Programación Concurrente

Trabajo Práctico

Se desea implementar en Java un programa concurrente que a partir de una lista de números verifique cuáles son *números perfectos*. Un número es perfecto si es igual a la suma de sus divisores positivos. Por ejemplo 6 es un número perfecto ya que es igual a la suma de sus divisores $1+2+3$. El programa debe tomar como parámetro una lista de números grandes (utilizando la clase `BigInteger`) y un entero que indique la cantidad de threads a utilizar.

El programa debe respetar la siguiente estructura:

1. Una clase `Main` con el punto de entrada del programa (por simplicidad puede declarar la lista de números a verificar aquí, o si lo desea puede leerla de un archivo). El programa debe terminar únicamente cuando todos los números dados hayan sido evaluados, y finalmente informar el tiempo transcurrido desde el inicio de la ejecución.

2. Una clase `Buffer` (implementada como un monitor utilizando métodos synchronized) que actúa como una cola FIFO concurrente de capacidad acotada. Es decir, bloquea a un lector intentando sacar un elemento cuando está vacía y bloquea a un productor intentando agregar un elemento cuando está llena. La capacidad del Buffer también debe ser un parámetro configurable desde la clase `Main`.

3. Una clase `Barrier` (implementada como un monitor utilizando métodos synchronized) que actúa como una barrera: Cada thread que llega a ella se bloquea, hasta alcanzar una cantidad threads predefinida. En ese momento la barrera despierta a todos los threads y les permite continuar. El objetivo de la clase `Barrier` es garantizar que el thread principal no termine hasta que todos los threads auxiliares hayan terminado de verificar hasta el último número.

4. Una clase `PerfectWorker` que extiende de `Thread` y realiza la verificación de si un número es perfecto o no. Un `PerfectWorker` debe tomar los números a verificar de un `Buffer` conocido al momento de su creación. Si el número a verificar es negativo el `Thread` debe preparase para finalizar su ejecución sincronizándose con los otros threads utilizando la clase `Barrier`.

5. Una clase `ThreadPool`, que se encarga de instanciar e iniciar la cantidad de `PerfectWorkers` pedida por un usuario.

Al iniciar el programa se debe delegar la iniciación de los threads necesarios en la clase `ThreadPool` y luego introducir de a uno los números a verificar en el `Buffer`. Cada `PerfectWorker` en funcionamiento debe tomar números de a uno del `Buffer` y verificar si son o no perfectos. Cabe destacar que es inadmisible utilizar una cantidad de threads menor a la solicitada por el usuario salvo cuando se tienen menos números que threads.

Además del código debe entregar datos de test entre los que haya al menos 100 números distintos incluyendo al menos 8 números perfectos.