**Sistema de Gestión Empresarial y Operativa de una Compañía Transportadora**

**Caso 1**

**Manejo de la concurrencia**

**Carlos Felipe Agudelo Ospina**

**cf.agudelo12**

**201328150**

**Sergio Yodeb Velásquez Yepes**

**sy.velasquez10**

**201315851**

**Infraestructura Computacional**

**Universidad de Los Andes**

**Bogotá D.C.**

**2015-2**

**Organización del programa**

Se tiene un programa con cuatro clases:

* **Buffer:** Es la clase principal del programa. Desde esta se ejecuta el main y además se crean tanto los threads que modelan a los clientes como los threads que modelan los servidores. Esta clase tiene el buffer de mensajes, un atributo del tamaño del buffer y un contador de la cantidad de mensajes dentro del programa.
* **Servidor:** Extiende de la clase thread. Modela los servidores que han de leer los mensajes de los clientes y responder a estos. Cuenta con una referencia al buffer.
* **Cliente:** Extiende de la clase thread. Modela los clientes que enviarán mensajes al buffer. Cuenta con referencia al buffer, un id y la cantidad de mensajes que deben enviar.
* **Mensaje:** Representa un mensaje enviado por un cliente al buffer.

**Funcionamiento del programa**

El main del buffer se encarga de leer el tamaño que tendrá el buffer, la cantidad de servidores con los que contará el programa, la cantidad de clientes que realizarán consultas y la cantidad de consultas que harán estos a partir de un archivo properties. Se encarga además de crear la cantidad de threads de clientes, pasarles la información de la cantidad mensajes que deben enviar, una referencia al buffer y su id correspondiente, y crear los servidores correspondientes, pasarle una referencia al buffer y ejecutarlos a todos.

El run del cliente se encarga de crear iterativamente los mensajes del cliente, enviarlos al buffer y al final de cada iteración informar al buffer de que debe restar el número de mensajes a responder, porque ya fue respondido.

El run del servidor se encarga de hacer que el servidor se ejecute mientras que haya mensajes pendientes de atención y dentro de este se trata de leer un mensaje del buffer. Si el mensaje se pudo leer se responde y se pasa a la siguiente iteración, si no pudo leerse se pasa a la siguiente iteración. Al final de cada iteración el servidor se encarga de ceder su procesador para que otros servidores puedan ejecutar haciendo uso del método yield().

El envío de mensajes por parte de los clientes y la lectura de los mensajes por parte de los servidores se realizan a través del buffer. El envío de mensajes funciona bajo la siguiente algorítmica: El cliente envía el mensaje al servidor; si el buffer está lleno el cliente se duerme sobre el buffer, si no se agrega el mensaje al buffer. La lectura de mensajes funciona de la siguiente forma: Si no hay mensajes dentro del buffer se retorna null y el servidor vuelve a intentar leer después de ceder su procesador; si hay mensajes se retira el primero, se notifica a la cola del buffer para despertar a un posible cliente dormido esperando por espacio en el buffer y se retorna el mensaje para que el servidor pueda responderlo.

El método de entrega del mensaje en el buffer no es un método sincronizado, pero se compone de dos bloques sincronizados, el primer bloque esta sincronizado con respecto al buffer y se encarga de revisar si hay espacio para el mensaje, si no lo hay el thread del cliente se duerme sobre el buffer. Si hay espacio, o en el momento en el que se despierte el cliente que estaba dormido, se pasa al otro bloque sincronizado sobre el mensaje; allí dentro de otro bloque sincronizado sobre el buffer se agrega a la lista de mensajes del buffer el mensaje actual y se acaba ese bloque, luego se duerme el thread del cliente sobre el mensaje, porque se está en el bloque sincronizado con el mensaje.

El método de recibir el mensaje es un método sincronizado en el buffer que funciona de la manera que ya se explicó.