UNC

PROGRAMACIÓN CONCURRENTE

Trabajo Práctico Nº 1

GRUPO:

Error 404

INTEGRANTES:

Chagay Vera Adriel Ian

Klincovitzky Sebastian

Severini Montanari Alejo

Vega Cuevas Silvia Jimena

Wortley Agustina Daniela

DOCENTES:

Dr. Ing. Micolini Orlando

Ing. Ventre Luis

Ing. Mauricio Ludemann

2020

FCEFyN

Enunciado

En un buffer cuya capacidad son 25 lugares, acceden múltiples consumidores y múltiples productores (es decisión del grupo la cantidad de consumidores y de productores). Este buffer tiene pura pérdida, es decir que si está lleno, el productor descarta los elementos. Cada consumidor puede extraer solo de a un elemento por vez y demora un tiempo aleatorio en milisegundos para consumirlo.

Cada productor puede añadir solo de a un elemento por vez y demora un tiempo aleatorio en milisegundos para generarlo.

El sistema posee un log que escribe cada 2 segundos, y tiene por objetivo realizar una estadística del uso del buffer y de los estados de los consumidores.

El log debe registrar:

- Cantidad de lugares ocupados del buffer.
- Estado de cada consumidor (ocupado consumiendo, o disponible).

Ejercicios

1) Hacer un diagrama de clases que modele el sistema del buffer con los productores y consumidores.

2) Hacer un solo diagrama de secuencias que muestre la siguiente interacción:

a) Con el buffer vacío, un productor coloca un producto satisfactoriamente

b) Luego, dos consumidores llegan al mismo tiempo a consumir. Mostrar qué pasa con cada consumidor.

3) Modelar el buffer, los productores y los consumidores, con objetos en Java. Se deben generar y consumir 1000 elementos en total.

4) Explicar los motivos de los resultados obtenidos.

5) Debe hacer una clase Main que al correrla, inicie el programa.

6) Los tiempos de generación y consumo de elementos en el buffer, deben configurarse para que el programa demore entre 60 segundos y 120 segundos en finalizar.

Desarrollo

Para empezar con nuestro proyecto, se realizó el diagrama de clases, y el diagrama de secuencias, para dejar bien definido los conceptos, funciones y relaciones de las diferentes partes del programa. Posteriormente se trabajó en el código del trabajo, primeramente asegurándonos de su correcto funcionamiento, para finalmente, trabajar con los detalles más finos en relación a cantidades y la resolución del ejercicio.

Código

Si bien no se mostrara el codigo aqui, se explicara que se busco que haga cada bloque del programa.

Main

Principalmente la función del main de nuestro programa, es inicializar las demás partes del programa, entre lo que se incluye la asignación de hilos a Productor y Consumidor.

Buffer

El buffer se podría decir que es el “inventario” donde los productores almacenan y de donde los consumidores consumen. Se puede encontrar una gran concurrencia en la instancia de esta clase, previniendo que solo un productor o consumidor tenga a la vez tenga acceso a esta, ya sea para colocar o retirar artículos.

También se debe observar que posee métodos que resultan útiles para gestionar el estado del buffer y poder llevar un registro del mismo.

Productor

El productor generará un Artículo nuevo durante un tiempo aleatorio de milisegundos, e intenta ponerlo en el buffer. En caso de poder hacerlo, sigue normalmente, caso contrario descarta el Artículo previamente.

Artículo

Realmente, en este ejercicio Artículo no tiene ningún peso, sin embargo aprovechamos para usarlo para mantener estadísticas sobre creación, consumo y descarte de los mismos.

Consumidor

Buscará consumir constantemente el artículo provisto por el productor, lo que le toma un tiempo aleatorio de milisegundos. Además proveerá su estado al log, para facilitarle su tarea.

Log

Se trata de un hilo que se encargará de acceder al estado del buffer, y de los consumidores, para luego guardarlo en un archivo txt y así tener registro de lo que sucede en el buffer a lo largo de la ejecución.

Decisiones de diseño

Como primer diseño, habíamos pensado hacer el Buffer de tal modo que cada uno de los 25 lugares del mismo posea un Lock propio, para que de ésta forma si un productor estaba dejando su artículo en la primer posición del buffer, otros productores y consumidores pudiesen estar interaccionando con el resto de posiciones. Pero luego de llevarlo a cabo nos dimos cuenta que hacerlo de esta forma nos daba muchos errores a la hora de compilar, intentamos solucionarlos pero no pudimos, así que decidimos simplificar la resolución (sabiendo que de esta forma se hacía menos eficiente el programa) permitiendo el ingreso de solo un hilo a todo el Buffer al mismo tiempo (cuando antes se lo permitíamos a 25 hilos, de los cuales cabe aclarar, no eran todos hilos físicos claramente).

En cuanto a la cantidad de productores y consumidores, dado que la computadora del Ingeniero Ventre (que es dónde se correrá el código) es de 8 núcleos, elegimos 5 productores y 3 consumidores, completando así los 8 núcleos físicos.

Elegimos 5 productores y 3 consumidores ya que con 4 productores y 4 consumidores no obteníamos pérdida de artículos, y con 6 productores y 2 consumidores la pérdida era mayor que lo que se consumía.

El programa concluye al consumirse 1000 artículos (sin contar los que se van descartando por los productores que no tienen espacio en el buffer para dejarlos).

Los tiempos que tardan los productores y consumidores en realizar sus tareas es un tiempo aleatorio entre 60 y 100 milisegundos. Elegimos estos valores ya que

poniendo como límite 1000 artículos consumidos son los que generan que el programa se ejecute entre 1 y 2 minutos.

Conclusiones

Llevando a cabo este trabajo pudimos darnos cuenta de la dificultad que tiene el manejo de programas concurrentes, especialmente en el debuggeo del mismo. Es difícil poder seguir cada hilo, a pesar de que manejamos solo 8 en este caso.

Como extra también podemos decir que nos sirvió el trabajo para aprender a organizarnos con el manejo de versiones, en este caso GitHub, ya que tuvimos algunos problemas con ciertos archivos y gracias a esto pudimos solucionarlo.

Bibliografía

Java 9 Concurrency Cookbook, Second Edition.

Material de la cátedra Programación Concurrente.