Task 3

```
import java.io.*;

import java.net.*;

import java.util.*;

import java.util.concurrent.*;

// ------------------ MAIN ------------------
public class ChatApp {

public static void main(String[] args) {

  Scanner scanner = new

Scanner(System.in);

 System.out.println("Select mode:\n1. Start Server\n2. Start Client");

  String choice = scanner.nextLine();

        try {

 if (choice.equals("1")) {
                ChatServer.startServer();
            }
else if (choice.equals("2")) {
                ChatClient.startClient();
            }
 else {
                System.out.println("Invalid option.");
            }
        }
catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

// ------------------ SERVER ------------------
class ChatServer {
```

```
private static final int PORT = 12345;

private static final Set<ClientHandler> clients = ConcurrentHashMap.newKeySet();

    public static void startServer() throws IOException {

 ServerSocket serverSocket = new ServerSocket(PORT);

 System.out.println("Server started on port " + PORT);

        while (true) {

Socket clientSocket = serverSocket.accept();

 System.out.println("New client connected: " + clientSocket);

            ClientHandler clientHandler = new ClientHandler(clientSocket);

clients.add(clientHandler);
            new Thread(clientHandler).start();
        }
    }

    public static void broadcast(String message, ClientHandler sender) {

for (ClientHandler client : clients) {

 if (client != sender) {
                client.sendMessage(message);
            }
        }
    }

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

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

        public ClientHandler(Socket socket) {
```

```
            this.socket = socket;
        }

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

                out.println("Enter your name:");
                name = in.readLine();
                System.out.println(name + " joined the chat.");
                broadcast(name + " joined the chat.", this);

                String message;
                while ((message = in.readLine()) != null) {
                    if (message.equalsIgnoreCase("exit")) {
                        break;
                    }
                    System.out.println(name + ": " + message);
                    broadcast(name + ": " + message, this);
                }
            } catch (IOException e) {
                System.out.println("Connection lost with " + name);
            } finally {
                try {
                    socket.close();
                } catch (IOException e) {}
                removeClient(this);
                System.out.println(name + " left the chat.");
                broadcast(name + " left the chat.", this);
            }
        }

        public void sendMessage(String message) {
            out.println(message);
        }
    }
}

// ----------------- CLIENT -----------------
class ChatClient {
    private static final String SERVER_HOST = "localhost";
    private static final int SERVER_PORT = 12345;
```

```
public static void startClient() {
    try {
        Socket socket = new Socket(SERVER_HOST, SERVER_PORT);
        System.out.println("Connected to chat server.");

        new Thread(new ReadTask(socket)).start();
        new Thread(new WriteTask(socket)).start();

    } catch (IOException e) {
        System.out.println("Unable to connect to server.");
    }
}

static class ReadTask implements Runnable {
    private final BufferedReader in;

    public ReadTask(Socket socket) throws IOException {
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
    }

    public void run() {
        String message;
        try {
            while ((message = in.readLine()) != null) {
                System.out.println(message);
            }
        } catch (IOException e) {
            System.out.println("Disconnected from server.");
        }
    }
}

static class WriteTask implements Runnable {
    private final PrintWriter out;
    private final BufferedReader console;

    public WriteTask(Socket socket) throws IOException {
        out = new PrintWriter(socket.getOutputStream(), true);
        console = new BufferedReader(new InputStreamReader(System.in));
    }

    public void run() {
        String input;
```

```
            try {
                while ((input = console.readLine()) != null) {
                    out.println(input);
                    if ("exit".equalsIgnoreCase(input)) {
                        break;
                    }
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```