**INFRAESTRUCTURA COMPUTACIONAL**

**CASO UNO**

**Descripción general:**

El programa diseñado permite la creación de un numero determinado de clientes y de servidores (cantidad de clientes, cantidad de servidores y numero de mensajes que los clientes envían están determinados por el usuario a través del archivo info.properties en la carpeta data). Con los cuales los primeros envían mensajes que se almacenan en el buffer y los segundos los van procesando para luego seguir la comunicación.

La clase Main posee el método principal en donde se lee el archivo de propiedades y se crean los clientes y servidores con las características especificadas.

**Cliente genera mensaje:**

Al iniciar el programa se generan todos los clientes que habrá durante la ejecución, cada uno de ellos representa un Thread (La clase cliente extiende a Thread) que ejecuta el método run() de la clase Cliente. Al momento de crear un cliente se requiere recibir por parámetro el buffer donde guardar los mensajes y la cantidad de mensajes que debe enviar el cliente antes de terminar la ejecución. Posteriormente se crea el mensaje con un número que comienza en 0 al primer mensaje.

Al finalizar el envío del mensaje se genera una notificación a los threads dormidos en buffer (serán los servidores disponibles dado que no había mensaje hasta ese momento).

**Cliente envía mensaje a buffer:**

Cuando comienza la ejecución del run() de cada uno de los threads estos entran en un loop hasta que la cantidad de mensajes por enviar sea igual a 0. Dentro de esta iteración existen dos tipos de situaciones en las que puede caer el Thread; La primera es que el buffer tenga espacio, con lo cual se envía el mensaje al buffer y se deja esperando (wait()) al Thread dentro de su mensaje con la llamada de un método a ese objeto. La segunda opción es que el buffer no tenga espacio para recibir el mensaje por lo cual el cliente hará una espera activa en donde preguntara si el buffer ya tiene espacio hasta que efectivamente lo tenga, en los momentos en que no tenga el espacio, el cliente libera el procesador (yield()) hasta tener espacio dentro del buffer.

**Servidor retira mensaje de buffer:**

Al inicio del programa se inicializan un numero n de servidores definidos por el usuario en el documento que lee el programa. Al iniciar el thread (al entrar en el método run() ) se pide un mensaje a buffer, en caso de tenerlo, la ejecución continua normalmente, se aumenta en uno el número recibido en el mensaje y este se retira del buffer. En caso contrario, en el que el buffer este vacío, el servidor procede a dormirse en el buffer (wait() ) a esperar al momento en que un cliente agregue un mensaje y lo despierte.

**Servidor despierta a cliente en mensaje:**

Cuando el servidor termina de procesar un mensaje leído del buffer, se procede a despertar al thread dormido en ese mensaje, este thread es el cliente dueño del mensaje, con lo cual cuando se despierte puede seguir la ejecución normal del programa y la comunicación. O la finalización del thread en caso de haber cumplido con la cantidad de mensajes a enviar establecida por el usuario.

**Consideraciones adicionales por clases:**

Cliente:

Cuando un cliente cumple con la cantidad de mensajes por enviar, actualiza una variable en la clase buffer que representa la cantidad de clientes por atender en el sistema. Esto le permite a la clase servidor preguntarle al buffer si todavía se requiere la ejecución de los servidores.

Servidor:

Los servidores quedan “encendidos” hasta que la cantidad de clientes atendidos según el atributo en buffer es igual a 0.

Buffer:

Dado que esta clase puede experimentar la concurrencia de varios servidores al tiempo, los métodos que son usados por varios threads son sincronizados.

Mensaje:

Dado que esta clase puede experimentar la concurrencia de varios clientes y servidores al tiempo, los métodos que son usados por varios threads son sincronizados.