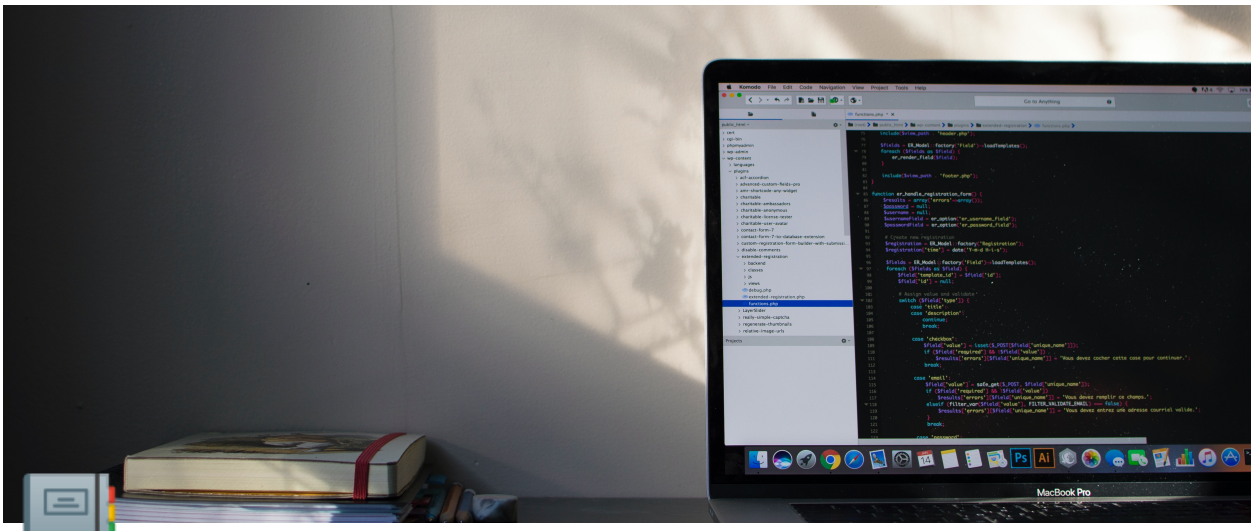



# READ ME SAMPLE:

THREE TIERS OF ANY SOFTWARE:

1. GUI - GRAPHICAL USER INTERFACE
2. BL - BUSINESS LOGIC
3. DATABASE

💡 GRAPHICAL USER INTERFACE - design part

▼ PANEL & LABEL

1. contains EXIT button
2. Label (CLIENT or SERVER)

▼ TEXTBOX

1. Text Field where we type our messages
2. To enter IP Address of server (*Only on Client's UI*)

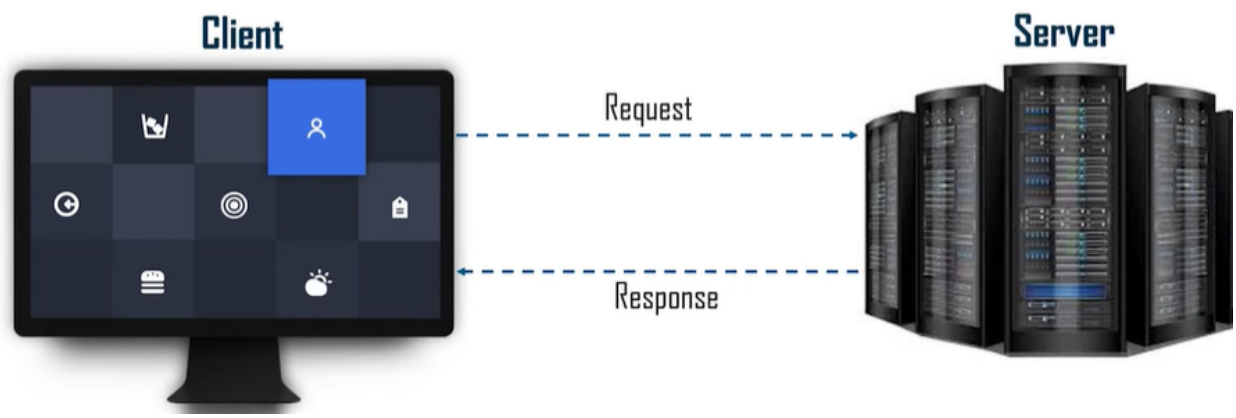
▼ BUTTON

1. SEND
2. EXIT (Disconnect from chat)

▼ TEXTAREA

1. Incoming & Outgoing chats with Vertical Scroll Bar attached

*CHAT IMPLEMENTATION* - SERVER & CLIENT .




---

💡 BUSSINESS LOGIC - adds functionality/working for application

***SOCKET PROGRAMMING*** **-**

- Client side program —> to send a request to Server
- Server side program —> to get response back to Client by accepting it's request

*Socket acts as a inter-link which helps for connection to network between Client and Server. To connect from one machine to another we need Socket Connection.*

1. SERVER SIDE PROGRAMMING:

▼ STEPS:

1. Establish a Socket Connection and start server
2. Wait for Client Request
3. Communication with client using input and output stream
4. Closing the connection

- A port is created. We can choose our port to be anything other than reserved ports which are from 1 to 1024.. So, anything >1024 is fine
- We declare the Socket accept() method which returns the socket and establishes a connection between client and server
- din and dout tracks i/p , o/p streams of data that is being transferred
- An empty string msgin is created
- Then, Whatever string is received that is to be read and stored inside msgin
- Finally we print the msgin i.e.., msg received from client on to Textarea including messages sent from server..

2. CLIENT SIDE PROGRAMMING:

▼ STEPS:

1. Establish a Socket Connection
2. Provide IP Address & TCP port number of Server on which you wish to communicate.
3. Communicate over Socket connection — streams are used to input and output the data
4. Close the connection

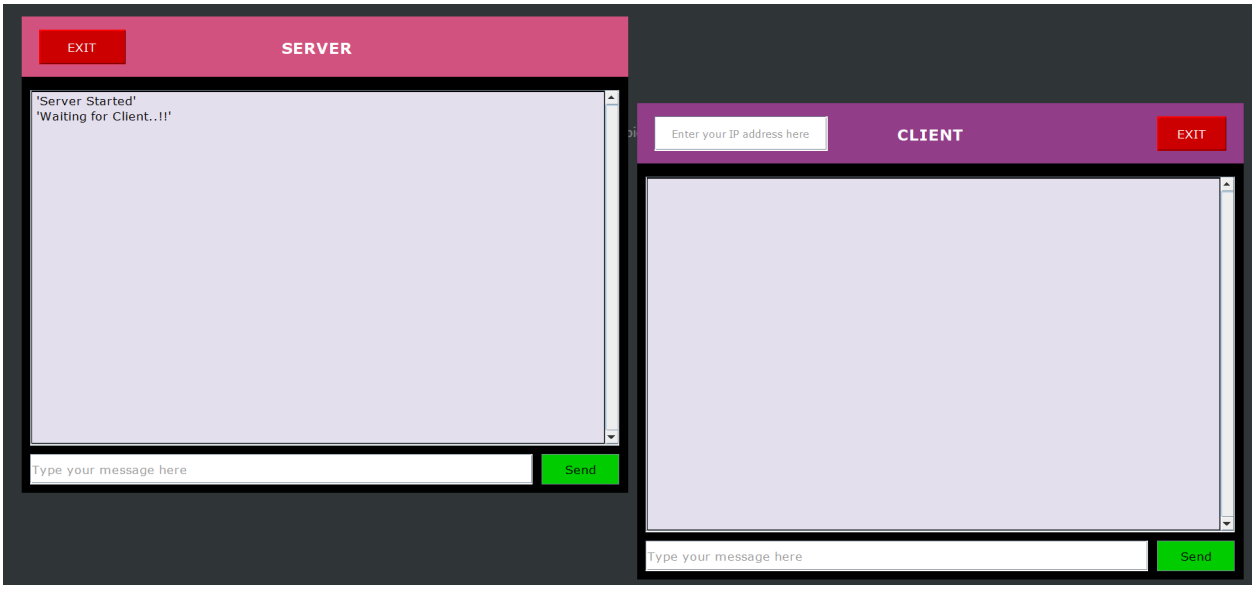
- No need to create serversocket again bcoz we will have only one server & all interactions will be done along allotted port to various clients.
- Here Socket is declared as new socket with IP Address and allotted/target port number of server.

---

THEORY:

1. TCP - Transmission Control Protocol
    - Connection Oriented Protocol
    - To obtain this we require Server and ServerSocket classes from java.net.* package
    - For transfer of data we require DataInputStream and DataOutputStream classes from java.io.* package
    - Lossless & Reliable Communication is achieved through this protocol.

## FINAL UI:




## INSTRUCTIONS / MANUAL:

1. Run server first and it automatically starts and waits for Client to send request
2. Then run client and Enter IP Address of server in the TextField (ToolTipText is also available)

    *TO-DO : to test Exception cases*

a. Try entering wrong host names / Wrong IP address

b. Try running CLIENT first / after starting server Disconnect the server

The above Exceptions are informed to the user as a error so they can check their connectivity or entered IP address and stuff

3. If you entered correct IP Address after starting server then it appends

   a. Socket Connection Established succesfully...!! — Client's Textarea.

   b. 'Client request Accepted..Connection Established..!!' — Server's Textarea

   NOTE : After establishing the connection you can no longer edit IP Address TextField.

4. Now, you are ready for interaction.
5. Commmunicate as much as you want by Typing your messages in the TextField provided and click SEND button.
6. Finally, when you would like to disconnect the chat Click EXIT button.
7. When either of client or server is disconnected and then you try to send a text to other, the text will be appeared on the respective TextArea but it is not sent.
8. This is informed to the user through a warning dialog box.

*—M.V.S.M.SATVIKA*