**LABORATÓRIO DE PROGRAMAÇÃO AVANÇADA**
**DÉCIMO TRABALHO PRÁTICO**

# Chat Multithread com Socket TCP

Este trabalho visa a criação de um **chat** simples utilizando **sockets TCP** e **múltiplas threads**. Pode ser feito em **Java** ou **C++**. Imagine se o WhatsApp armazenasse todas as mensagens em uma fila única e as processasse usando uma única thread. Provavelmente, o serviço seria precário e ninguém o usaria. Portanto, a ideia é que o **chat** use **threads** para controlar tanto o fluxo de mensagens **enviadas** quanto das mensagens **recebidas** por cada cliente.

## SERVIDOR

O **servidor** será uma unidade centralizadora das conexões recebidas dos **clientes**, via socket, e terá como responsabilidade o envio das mensagens recebidas de um cliente para **todos os demais clientes** conectados no servidor. Quando um cliente se conecta, o servidor cria uma thread para aquele cliente, ou seja, cada **conexão** terá sua respectiva **thread** e o servidor fará a gestão disso.

Como o servidor vai usar múltiplas threads, a classe servidor deve ou estender a classe **`Thread`** ou implementar a classe **`Runnable`**. É sugerido que esta classe mantenha uma lista de todos os usuários (ou clientes) conectados. Pode-se usar **`Array List`** sem problemas.

Como o servidor vai utilizar múltiplas threads, faz-se necessário a inclusão do método **`run`**, que será executado toda vez que um cliente novo chegar ao servidor. O objetivo deste método é ficar verificando se existe alguma mensagem nova vinda do cliente. Caso exista, esta será lida e, em seguida, esta mesma mensagem deve ser enviada a **TODOS** os usuários conectados no chat.

Uma possível estratégia para o servidor gerenciar todos os clientes conectados seria que o **construtor** da classe servidor recebesse como parâmetro o **socket** da conexão com o cliente e que tal construtor estabelecesse os streams de entrada e saída. Dessa forma, a **lista de clientes** poderia ser simplesmente o **stream** de saída de cada **socket**. Nesse caso, ficará mais fácil enviar a mensagem de um cliente para todos os outros clientes conectados. Obviamente que é necessário percorrer a lista de clientes (em outras palavras, os **streams** de saída dos clientes) e mandar uma cópia da mensagem para cada um.

O método **main** do servidor simplesmente deve gerar o **ServerSocket** e aguardar por conexões do cliente. Quando a conexão com o cliente for estabelecida, deve ser instanciado uma nova thread e chamado o respectivo método **start().**

**CLIENTE**

Cada usuário do chat criará um objeto (instância) do **cliente**, que vai gerar o socket para se conectar com o servidor. Para tanto, o cliente deverá informar o endereço IP do socket do servidor e a respectiva porta. Portanto, é necessário executar o servidor antes. Outro parâmetro é o nome do cliente, que é um identificador para se saber quem está enviando as mensagens.

Um problema interessante do chat é que o cliente pode tanto receber mensagens quanto enviar. Para evitar a possibilidade de uma leitura do teclado bloquear todo o sistema, aqui faz-se necessário que o cliente estabeleça DUAS threads, uma para enviar as mensagens e a outra para receber as mensagens. A sugestão é que a classe cliente tenha, internamente, duas outras classes que estendem a classe Threads. Essas duas novas threads serão instanciadas (e também chamando o método start()) na classe cliente.

**IMPORTANTE**: Implemente uma forma eficaz do cliente sair oficialmente do sistema e, dessa forma, evitar que haja algum envio de mensagens para um socket **não mais existente**.

**EXECUÇÃO**

Abra quatro terminais. No primeiro chame o programa servidor e nos três outros chame três clientes com nomes diferentes. Envie as mensagens de cada cliente e verifique que todos os outros recebem a mesma mensagem.

**OBSERVAÇÕES**

- Se por acaso a especificação do problema não estiver totalmente clara, por favor incluam em um arquivo README todas as **suposições**
- Enviar os CÓDIGOS para o email do **professor** e do **monitor**
- Inclua ***printscreens*** das telas mostrando a execução
- Por favor, coloquem no assunto do email: **[LPAv-TP10]**

- - -
Data de entrega: **06/06/2019** (quinta-feira, até meia-noite)
Após este prazo sera descontado **0,5 por hora** (ou fração) de atraso.