Universidad de los Andes.

Infraestructura Computacional.

Integrantes:

* Juan Diego Gonzalez 201531418
* Carlos Peñaloza 201531973
* Camilo Montenegro 201531747

Caso 1

**Diseño**

El programa está estructurado en 5 clases: Buffer, Cliente, Mensaje, Principal y Servidor en donde la función desempeñada por cada una es diciente mediante el nombre dado. A grandes rasgos el programa maneja una estructura Productor - Consumidor en donde las clases Servidor y Cliente son implementadas mediante threads. Por el otro lado, el cliente, mensaje y buffer son objetos necesarios para la ejecución del programa.

**Funcionamiento**

La clase Principal cumple la función de ejecutar el main en donde inicialmente se hace una lectura de los parámetros con los cuales debe funcionar el programa. Estos parámetros son leídos de un archivo .Properties y son cantidad de clientes, cantidad de servidores, mensajes y el tamaño del buffer. A partir de los datos tomados se inicia la creación e iniciación de los threads de servidor, lo anterior se da pasando el buffer por parámetro e invocando el método start(). Luego de tener todos los threads de servidor en ejecución se procede a la creación de los threads clientes donde se les asigna un ID, la cantidad de mensajes y el buffer. Debido a que el programa está basado en la arquitectura productor-consumidor, la interacción entre una instancia de cliente y del servidor se lleva a cabo utilizando el mismo objeto Buffer.

Para las clases de tipo thread se implementa principalmente el método Run(), en donde para el caso del servidor este inicia la ejecución intentando obtener el proximo mensaje del Buffer, si aún no hay mensajes en el Buffer el thread continúa en espera hasta que logra obtener el mensaje y dar un respuesta en el mismo objeto mensaje. En la clase Cliente el método Run() se encarga que de generar un mensaje con la información y el Buffer, este mensaje creado se intenta insertar al Buffer, si no es posible insertarlo, el cliente se duerme hasta que tenga cupo. Durante este procedimiento, el cliente debe ceder el procesador luego de cada intento, lo cual se realiza con el método Yield(). Luego de la correcta inserción del mensaje en el Buffer, el cliente queda en espera pasiva hasta recibir una respuesta por parte del servidor el cual notifica al cliente de continuar con el procedimiento. Finalmente, se hace uso de un monitor con el fin de asegurar que sólo una instancia de tipo thread se esté ejecutando en la zona crítica la cual es imprimir en pantalla el resultado de la consulta.

Es necesario recalcar el funcionamiento del objeto mensaje en la arquitectura del programa pues este objeto maneja un método insertar() el cual, además de ser synchronized, permite la inserción del mensaje en el buffer y se queda dormido esperando a recibir respuesta por parte del servidor. Luego de que ya se ha recibido una respuesta, el sistema se encarga de despertar (notify()) el proceso dormido y finalmente se retorna el mensaje respuesta.

A continuación se muestra un ejemplo de funcionamiento en donde hay 10 clientes, 2 servidores y cada cliente tiene 2;1;3;1;1;1;1;2;2;1 número de mensajes respectivamente. El tamaño del Buffer es de 2.

Se puede evidenciar que el servidor logra atender la cantidad total de mensajes por cliente.



