```
import java.io.*;
import java.net.*;
import java.util.*;

// ========== SERVER CODE ========== //
class ChatServer {
    private static Set<ClientHandler> clientHandlers = new HashSet<>();

    public static void startServer(int port) {
        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Server started on port " + port + ". Waiting for clients...");

            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New client connected.");
                ClientHandler clientHandler = new ClientHandler(socket);
                clientHandlers.add(clientHandler);
                new Thread(clientHandler).start();
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    static void broadcast(String message, ClientHandler sender) {
        for (ClientHandler ch : clientHandlers) {
            if (ch != sender) {
                ch.sendMessage(message);
            }
        }
    }

    static void removeClient(ClientHandler clientHandler) {
        clientHandlers.remove(clientHandler);
    }

    static class ClientHandler implements Runnable {
        private Socket socket;
        private PrintWriter out;
        private BufferedReader in;

        public ClientHandler(Socket socket) {
            this.socket = socket;
        }
        public void sendMessage(String message) {
            out.println(message);
        }

        @Override
        public void run() {
            try {
                in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                out = new PrintWriter(socket.getOutputStream(), true);

                String name = in.readLine();
                System.out.println(name + " has joined.");
                ChatServer.broadcast(name + " joined the chat", this);

                String msg;
                while ((msg = in.readLine()) != null) {
                    System.out.println(name + ": " + msg);
                    ChatServer.broadcast(name + ": " + msg, this);
                }
            } catch (IOException e) {
                System.out.println("Client disconnected.");
            } finally {
                ChatServer.removeClient(this);
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

// ========== CLIENT CODE ========== //
class ChatClient {
    public static void startClient(String hostname, int port) {
        try (Socket socket = new Socket(hostname, port)) {
            new Thread(new ReadHandler(socket)).start();
            new Thread(new WriteHandler(socket)).start();
        } catch (IOException ex) {
            System.out.println("Could not connect to server.");
            ex.printStackTrace();
        }
```

```
}

static class ReadHandler implements Runnable {
    private BufferedReader reader;

    public ReadHandler(Socket socket) {
        try {
            reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    @Override
    public void run() {
        String response;
        try {
            while ((response = reader.readLine()) != null) {
                System.out.println(response);
            }
        } catch (IOException ex) {
            System.out.println("Disconnected from server.");
        }
    }
}

static class WriteHandler implements Runnable {
    private PrintWriter writer;
    private BufferedReader console;

    public WriteHandler(Socket socket) {
        try {
            writer = new PrintWriter(socket.getOutputStream(), true);
            console = new BufferedReader(new InputStreamReader(System.in));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    @Override
    public void run() {
        try {
            System.out.print("Enter your name: ");
            String name = console.readLine();
```

```
                writer.println(name);

                String message;
                while ((message = console.readLine()) != null) {
                    writer.println(message);
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }
}

// ========== MAIN METHOD ========== //
public class ChatApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Start as (1) Server or (2) Client?");
        int choice = sc.nextInt();
        sc.nextLine(); // consume newline

        if (choice == 1) {
            System.out.print("Enter port number to start server: ");
            int port = sc.nextInt();
            ChatServer.startServer(port);
        } else if (choice == 2) {
            System.out.print("Enter server IP (e.g. localhost): ");
            String host = sc.nextLine();
            System.out.print("Enter server port: ");
            int port = sc.nextInt();
            ChatClient.startClient(host, port);
        } else {
            System.out.println("Invalid choice.");
        }
    }
}
```