Juan Fernando Castañeda Gutiérrez 201623311

Santiago Carrero Gil 201712470

**Caso 1 Diseño**

Nosotros diseñamos cinco clases: Main, Buffer, Servidor, Cliente y Mensaje.

La clase Main se encarga de crear el buffer común, los threads de servidor y cliente, y poner a ejecutar los últimos.

La clase Buffer se encarga de actuar como buffer entre servidores y clientes. Cuenta con una lista encadenada para almacenar los mensajes, siempre que estos no excedan una cantidad de n (parámetro especificado al crear la clase). Tiene además un atributo llamado númeroClientes, que se encarga de contar los clientes aún activos, con sus respectivos métodos para que un cliente pueda registrar su entrada y su salida, y para que un servidor pueda asegurarse de que aún existen clientes en circulación. Por último, un método para dejar un mensaje en caso de ser cliente (que retorna falso en caso de no poder dejar el mensaje) y un método para sacar un mensaje (que retorna null en caso de no existir ningún mensaje). Los métodos que agregan y sacan mensajes están sincronizados en la bolsa del buffer, y los métodos que controlan la entrada y salida de clientes funcionan en sincronización también, pero sobre una bolsa diferente.

La clase Mensaje tiene la información que debe cargar, en forma de un atributo llamado consulta, y un atributo llamado “esperando”, que sirve para dar información a un servidor de si ya el cliente está dormido sobre el mensaje.

La clase Servidor en su método run confirma siempre en su ejecución si aún existen clientes o si aún hay mensajes en la cola antes de seguir ejecutando. Luego si existen clientes pero no mensajes en la cola, hace yield() para liberar procesador en espera activa. Si a cambio hay mensajes en cola, saca uno y espera a que el atributo de mensaje “esperando” sea true, para luego sí modificar su valor y despertar al cliente dormido en el mensaje (aquí es necesaria la sincronización con mensaje para levantar al cliente dormido en el mensaje). El atributo “esperando” asegura que el servidor no despierte a un cliente que aún no se ha dormido, y por tanto la ejecución del programa no termine en un loop infinito.

La clase Cliente tiene una cantidad de mensajes. El método run de Cliente se ejecuta un número cantidadDeMensajes de veces, cada una de esas veces genera un nuevo mensaje e intenta dejarlo en el buffer. Si el buffer está lleno, hace yield() en espera activa para liberar el procesador. Ahora, si es capaz de meter el mensaje en el buffer, intenta dormirse dentro del mensaje, pero antes cambiando el valor de “esperando” de mensaje a true. Todo esto último lo ejecuta en exclusión mutua sobre mensaje para que no se cambie el valor de “esperando” sin él estar dormido sobre el mismo.