Práctica 3: Calabozos
Cómputo Concurrente 2024-2

Kassandra Mirael ♡

2024

Tiempo estimado de realizacion: ≈ 4hrs (+3hrs si hacen el extra)
Objetivo: Resolver un problema clásico de concurrencia (problema de los prisioneros), en 2 versiones, utilizando herramientas que Java ya nos proporciona.

Introducción

Ha trancurrido tiempo desde el ataque informatico sufrido, aunque se logro descifrar parte del msg, aún seguimos con la incógnita de quién realizo dicho ataque.

Debido a esto, la moral ha bajado y han empezado a haber combates clandestinos y debido a esto han mandado a varios al calabozo para que reflexionen sobre esto.

Un computólogo loco llamado Huey Emmerich, recomienda ponerles un ejercicio para que estos puedan salir antes del calabozo, de esta manera reforzar su pensamiento para futuros ataques...

Especificaciones

En esta práctica hay dos variantes del Problema de los Prisioneros a resolver, la primera es obligatoria y la segunda es optativa.
En ambos casos se debe crear un archivo llamado Constante.java, en el cual se incluyan constantes de configuración para la practica.
Para los mensajes de los prisioneros normales, se usara el color **AZUL** mientras para el que da el msg de que pasaron todos es **ROJO**.

Obligatorio

Descripción del problema: Existen N prisioneros y un guardia. El guardia les comenta a los prisioneros: *Pasaré a un prisionero a la vez a un cuarto D, en este cuarto hay un interruptor*

L (que tiene dos estados: ON/OFF). Cada prisionero visitará el cuarto D con frecuencia arbitraria. En cualquier momento un prisionero debe declarar "Ya hemos pasado todos". Si es correcto, todos los prisioneros son liberados, de lo contrario serán aventados a los cocodrilos.

Crea un algoritmo concurrente que ayude a los prisioneros a ser liberados, en este caso, supón que **no** conoces el estado inicial del interruptor L.

Toma en cuenta que:

- Solo tenemos 1 interruptor.
- No se conoce el estado inicial del interruptor.
- Pasan de manera pseudoaleatoria a los prisioneros.

Recuerda generar la simulación correspondiente en el main.

Optativo

En este caso, nos basaremos en el realizado en la tarea 2, omitiendo la parte donde se asegura que cada cierto número de prisioneros pasan, quedando de la siguiente manera:

- Tener 2 o 3 interruptores (Seleccionar solo una version, si hacen 2 no se tomará ninguna en cuenta)
- No pasan cada cierto tiempo, pasan de manera pseudoaleatoria
- Se conoce el estado inicial del interruptor

Para este caso deberán generar un nuevo proyecto en maven, en el cual se deberan incluir test equivalentes al original, así mismo en el main se debe incluir una ejecución para un número n de prisioneros.

Consideraciones

No se pueden utilizar cosas externas (paquetería) que no estén declaradas en la práctica (**NO Synchronized, NO Atomic, NO Semaphore, NO java.util.Concurrent**)

Cuestionario

Tu solución propuesta cumple:

- Exclusión Mutua
- No Deadlock
- Libre de Hambruna

De ser asi, demuestra cada propiedad.

Ademas contesta las siguientes preguntas justificando lo siguiente

- ¿Tu solución cumple para n prisioneros?
- Si tu programa tarda mucho en terminar, ¿por qué crees que pasa esto?
- De la pregunta anterior, que podrias proponer para mejorar los tiempos.

Estas preguntas debes contestarlas para cada version del problema (en caso de que realizes ambos).

Por ultimo, analiza bien el siguiente enunciado:
“Si los candados cumplen con exclusión mututa, no deadlock o libre de hambruna, es decir, con las propiedades para un candado seguro, entonces el sistema donde lo utilizemos tambien las cumplira.”
De esto justifica porque si se cumple o porque no.

Añade lo aprendido en esta practica, asi como las dificultades que tuviste para realizarla.

Por último...