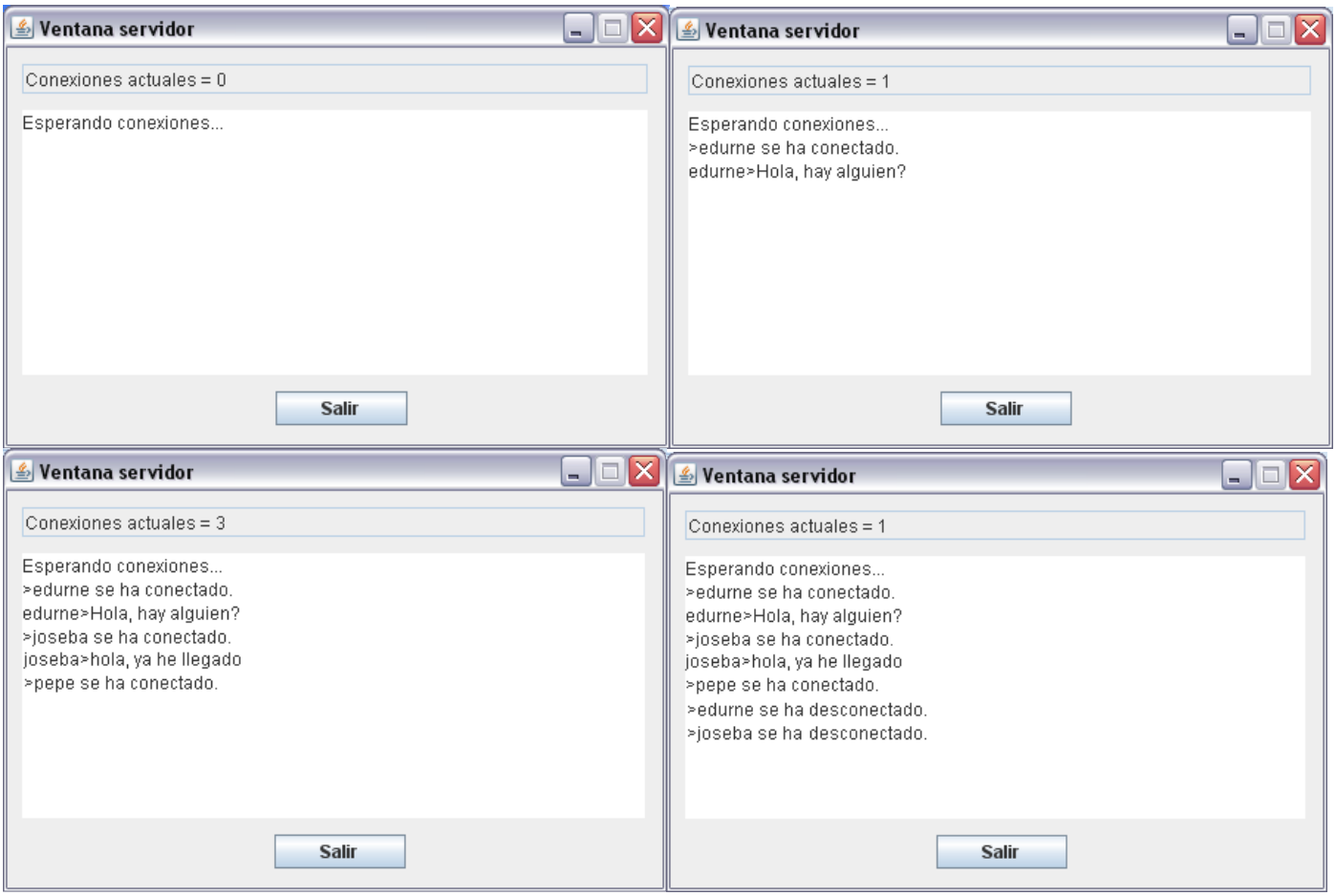
Ejercicio 3: Chat avanzado

En este chat, el servidor actuará de intermediario para que dos o más clientes puedan entablar una conversación.

INTERFAZ DEL SERVIDOR

El servidor muestra la siguiente interfaz:



En el TextField superior (no editable) se muestra el número actual de clientes conectados.

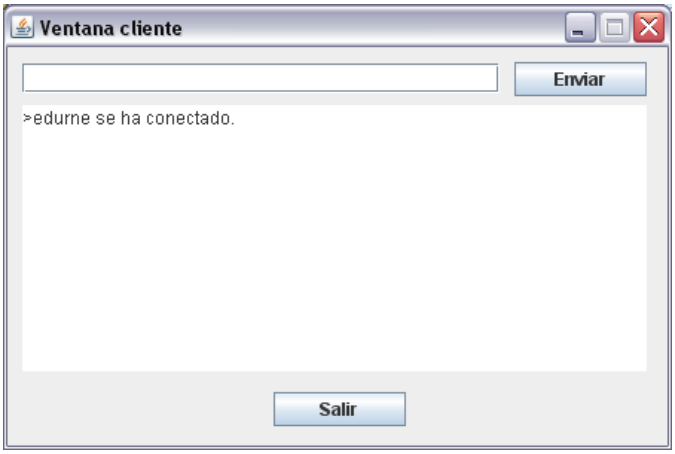
En el TextArea (no editable), se va mostrando todos los acontecimientos que van ocurriendo: cuando se conecta un cliente, cuando se desconecta, los mensajes que se envían…

La ventana no se puede maximizar y el botón 'X' de salida no hace nada. Es necesario pulsar el botón 'Salir' para terminar la aplicación.

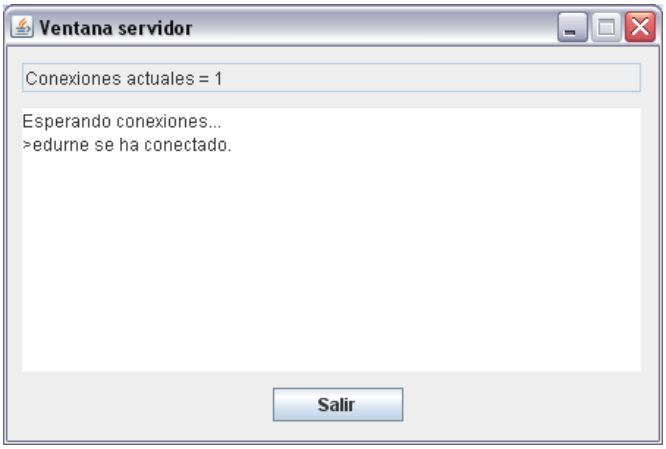
INTERFAZ DEL CLIENTE

El cliente muestra una ventana para introducir un nick. Si se deja en blanco o si no se puede conectar al servidor se muestra un mensaje. Esta ventana no se puede maximizar.

Una vez se ha realizado la conexión con el servidor se muestra la siguiente ventana (y se cierra la anterior). Esta ventana no se puede maximizar y el botón 'X' de salir no hace nada, es necesario pulsar el botón 'Salir'.



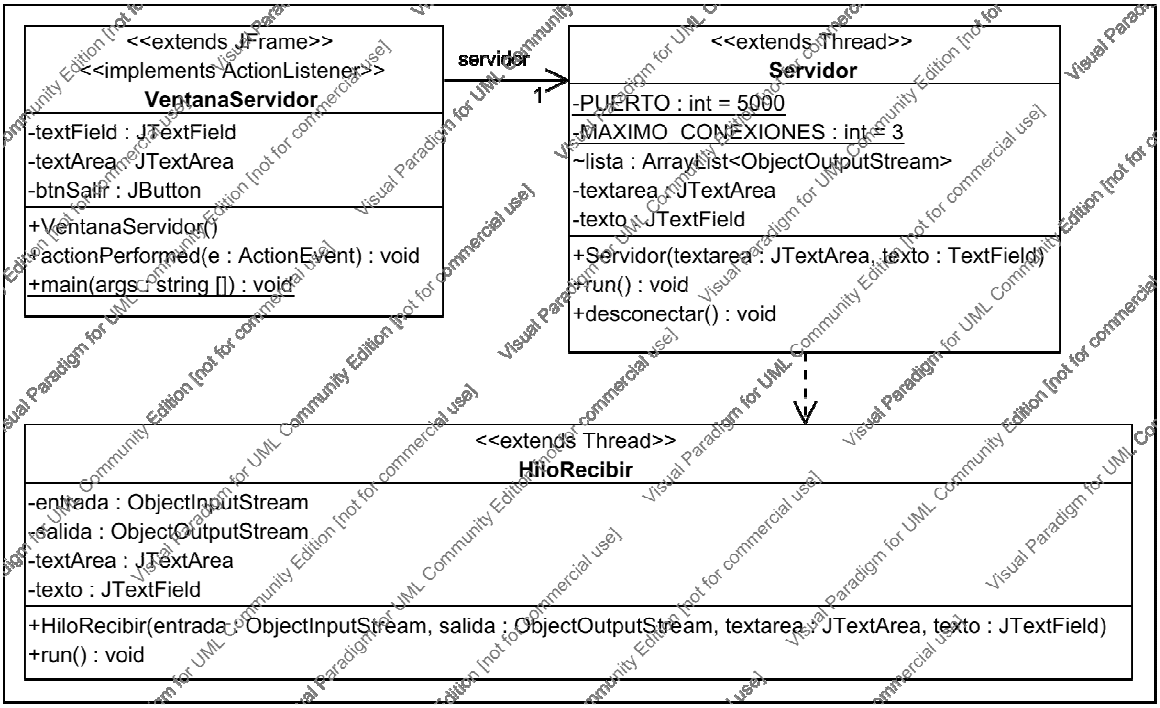
En el servidor también se tiene que mostrar que un cliente se ha conectado, y se actualizará el número de conexiones actuales.

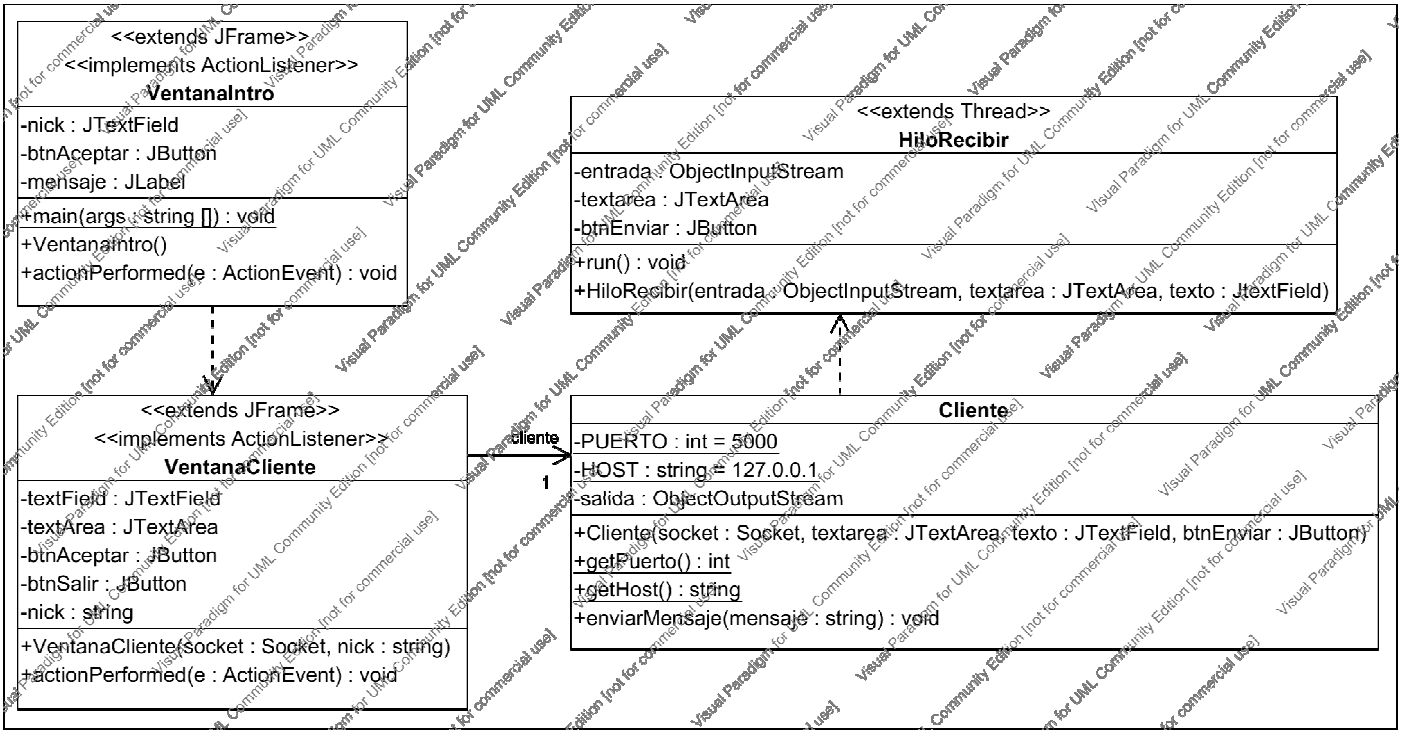


Cuando un cliente escriba en el chat, se visualizará tanto en su ventana de chat, como en la del servidor, como en la del resto de clientes que están conectados. Para escribir en el chat, en el TextField superior se escribe el texto que se desea enviar y se pulsa el botón 'Enviar'. Si no se ha escrito nada, el botón no hará nada. Tras pulsar enviar se limpiará el TextField.

DIAGRAMAS DE CLASES

A continuación se muestra el diagrama de clases del cliente y del servidor. Son orientativos, esto quiere decir que lo podéis hacer así, pero también de otra manera diferente o mejor.





FUNCIONAMIENTO SERVIDOR

La función main de VentanaServidor se encarga de crear y visualizar la ventana. En el constructor de la ventana, se crea un objeto Servidor (es un Thread) y se inicia.

Cuando se pulsa el botón 'Salir' se llama al método desconectar del objeto Servidor creado en el constructor.

Este hilo, en su método run, crea el ServerSocket y muestra en la ventana que el número de conexiones actuales que es 0 y que está a la espera de conexiones. Después, en un bucle infinito, si no se ha alcanzado el número de máximas conexiones, espera a que un cliente se conecte.

Cuando un cliente se conecta, se crean sus streams de salida y de entrada. El stream de salida se guarda en un arraylist. Se actualiza el número de conexiones a mostrar en la ventana y se indica que un cliente se ha conectado.

Entonces, se crea otro hilo, HiloRecibir, y se inicia.

Este segundo hilo (habrá uno por cada cliente conectado), en su método run, en un bucle infinito, lee del stream de entrada el mensaje y lo reenvía a todos los streams de salida que contiene el arraylist (incluído él mismo).

NOTAS ADICIONALES

Siempre se enviará junto al mensaje el Nick de la persona que lo envía. En todos los clientes y en el servidor, siempre que alguien se conecte o desconecte aparecerán los mensajes "*Nick entra en el chat"* o *"Nick ha abandonado el chat"*. Además, en cada ventana cliente, se mostrará cómo título el Nick del usuario.

Un cliente solo se puede desconectar cuando pulsa el botón 'Salir'. En ese caso se envía al servidor un mensaje con un * y finaliza la aplicación. El hilo del servidor cuando reciba ese * tiene que eliminar del arraylist su stream de salida, y por lo tanto, se actualizará el número de clientes conectados en uno menos.

Cuando se pulsa el botón 'Salir' del servidor, se envía el mensaje * a todos los clientes que estén conectados y se finaliza la aplicación. Cuando los clientes reciban un *, tienen que mostrar en la ventana un mensaje indicando que se ha caído el servidor y deshabilitar el botón y el textField de envío de mensajes.

No es posible crear más de un stream de entrada y de salida para cada cliente. Además, para evitar problemas, siempre hay que crear primero el stream de salida y después el de entrada. Esto lo haremos en el bucle infinito de la clase Servidor, justo antes de crear el objeto HiloRecibir (HiloRecibir recibe en el constructor directamente los streams, no el socket).

MEJORAS OPCIONALES

- En el cliente, al escribir un texto y pulsar la tecla “Enter”, es como si se pulsara el botón “Enviar”.
- El nick tiene que ser único en el chat, si un cliente intenta conectarse con un nick que ya está siendo utilizado, se le pedirá que vuelva a introducir otro nick diferente.
- En el servidor se mostrará una lista con los nicks de los clientes que están en el chat. Cuando un cliente se conecte, se añadirá a la lista. Cuando un cliente se desconecte, se eliminará de la lista. La lista quedará siembre ordenada alfabéticamente.

- El servidor puede seleccionar una persona de la lista y expulsarla del chat.

Al subir el ejercicio a Moodle indicad qué mejoras se han realizado y si hay algo que no funcione o no se haya implementado.