Trabalho 2– Comunicação TCP em Java: Implementação de um Chat

Objetivo: desenvolver uma aplicação em Java que utilize sockets TCP para permitir a comunicação entre múltiplos usuários em tempo real, simulando o funcionamento básico de um mensageiro (chat).

Descrição: o trabalho consiste em implementar um sistema de chat baseado em comunicação TCP. A aplicação deverá ser composta por um **servidor** e múltiplos **clientes**, permitindo que os usuários troquem mensagens entre si por meio de uma conexão persistente.

Requisitos mínimos

1. **Servidor:**
    - Aceitar conexões simultâneas de múltiplos clientes.
    - Reencaminhar as mensagens recebidas de um cliente para todos os demais (broadcast).
    - Exibir no console informações sobre conexões ativas.
2. **Cliente:**
    - Permitir ao usuário enviar mensagens via terminal ou interface gráfica (opcional).
    - Exibir em “tempo real” as mensagens recebidas dos demais usuários.
    - Conectar-se ao servidor.
3. **Comunicação:**
    - Utilizar exclusivamente sockets TCP para o envio e recebimento de mensagens.
    - Garantir o tratamento adequado de exceções.

Funcionalidades:

1. **Identificação de usuários:**
    - Permitir que cada cliente escolha um nome de usuário.
    - Exibir o nome de quem enviou cada mensagem.
2. **Mensagens privadas (comando):**
    - Implementar um comando como /privado :<usuário> :<mensagem> para enviar mensagens privadas entre clientes.
3. **Comando de lista de usuários:**
    - Implementar um comando como /usuarios para exibir a lista de clientes conectados.
4. **Mensagens com carimbo de data e horário:**
    - Incluir a data e hora em que cada mensagem foi enviada ou recebida.

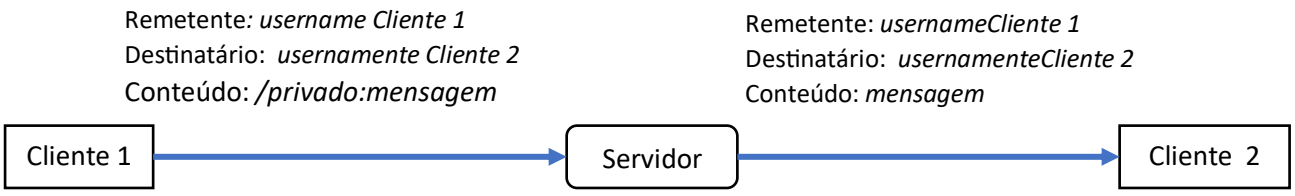
Exemplo: [Data e Hora] <Remetente> -> <Destinatário>: Mensagem

5. **Utilizar a classe Mensagem:**

A comunicação entre cliente e servidor será feita por meio de objetos da classe Mensagem, que deverá ser implementada como uma classe serializável. Essa classe encapsula as informações essenciais da mensagem, como nome do remetente, horário, destino e conteúdo.

Por convenção:

- Quando *destinatario == null*, significa que a mensagem **deve ser enviada para todos os usuários conectados**. Esse comportamento é conhecido como **broadcast**.
- Quando uma mensagem for privada o seu campo conteúdo deve seguir o formato */privado:mensagem*. Ao receber essa mensagem o servidor deve remover o prefixo /privado: do conteúdo, e encaminhá-la para o cliente especificado no destinatário. Conforme ilustrado na figura abaixo.



Critérios de Avaliação

- Funcionamento correto da comunicação entre cliente e servidor.
- Organização e clareza do código-fonte.
- Tratamento de erros e estabilidade da aplicação.
- Implementação de funcionalidades extras (bônus).
- Interface amigável (caso implementada).
- Comentários e documentação do código.

Entrega

- Código-fonte completo (cliente e servidor).
- Relatório simples explicando o funcionamento da aplicação, estrutura do código e instruções de execução.
- (Opcional) Capturas de tela demonstrando a aplicação em funcionamento.