]

begin

l.bloquear();

intentar

devolver Arrays.copyOf(tiempoColaMesa, tiempoColaMesa.length);

finalmente

l.desbloquear;

end

public introducirClienteColaMesa(mesaAsignada, tiempoMesa: entero)

begin

l.bloquear();

intentar

tiempoColaMesa[mesaAsignada] <- tiempoColaMesa[mesaAsignada] + tiempoMesa;

finalmente

l.desbloquear();

end

publico mesaMenorTiempoEspera(): entero

begin

l.bloquear();

tiempoActual, mesaActual: entero;

tiempoActual <- tiempoColaMesa[0];

mesaActual <- 0;

intentar

for i in 1 … tiempoColaMesa.length hacer

si tiempoColaMesa[i] < tiempoActual entonces

tiempoActual <- tiempoColaMesa[i];

mesaActual <- i;

fin\_si

finalmente

l.desbloquear();

devolver mesaActual;

end

publico solicitarMesa(id, mesaAsignada, tiempoMesa: entero) provoca InterruptedException

begin

l.bloquear();

intentar

si clienteColaMesa[mesaAsignada] == id entonces

clienteColaMesa[mesaAsignada] <- -1;

tiempoColaMesa[mesaAsignada] <- tiempoColaMesa[mesaAsignada] – tiempoMesa;

colaMesa[mesaAsignada].signal();

fin\_si

sino

mientras clienteColaMesa[mesaAsignada] > -1 hacer

colaMesa[mesaAsignada].await();

fin\_mientras

clienteColaMesa[mesaAsignada] <- id;

fin\_sino

finalmente

l.desbloquear();

end

\*\*\*HILO CLIENTE\*\*\* EXTIENDE THREAD

id, tiempoMaquina, tiempoMesa: entero;

monitorMaquina: MonitorMaquina;

monitorMesa: MonitorMesa;

Constructor HiloCliente(id, tiempoMaquina, tiempoMesa: entero, monitorMaquina: MonitorMaquina, monitorMesa: MonitorMesa)

begin

id <- id;

tiempoMaquina <- tiempoMaquina;

tiempoMesa <- tiempoMesa;

monitorMaquina <- monitorMaquina;

monitorMesa <- monitorMesa;

end

publico run()

begin

mesaAsignada: entero;

intentar

monitorMaquina.solicitarMaquina(id, tiempoMaquina, tiempoMesa, monitorMesa);

Thread.dormir(tiempoMaquina);

mesaAsignada <- monitorMaquina.solicitarMaquina(id, tiempoMaquina, tiempoMesa, monitorMesa);

monitorMesa.solicitarMesa(id, mesaAsignada, tiempoMesa);

Thread.dormir(tiempoMesa);

monitorMesa.solicitarMesa(id, mesaAsignada, tiempoMesa);

capturar(e: InterruptedException)

e.imprimirTrazaPila();

end

\*\*\*MAIN\*\*\*

publico estático main(args[]: array de Strings)

begin

r: Random;

monitorMaquina: MonitorMaquina;

monitorMesa: MonitorMesa;

clientes[]: array de HiloCliente;

r <- Random();

monitorMaquina <- MonitorMaquina();

monitorMesa <- MonitorMesa();

clientes[] <- HiloCliente[50];

for i in 0 … clientes.length hacer

clientes[i] <- HiloCliente(i, r.nextInt(1000), r.nextInt(1000), monitorMaquina, monitorMesa);

clientes[i].start();

fin\_for

end