Aula 22 – Cliente Servidor

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

A comunicação falha, pois não existe um servidor para ser conectar, portanto a comunicação é recusada.

Texto

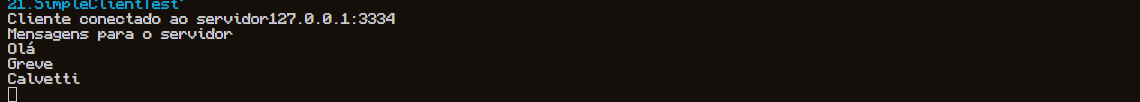
Descrição gerada automaticamente

O servidor é iniciado, porém não recebe nenhuma mensagem pois só o server está ligado, sem receber nenhum input do cliente.

Interface gráfica do usuário, Texto, chat ou mensagem de texto

Descrição gerada automaticamente

Agora ele conecta, pois existe um servidor para conectar e receber as mensagens.



Enviando mensagens para o servidor



Servidor recebendo as mensagens

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Encerrando o cliente

Texto

Descrição gerada automaticamente



O servidor não recebe as mensagens da instancia 2



Porém recebe da instância 1



Tentando criar uma segunda instância de servidor, ela não é criada

**Exercício 4**

**Função Lambda**

Trata-se de uma função sem declaração, assim diminuindo a quantidade de quantidade de código feito para funções, pois não necessita de nome, retorno ou modificador de acesso. Ela é declarada no mesmo lugar de uso.

(argumento) -> (corpo), podendo colocar vários comandos na mesma.

Exemplo de thread:

Runnable r = new Runnable() {

     public void run() {

          System.out.println("Thread com classer interna!");

     }

};

new Thread(r).start();

Com função lambda:

Runnable r = () -> System.out.println("Thread com função lambda!");

new Thread(r).start();

Nota-se que a quantidade de código e linhas necessárias é bem menor

Elas são muito utilizadas para escrever threads conjuntamente com a interface runnable, ou Collections com o método forEach e por fim para ordenação de listas com a interface Comparator. O desenvolvimento de APIs também é um grande uso das funções Lambda.

Threads em Java

As threads auxiliam em executar diversos processos ao mesmo tempo, assim reduzindo o tempo necessário para rodar um conjunto de instruções. Existe uma interface padrão para thread chamada Runnable, já citada acima e pode ser aplicada como runnable, caso uma classe estenda a outra. É uma ferramenta que pode ser utilizada mas não abusada, visto que se um depender do outro, o processo pode ficar ainda mais lento.

Por exemplo, em um app de banco, onde cada thread pode ser um usuário e o próprio banco.

class Consumer extends Thread {

    // Referência para buffer compartilhado

    Buffer buffer;

    // Construtor

    public Consumer(Buffer buffer, String c) {

        super(c);

        this.buffer = buffer;

    }

    // Método redefinido que executa a função da thread

    @Override

    public void run() {

        try {

            // Tenta consumir um número inteiro

            while (true) {

                buffer.get();

                // Dorme 200 ms

                Thread.sleep(200);

                // Ou comente a linha anterior e

                // descomente a linha de baixo

                // e dê a chance para outra thread

                // Thread.yield();

            }

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}

class Producer extends Thread {

    // Referência para buffer compartilhado

    Buffer buffer;

    // Construtor

    public Producer(Buffer buffer, String p) {

        super(p); // chama o construtor de Thread e passa o nome do parâmetro

        this.buffer = buffer;

    }

    // Método redefinido que executa a função da thread

    @Override

    public void run() {

        Random random = new Random();

        try {

            // Tenta produzir um número inteiro aleatório

            while (true) {

                buffer.put(random.nextInt(1000));

                // Dorme 200 ms

                Thread.sleep(200);

                // Ou comente a linha anterior e

                // descomente a linha de baixo

                // e dê a chance para outra thread

                // Thread.yield();

            }

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}

Este exemplo foi de um projeto de Sistemas Operacionais, onde já vimos o conceito e aplicamos. Para tal, podemos executar várias operações de diferentes clientes ao mesmo tempo.