07

Mãos à obra: Enviando e recebendo comandos do servidor

Altere a classe `ClienteTarefas` para ela enviar e receber comandos do servidor, paralelamente:

1) Crie duas thread dentro do métodos `main` , logo após do `System.out.println("conexão estabelecida");`

Você pode pode implementar uma classe que é um `Runnable` diretamente no método através do recurso denominado *classe anônima* do Java:

```
Thread threadEnviaComando = new Thread(new Runnable() {

    @Override
    public void run() {
        //aqui vem mais
    }
});

Thread threadRecebeResposta = new Thread(new Runnable() {

    @Override
    public void run() {
        // aqui vem mais
    }
});
```

2) Dentro do `Runnable` para enviar comando coloque o código para ler os dados, usando o `PrintStream` para enviar e o Scanner para ler do seu teclado:

```
// dentro do método run da threadEnviaComando
try {
    System.out.println("Pode enviar comandos!");

    //saida baseado no OutputStream
    PrintStream saida = new PrintStream(socket.getOutputStream());

    //estabelecendo a leitura do teclado
    Scanner teclado = new Scanner(System.in);


    //aguardando "enter" do teclado
    while (teclado.hasNextLine()) {

        //pegando o que foi digitado no console
        String linha = teclado.nextLine();

        //se for vazio não vamos enviar nada para o servidor
        if (linha.trim().equals("")) {
            break;
        }
```

```
            //enviando para servidor
            saida.println(linha);
        }

        //fechando os recursos
        saida.close();
        teclado.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
```

O código faz nada mais do que esperar que você digite algo no console e, ao apertar `enter`, enviará o conteúdo digitado para o servidor.

3) Agora vamos focar na outra thread para receber os dados do servidor. No método `run` da `threadRecebeResposta`:

```
try {
    System.out.println("Recebendo dados do servidor");

    //leitura dos dados que vem do servidor
    Scanner respostaServidor = new Scanner(socket.getInputStream());

    //imprimindo resposta do servidor no console
    while (respostaServidor.hasNextLine()) {
        String linha = respostaServidor.nextLine();
        System.out.println(linha);
    }

    respostaServidor.close();
} catch (IOException e) {
    throw new RuntimeException(e);
}
```

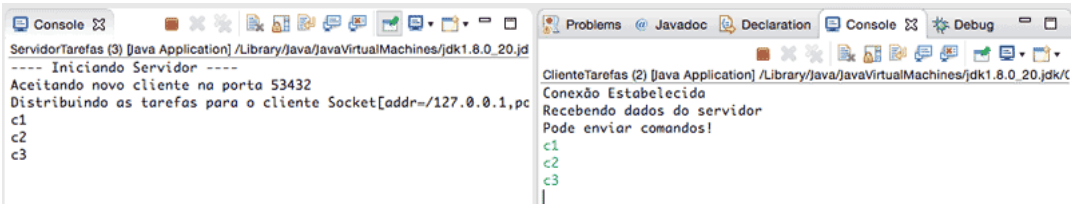
3) No final do método `main`, mas antes do `socket.close()` inicie as duas threads:

```
threadRecebeResposta.start();
threadEnviaComandos.start();
```

4) O `Socket` só pode ser fechado quando a thread que envia os comandos terminar, para implementar isso, "junte" a thread `MAIN` à thread `threadEnviaComandos`, através do método `join`:

```
threadEnviaComandos.join();
```

5) Teste o novo cliente e envie um comando (`c1`, `c2` etc), digitando diretamente no console do cliente! O servidor deve imprimir o comando.


Console
ServidorTarefas (3) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jd
---- Iniciando Servidor ----
Aceitando novo cliente na porta 53432
Distribuindo as tarefas para o cliente Socket[addr=/127.0.0.1,pc
c1
c2
c3
Problems
Javadoc
Declaration
Console
Debug
ClienteTarefas (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/C
Conexão Estabelecida
Recebendo dados do servidor
Pode enviar comandos!
c1
c2
c3