Java Chat Application – Interview Questions & Answers

---

1. What is this project?

A client-server chat application built using Java Sockets, where multiple clients can connect to a server and exchange messages.

2. Which protocol is used? Why?

- TCP is used (Socket, ServerSocket).
- Provides reliable, ordered, error-free communication, essential for chat.

3. What is the role of ChatServerX.java?

- Starts a ServerSocket on a port.
- Accepts client connections.
- Creates a thread for each client.
- Relays messages between clients.

4. What is the role of ChatClientX.java?

- Connects to the server using Socket.
- Uses streams to send and receive messages.
- Runs continuously until client disconnects.

5. How does communication flow?

1. Server starts and listens on a port.
2. Client connects via IP + port.
3. Both exchange data using input/output streams.

4. Communication continues until one side closes connection.

6. Why is multithreading required in this project?

- Server must handle multiple clients simultaneously.
- Clients need separate threads for sending and receiving messages concurrently.

7. What Java classes are used for networking?

- ServerSocket → Listens for client connections.
- Socket → Represents client-server connection.
- InputStreamReader, BufferedReader → Read messages.
- PrintWriter → Send messages.

8. What happens when the server is down?

Client throws ConnectException because it cannot establish connection.

9. How does the server accept multiple clients?

- Calls accept() in a loop.
- For each new client → starts a new thread.

10. What exceptions are handled in this project?

- IOException (input/output failure).
- SocketException (connection reset).
- UnknownHostException (invalid server IP/host).

11. What is the difference between TCP and UDP?

- TCP: Reliable, connection-oriented (used here).
- UDP: Faster, connectionless, but unreliable.

12. Why not use UDP for chat?

Because UDP doesn’t guarantee delivery or order of messages → chat would lose/misorder messages.

13. How do you send a message?

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

out.println("Hello");

14. How do you receive a message?

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String msg = in.readLine();

15. What is the role of ports in this project?

- Port uniquely identifies a server process.
- Example: ServerSocket(5000) → server listens on port 5000.
- Clients connect to the same port.

16. Can multiple clients run on the same machine?

Yes, if they connect using the same server IP and port, since server differentiates them by socket connections.

17. What happens if two clients send messages at the same time?

Server handles them in separate threads → messages are processed independently.

18. How do you stop the server gracefully?

- Close all client sockets.
- Call serverSocket.close().
- Interrupt running client threads.

19. What is a blocking call in this project?

- ServerSocket.accept() → waits until a client connects.
- BufferedReader.readLine() → waits until a message is received.

20. How do you handle multiple client broadcasts?

- Maintain a list of client output streams in the server.
- When one client sends a message → send it to all connected clients.

21. How is the project executed?

1. Run ChatServerX (server starts).
2. Run one or more ChatClientX programs.
3. Clients exchange messages via server.

22. What are common networking errors you faced?

- Port already in use (BindException).
- Server not running (ConnectException).
- Connection lost (SocketException).

23. How can you improve this project?

- Add GUI (Swing/JavaFX).
- Support private chat.
- Implement user authentication.
- Add encryption for secure chat.
- Use database to store chat history.

24. How do you differentiate clients?

- Assign unique IDs or usernames on connection.
- Server maintains a map of clients.

25. What is the role of threads in ChatServerX?

Each client is handled by a separate thread → allows parallel communication.

26. How would you implement a group chat feature?

- Maintain a list of client sockets.
- When a client sends a message → broadcast to all others.

27. What is the difference between single-threaded and multi-threaded server?

- Single-threaded → handles one client at a time.
- Multi-threaded → handles multiple clients simultaneously (used here).

28. How do you close a client connection?

- Client sends a disconnect message.
- Server closes client socket.
- Thread handling client is terminated.

29. What is the difference between Socket and ServerSocket?

- Socket → represents a client-server connection.
- ServerSocket → listens for incoming client requests.

30. What is the biggest learning from this project?

Understanding network programming in Java, handling concurrency, and managing real-time communication between multiple systems.