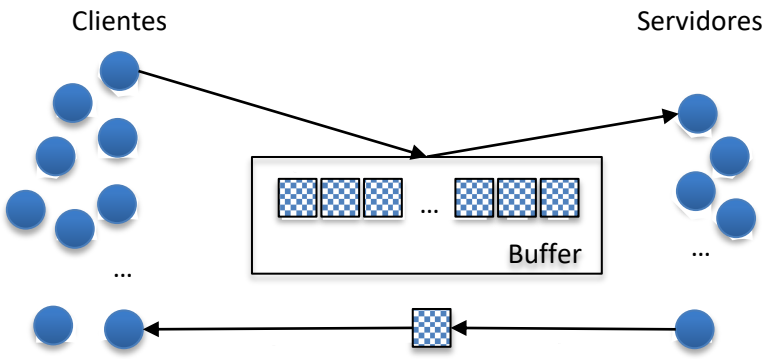
Universidad de los Andes
Departamento de Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional

Caso 1
Manejo de la Concurrencia

Queremos implementar un esquema con múltiples servidores que manejen concurrencia para responder a las consultas que reciben:

- Los clientes generan consultas que consisten en números generados en secuencia o al azar y esperan una respuesta del servidor que consiste en el número de la consulta incrementado en 1. Después de leer la respuesta, descartan dicha información.
- Los clientes no se comunican directamente con los servidores, Lo hacen por medio de un buffer. Esto permite implementar un esquema de atención que no depende del número de clientes o del número de servidores.
- Los servidores buscan las consultas en el buffer, las leen, generan una respuesta y comunican al cliente que generó dicha consulta que su respuesta está lista.
- La siguiente figura presenta el esquema de comunicación. Más adelante se aclaran los detalles de dicha comunicación.



Objetivo

Diseñar un mecanismo de comunicación para manejar las consultas de múltiples clientes. Para este caso, los clientes y los servidores 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`, `notifyAll y yield`.

Funcionamiento

Cada *thread* cliente genera una consulta, la deposita y espera la respuesta. Los clientes repiten este comportamiento un número determinado de veces y terminan. El número de clientes y el número de mensajes que envía cada uno deben ser parámetros de configuración (cada cliente recibirá al comienzo un número particular de consultas que deberá enviar). Para cada consulta, el *thread* cliente debe generar un objeto de tipo Mensaje e

inicializarlo, después lo envía. Cuando termina de enviar las consultas, 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 buscando consultas almacenadas en el 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 parámetro de configuración.

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 debe ser procesado por el `main` del programa.

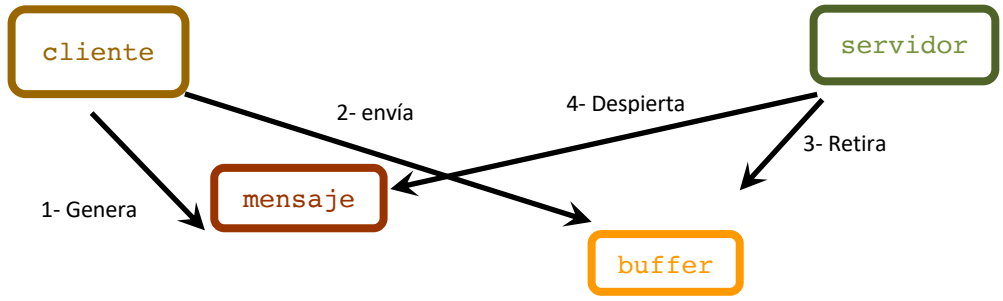
Diseño:

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 o al número de servidores.

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 **2** 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 word, deben estar los nombres y carnés de los integrantes del grupo**. Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente, sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
- **Se debe entregar por Sicua+ a más tardar el 20 de febrero a las 23:55 (p.m.)**