

Java Chat Application

1. Steps in Communication

The chat application consists of a Java-based server and a client that communicates over a TCP connection. The server starts by opening a `ServerSocket` on port `1234` and waits for a client to connect. When the client launches, it opens a `Socket` and connects to the server using the same port.

Once connected, both server and client use `BufferedReader` and `PrintWriter` for message exchange. The client sends a message, and the server reads it, prints it to the console, and sends back an acknowledgment. This continues in a loop until the client sends "quit", which signals both sides to terminate the session gracefully.

2. Understanding of Socket Creation and Data Flow

In the server, socket creation begins with:

```
ServerSocket serverSocket = new ServerSocket(1234);
```

```
Socket clientSocket = serverSocket.accept();
```

- `ServerSocket` listens for incoming connections on a specific port.
- `accept()` blocks until a client connects, then returns a `Socket` object for communication.

In the client:

```
Socket socket = new Socket("localhost", 1234);
```

- This creates a socket and connects it to the server's IP and port.

Data flow:

- Input: Messages are read using `BufferedReader` connected to the socket's input stream.
- Output: Messages are sent using `PrintWriter` connected to the socket's output stream.

The use of `BufferedReader` and `PrintWriter` enables easy, line-based communication between the client and server. Both ends loop through sending and receiving messages until the session ends.