**Documento – Caso1**

Santiago Esteban Gamboa - 201723080

Juan José Martínez – 201729865

ISIS - 2203

**Funcionamiento Global**

Se realizó según los lineamientos dados por el documento general, estableciendo de la siguiente manera:

* Mensaje: El cual va a ser el elemento que permitirá probar el comportamiento de la aplicación a nivel global, respecto al manejo de hilos a la hora de realizar sincronización y exclusión mutua.
* Cliente: Hacen parte de los hilos de la aplicación y son un elemento importante sobre el cual se plantean restricciones a la hora de acceder al buffer. Realizan espera pasiva o activa dependiendo de si ya han depositado un mensaje.
* Servidor: Hilos de espera activos que reaccionan ante la presencia de mensajes dentro del buffer. Gracias a ellos el buffer se puede desocupar relativamente rápido, además que eso permite la resolución de más mensajes de clientes.
* Buffer: El intermediario de nuestra aplicación y aquello que nos permite realizar procesos de sincronización y exclusión mutua, tanto a la hora de entrar un mensaje como a la hora de retirarlo.
* Main: Clase principal. Desde aquí es donde se lanza la aplicación y se crean los hilos de ejecución, también se obtienen los datos del archivo de propiedades el cual indica el número de clientes, servidores, mensajes y el tamaño del buffer.

**Interacción**

En este caso tenemos que tratar dos problemas importantes: variables concurrentes e inanición. Las variables concurrentes por proteger en este caso son la lista de mensajes y la capacidad del buffer. En cuanto a la inanición, se evita mediante el método yield, cediendo el procesador por parte de ambos tipos de threads (clientes y servidores). De esta manera garantizamos el cumplimiento adecuado de las reglas establecidas a la hora de realizar la comunicación del proyecto:

* Cliente-Buffer: Conceptualmente el cliente intenta agregar mensajes al buffer, de no ser posible se queda en espera activa. A su vez, el mismo cliente cede el procesador al no poder agregar el mensaje, posteriormente cuando el buffer se vacíe el cliente tendrá la oportunidad de intentar ingresar el mensaje de nuevo. Una vez agregado el mensaje, el cliente se queda esperando de manera pasiva a que su mensaje sea respondido.
* Servidor-Buffer: A la vez que el cliente agrega mensajes al buffer, el servidor es el encargado de retirarlos y responderlos. Similar al cliente, el servidor se mantiene en espera activa revisando constantemente si hay mensajes por responder, de no ser así cede el procesador. Otra forma de ver este tipo de espera es como un elemento asíncrono, ya que el servidor no sabe en qué momento llega un mensaje al buffer.
* Mensaje-Buffer: El buffer como es sabido almacena los mensajes, sin embargo, es necesario tener en cuenta el acceso concurrente a la lista de mensajes en el Buffer. En este caso se debe sincronizar el acceso a la estructura de datos correspondiente y a los mensajes que hay en ella. Esto se realiza mediante un monitor **msg,** el cual permite no sólo regular la entrada de los mensajes, sino la salida en el momento en que un servidor vacío esté buscando algo por atender.