**Integrantes:**

Henry Fabián Vargas 201315811

Mario Eduardo Hernández 201312862

**Caso I.**

**Manejo de la Concurrencia**

Se crearon 4 clases y un archivo properties, entre las 4 clases se ubican Mensaje, Buffer, Cliente y Servidor. A continuación enunciamos las relaciones entre estas entidades definidas.

**Funcionamiento General:**

Básicamente el programa lee un archivo properties que posee la cantidad de clientes que van a enviar mensajes, la cantidad de mensajes que puede soportar el buffer, el número de Threads del servidor y el número de mensajes que envía cada cliente. Con esta información el programa crea al servidor con sus respectivos Threads, al buffer y a los clientes ; estos clientes empiezan a almacenar sus mensajes en el buffer , esperando que alguno de los Threads del servidor los conteste y por ende despierte al cliente para que siga enviando sus mensajes. Cuando un cliente envía todos sus mensajes le notifica al buffer que acabó con sus mensajes , y cuando todos los clientes hayan notificado su terminación , el buffer termina su ejecución avisándole al servidor este hecho y él acaba su ejecución también.

**Mensaje: (Espera Activa)**

Esta clase representa un mensaje enviado por un cliente, al momento de aumentar su atributo del número, se activa un notify(), éste se activa desde cada thread del servidor que al recibir su mensaje aumenta el valor del entero. De igual manera contiene un wait() que se llama en caso de que el buffer esté lleno y no pueda recibir el mensaje.

**Buffer: (Esquema Productor-Consumidor)**

Buffer contiene un atomicBoolean que indica si el buffer cumplió con todo el número de clientes y se liberó de todos los mensajes. Se usó atómico puesto que es un atributo que cierra los threads del servidor. Adicionalmente tiene un notify() que se activa cuando ingresa un mensaje (despierta al cliente que lo llama), un wait() cuando la lista de objetos en el buffer está vacía (duerme los thread del servidor) y un notifyAll() que avisa a todos los threads del servidor que ya terminaron su labor.

**Cliente: (Método yield() y Espera Activa)**

El cliente se encarga de enviar los mensajes definidos a él, en caso de que no se halla podido enviar el mensaje al buffer, se realiza el método yield. Al final de que se puedan enviar todos los mensajes, se envía la notificación al buffer de que el cliente se retira pues ya envió sus mensajes.

**Servidor: (Espera Pasiva)**

El servidor corre un número específico de threads que se encarga de leer mensajes desde el buffer, al momento que puede ser accedido alguno, se procesa el mensaje, y se espera a que haya una notificación general que termine el proceso del servidor una vez se retiren los clientes y procesen todos los mensajes.

**Estructura general del properties:**

capacidadBuffer=n

numeroClientes=n

numeroThreads=n

numeroMensajesi=n

Donde n es un número entero positivo cualquiera

Donde i es un número entero positivo secuencial (0,...,n)

Todos los números pueden ser diferentes excepto i que debe seguir una secuencia desde 0 hasta el número de clientes que se desean probar y el n representa el número de mensajes.