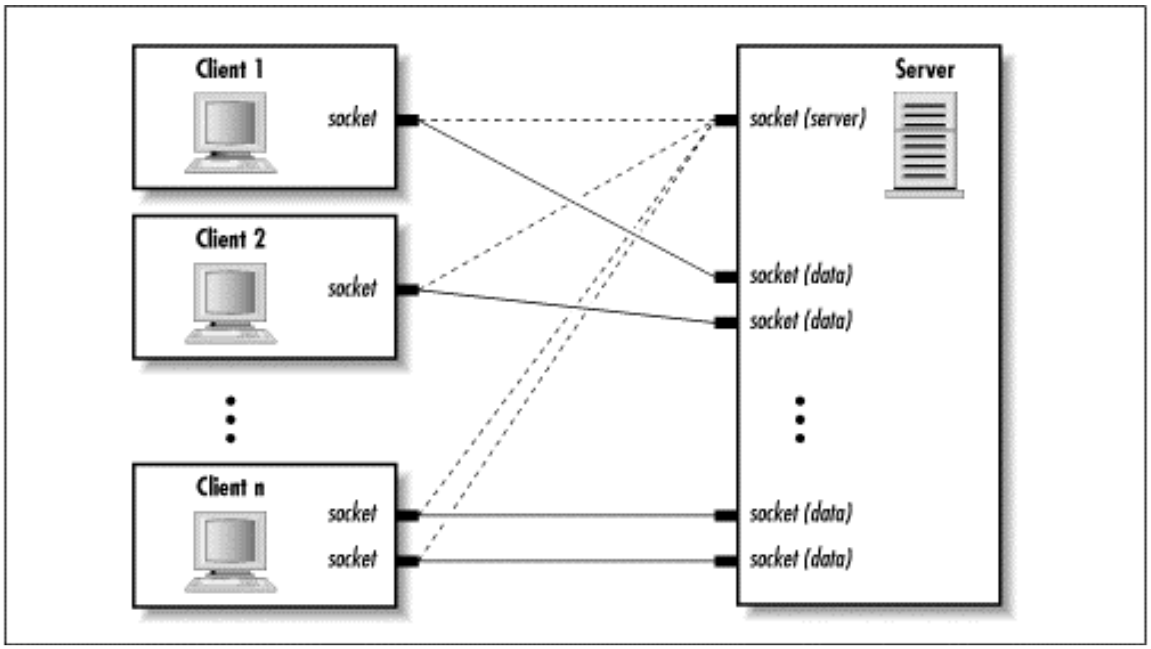
Chat project

In this chat, the server will act as an intermediary so that two or more clients can initiate a conversation.



SERVER USER INTERFACE

The server has a graphical user interface displaying the following:

- Number of current connections (how many clients are connected to the chat).
- Text area displaying all the events that happen during the communication. Example:
    - Waiting for connections…
    - Client_XXX: HELLO
    - Client_XXX: text message
    - Client_XXX: BYE
- Exit button.

CLIENT USER INTERFACE

In the client program, the first step is to login with a nickname. Once the login is done, the following will be displayed:

- Text area displaying all the events that happen during the communication. Example:
    - Client_XXX CONNECTED
    - Client_XXX: text message
    - Client_XXX DISCONNECTED
- Textbox to write a new message in the chat.
- Send button.
- Exit button.

HOW THE CHAT WORKS – SERVER

The main thread of the server program is going to manage the user interface. At the same time, a server thread will be listening for client connections in a given port.

When a client is connected, a new thread will be created to attend this client (for each connected client a new thread will be created). All the clients must be notified that a new client is connected.

Each of the client threads is continuously reading from the input stream of the socket. When a message is read, it is forwarded to all the clients. This means that the message is written in the output stream of all the clients. This message can be a text written by the user in the chat, or a notification that the client is disconnected. In both cases, all the rest of the clients must be notified.

If the server is closed (Exit button), before exiting it will notify all the clients about it.

HOW THE CHAT WORKS – CLIENT

The user must login with a nickname. If the connection with the server is correct, the client sends its nickname to the server. This means that the client writes in the output stream of the socket. Then, a new window is displayed. In case of error, an error message is displayed and the new window is not opened.

In this new window, the nickname of the user is displayed as the title of the window.

To write in the chat, the user writes a message in the textbox and clicks the Send button (or Enter key is pressed). If nothing is written in the textbox, the button does nothing. After clicking Send, the textbox is cleaned. This message is sent to the server through the socket.

To exit the chat, the client clicks the exit button. Before exiting, the client informs the server that it is going to disconnect.

In case the server notifies its disconnection or the connection is lost, the chat must be disabled, and a message like "Server disconnected" must be displayed.

ADDITIONAL NOTES

The nickname of the client is always sent with the text of the message.

When a client connects or disconnects, a message like "Hello" o "Bye" should be sent to notify the server.

All the client threads on the server must share a collection of objects, so that when a client sends a text message connects or disconnects all the connected clients are notified about it.

Very important: it is not possible to create more than one input and output stream for each client socket. And to avoid problems, you always have to create the output stream before the input stream.

Remember to close all the streams and sockets where necessary. Before exiting the server or the client, all streams and sockets must be closed, all threads stopped and all windows closed. Then you can exit (System.exit(0)).

Threads attending blocking calls (ServerSocket's accept() and ObjectInputStream's readObject() are blocking) cannot be interrupted. Just close the streams and the sockets and catch SocketException in the thread's run method.

All the catch blocks must print the stack trace (except in the previous case), do not delete them. It is recommended to set the name of the threads with the nickname of the client (in the server) and to print it in the catch, to help debugging.

EVALUATION

Only the applications that implement complete and error-free functionality will be evaluated.

It will be evaluated that the application is:

- Effective: the application does what it has to do
- Reliable: without bugs
- Efficient: efficient algorithms and data structures
- Organized: multilayer architecture, use of design patterns
- Stylish: the code follows the style rules of the programming language
- Documented: the code is commented
- Usable: easy to use, attractive interaction
- Not copied
- Completed with additional functionality

ADDITIONAL FUNCTIONALITY

- The nickname is unique. If a user tries to login with a repeated nickname, it will be notified to choose a different one.
- A list of the nicknames of the clients who are logged in the chat is displayed in the server user interface. When a client connects, it is added to the list. When a client disconnects, it is removed from the list. The list is sorted alphabetically.
- The server can select a user from the list and drop them from the chat.

SUGGESTION OF CLASS DIAGRAM

Note: HashMap is not thread-safe, use ConcurrentHashMap instead