Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional

CASO I.
Manejo de la concurrencia

El sistema financiero en línea *(*Novasoft) recibe solicitudes de los diferentes usuarios. En este caso queremos implementar el esquema de concurrencia para manejar las consultas que se hacen sobre él.

En principio, las consultas consisten en operaciones financieras que el servidor debe realizar, sin embargo, en este caso, no tendremos en cuenta el contenido de esos mensajes, puesto que deseamos concentrarnos en el esquema de concurrencia. Para probar el programa, las consultas pueden consistir en números generados en secuencia, o al azar, y la respuesta en este número incrementado.

Objetivo

Diseñar un mecanismo de comunicación para manejar las consultas de múltiples clientes sobre el servidor Novasoft. Para este caso, los clientes y el servidor serán *threads* en la misma máquina (en realidad debería ser un sistema distribuido; este es solo un prototipo).

El proyecto debe ser realizado en java, usando *threads*. Para la sincronización solo pueden usar directamente las funcionalidades básicas de Java: `synchronized`, `wait`, `notify` y `notifyAll`.

Funcionamiento

Cada *thread* cliente hace un cierto número de consultas y termina. El número de clientes y el número de mensajes que envía cada uno deben ser un número arbitrario (cada cliente tendrá asignado un número particular de mensajes enviados). Para cada mensaje, el *thread* cliente debe generar un objeto de tipo Mensaje e inicializarlo, después lo envía. Cuando termine, el cliente le debe avisar al buffer que se retira.

El servidor, por su lado, estará compuesto por varios *threads* para poder atender múltiples consultas simultáneamente. Estos *threads* deben terminar cuando no haya más clientes. Los *threads* servidores estarán continuamente solicitando mensajes al buffer y respondiendo las respectivas consultas (responder consiste en incrementar el valor del mensaje y avisarle al cliente que puede continuar). El número de servidores también debe ser un número arbitrario.

El número de clientes, el número de servidores, el número de consultas de cada uno de los clientes y el tamaño del buffer deben estar en un archivo, el cual será procesado por el `main` del programa.

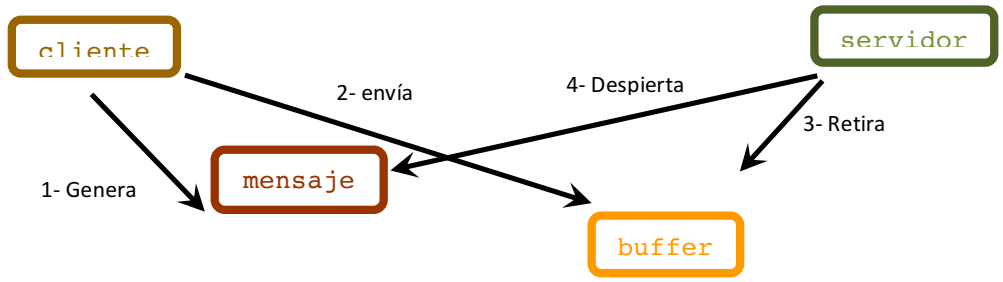
Diseño:

En el sistema tendremos: clientes, servidores, buffer y mensajes. Los clientes y servidores son los antes descritos.

En cuanto al buffer, es un sitio donde los clientes almacenan los mensajes para que sean recogidos por los servidores; este buffer debe tener una cierta capacidad limitada, y funcionar en esquema productor-consumidor. Por su parte, los mensajes son objetos con la consulta que hace el cliente, y donde el servidor deja la respuesta.

El funcionamiento es el siguiente: un cliente genera un mensaje, e intenta depositarlo en el buffer; si no es posible, se queda en espera activa. Sin embargo, el cliente debe ceder el procesador después de cada intento (método yield). Una vez depositado el mensaje, el cliente debe quedar a la espera de la respuesta del servidor; pero esta vez la espera se realiza dormido sobre el mismo objeto mensaje (espera pasiva).

Cada servidor, por su parte, está continuamente intentando retirar mensajes del buffer; si no es posible, vuelve a intentarlo (espera activa). Sin embargo, el servidor debe ceder el procesador después de cada intento (método yield). Una vez retirado el mensaje, genera una respuesta, y procede a despertar al cliente que se encuentra a la espera dormido en el mensaje.El esquema general es el siguiente:



Tanto el cliente como el servidor se comunican con el buffer; no se comunican directamente entre ellos. El buffer debe recibir la información de cuántos clientes hay, pero no de cuántos mensajes van a circular. El tamaño del buffer debe ser configurable; puede ser mayor, menor o igual al número de clientes.

Condiciones de entrega

- En un archivo .zip entregar los fuentes del programa, y un documento word explicando el diseño y funcionamiento del programa. En particular, para cada pareja de objetos que interactúan, explique cómo se realiza la sincronización, así como el funcionamiento global del sistema. El nombre del archivo debe ser: caso1_*login1*_*login2*.zip
- El trabajo se realiza en grupos de máximo **3** personas de la misma sección. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes).
- Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación puede afectar la nota de todos los miembros.
- El proyecto debe ser entregado por Sicua+ por uno solo de los integrantes del grupo. **Al comienzo del documento, deben estar los nombres y carnés de todos los integrantes del grupo**.
- **Se debe entregar por Sicua+ a más tardar el 13 de septiembre a las 23:55 p.m.**