

Expressões Lambda

Resultado: 8

Resultado da comparação: 11

Números pares: [2, 4, 6, 8, 10]

Funções Anônimas

Resultado:8

Resultado: 8

SimpleThreadsExample

Missao: Gerar um bilhao de numeros aleatorios!

- Criando 1 Thread(s) para isso!

- Iniciando Thread Principal...

- Encerrando o processamento...

Missao cumprida em 35,90 segundos!

MultipleThreadsExample

Missao: Gerar um bilhao de numeros aleatorios usando Threads!

-> Thread iniciada (main)...

-> Criando +1 Thread(s) para isso:

----> Thread criada (Thread-0)!

-> Thread finalizada (main) em 0,00s!

----> Iniciando Thread (Thread-0)...

----> Encerrando Thread (Thread-0)!

----> Missao da Thread (Thread-0) cumprida em 36,90s!

Missao: Gerar um bilhao de numeros aleatorios usando Threads!

-> Thread iniciada (main)...

-> Criando +2 Thread(s) para isso:

----> Thread criada (Thread-0)!

----> Thread criada (Thread-1)!

----> Iniciando Thread (Thread-0)...

----> Iniciando Thread (Thread-1)...

-> Thread finalizada (main) em 0,00s!

----> Encerrando Thread (Thread-0)!

----> Missao da Thread (Thread-0) cumprida em 19,38s!

Missao: Gerar um bilhao de numeros aleatorios usando Threads!

-> Thread iniciada (main)...

-> Criando +3 Thread(s) para isso:
----> Thread criada (Thread-0)!
----> Thread criada (Thread-1)!
----> Iniciando Thread (Thread-0)...
----> Iniciando Thread (Thread-1)...
----> Thread criada (Thread-2)!
-> Thread finalizada (main) em 0,00s!
----> Iniciando Thread (Thread-2)...
----> Encerrando Thread (Thread-2)!
----> Missao da Thread (Thread-2) cumprida em 12,40s!

----> Encerrando Thread (Thread-0)!
----> Missao da Thread (Thread-0) cumprida em 12,42s!

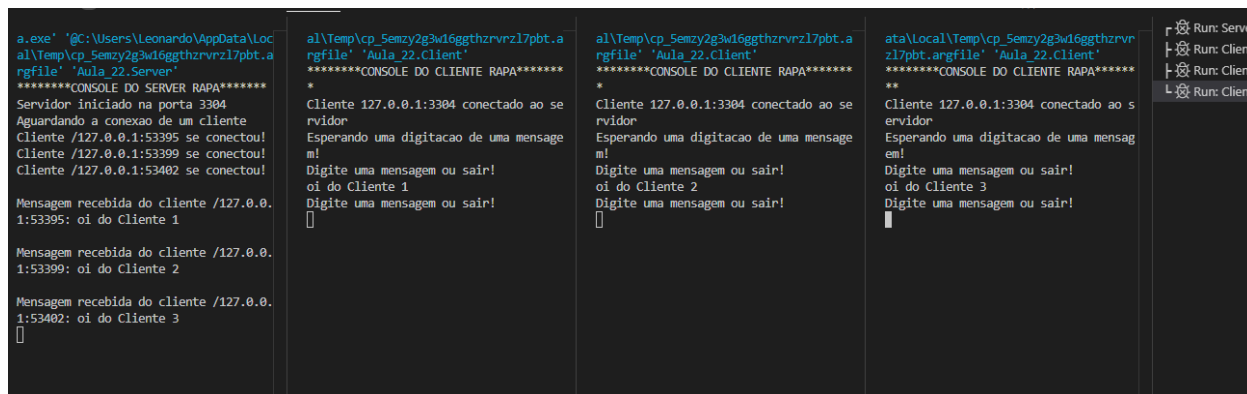
----> Encerrando Thread (Thread-1)!
----> Missao da Thread (Thread-1) cumprida em 12,46s!

4. O que acontece se na linha de comando digitada via console: java MultipleTreadExample.java N, o valor de N for aumentando? Há um limite pragmático para N? Justifique!

Quanto maior for o N, maior será o número de threads, e com isso menor será a carga de cada thread individual, considerando que os 1 bilhão de números serão divididos igualmente entre elas. Porém, existe um limite para o número de threads que possam ser criadas, isso se dá por uma multitude de fatores, sendo eles:

- Falta de memória, cada thread consome uma quantidade de recursos e memória do sistema.
- Overhead do Sistema Operacional, caso o número de threads for muito grande, o sistema pode gastar muito tempo gerenciando-as. Assim, existe um limite de threads que o sistema pode gerenciar eficientemente.

5. Registrar o funcionamento dos códigos dos exemplos 12, 13 e 14.



The image shows four terminal windows side-by-side, illustrating the execution of a multi-client server and its clients. The first window on the left shows the server's output, which includes messages like 'Servidor iniciado na porta 3304', 'Aguardando a conexao de um cliente', and 'Cliente /127.0.0.1:53395 se conectou!'. The subsequent three windows show the output of three separate client programs, each displaying '*****CONSOLE DO CLIENTE RAPA*****' and 'Cliente 127.0.0.1:3304 conectado ao servidor'. On the far right, there is a vertical toolbar with icons for running the server and clients, with the 'Run: Client' option being selected.

6. O que acontece se o número de clientes instanciados for elevado? Há um limite para a quantidade de cliente na aplicação do servidor fornecida? Justifique.

O código de multi cliente para servidor foi produzido sem um limite de clientes definido, então um número elevado de clientes continuará funcionando, com a única limitação sendo quantas threads o hardware do servidor é capaz de suportar.

7- B. Executar somente a classe Client.java e registrar o que acontece.

Quando a classe Client.java é inicializada, ela executa uma função start(), que tenta conectar o cliente ao servidor, mas já que a classe Server.java não foi inicializada, ela gera uma excessao e falha em iniciar o cliente.

7- C. Executar somente a classe Server.java e registrar o que acontece.

Quando a classe Server.java é inicializada, ela executa uma função start(), que gera o servidor e espera um cliente se conectar, mas já que a classe Client.java não foi inicializada, a classe Server.java fica esperando um cliente se conectar.

7- D. Sem encerrar a execução da classe Server.java, executar a primeira instância da classe Client.java e registrar o que acontece.

Nesse cenário, o servidor fica esperando um cliente se conectar. Logo após a inicialização da classe Client.java, o Console do Cliente avisa que o cliente se conectou ao servidor e agora é permitido enviar mensagens para o servidor.

7- E. Sem encerrar a execução da classe Server.java, executar a novamente a primeira instância da classe Client.java e registrar o que acontece.

Após a inicialização da classe Client.java, o Console do Cliente avisa que o cliente se conectou ao servidor e agora é permitido enviar mensagens para o servidor. A mensagem digitada depois é enviada ao servidor e exibida no console.

7- F. Sem encerrar a execução da classe Server.java, executar a segunda instância da classe Client.java e registrar o que acontece.

Nesse cenário, o servidor fica esperando um cliente se conectar. Logo após a inicialização da classe Client.java, o Console do Cliente avisa que o cliente se conectou ao servidor e agora é permitido enviar mensagens para o servidor.

Logo após a inicialização da segunda instancia da classe Client.java, o Console do Cliente avisa novamente que o cliente se conectou ao servidor. Ainda nesse caso, o Console do Servidor avisa que os dois clientes se conectaram e permite ambos clientes a mandarem mensagens para o servidor.

7- G. Sem encerrar a execução da classe Server.java, executar a terceira instância da classe Client.java e registrar o que acontece.

Nesse cenário, o servidor fica esperando um cliente se conectar. Logo após a inicialização da classe Client.java, o Console do Cliente avisa que o cliente se conectou ao servidor e agora é permitido enviar mensagens para o servidor.

Logo após a inicialização da segunda instância da classe Client.java, o Console do Cliente avisa novamente que o cliente se conectou ao servidor. Ainda nesse caso, o Console

do Servidor avisa que os dois clientes se conectaram e permite ambos clientes mandarem mensagens para o servidor.

Logo após a inicialização da terceira instância da classe Client.java, o Console do Cliente avisa novamente que o cliente se conectou ao servidor. Ainda nesse caso, o Console do Servidor avisa que os três clientes se conectaram e permite ambos clientes mandarem mensagens para o servidor.

7- H. Sem encerrar a execução da classe Server.java, executar a uma nova rodada de mensagens nas instâncias 1, 2 e 3 da classe Client.java e registrar o que acontece.

Nesse cenário, o servidor avisa que as mensagens foram recebidas, indicando a mensagem e o cliente que a enviou.

7- I. Encerrar as instâncias 1, 2 e 3 da classe Client.java e registrar o que acontece.

Nesse cenário, o Console do Cliente de cada instância avisa que o cliente foi finalizado.

7- J. Sem encerrar a execução da classe Server.java, executar uma nova instância da classe Server.java e registrar o que acontece.

Nesse cenário, o Console do Servidor da segunda instância avisa que não foi possível inicializar o segundo servidor, já que o endereço já está sendo utilizado no servidor da primeira instância da classe Server.java.