04

E agora? Como resolver?

Vimos que o uso de threads não é exclusividade do servidor, tanto isso é verdade que usamos threads para podermos enviar e receber dados do servidor ao mesmo tempo. Inclusive não usamos um pool de threads porque sempre trabalharemos com duas apenas.

Contudo, vemos o seguinte fragmento de código:

```
public class ClienteTarefas {

    public static void main(String[] args) throws Exception {

        Socket socket = new Socket("localhost", 12345);

        // código que cria as duas threads omitido

        threadRecebeResposta.start();
        threadEnviaComando.start();
        socket.close();
    }
}
```

Do jeito que está, a chamada `socket.close()` será chamada antes mesmo da thread `threadEnviaComando` terminar. Qual alteração você faria no código para que a instrução `socket.close()` seja executada apenas quando `threadEnviaComando` terminar sua execução?

Selecione uma alternativa

A Antes do `socket.close()` vamos chamar `Thread.sleep(millis)` com um valor muito alto de milli segundos.

B Vamos usar uma das duas threads criadas e chamar o método `join` , por exemplo: `threadEnviaComando.join()`

C Vamos transformar a Thread `main` em um *daemon-thread*.

D Simples, não vamos chamar `socket.close()` !