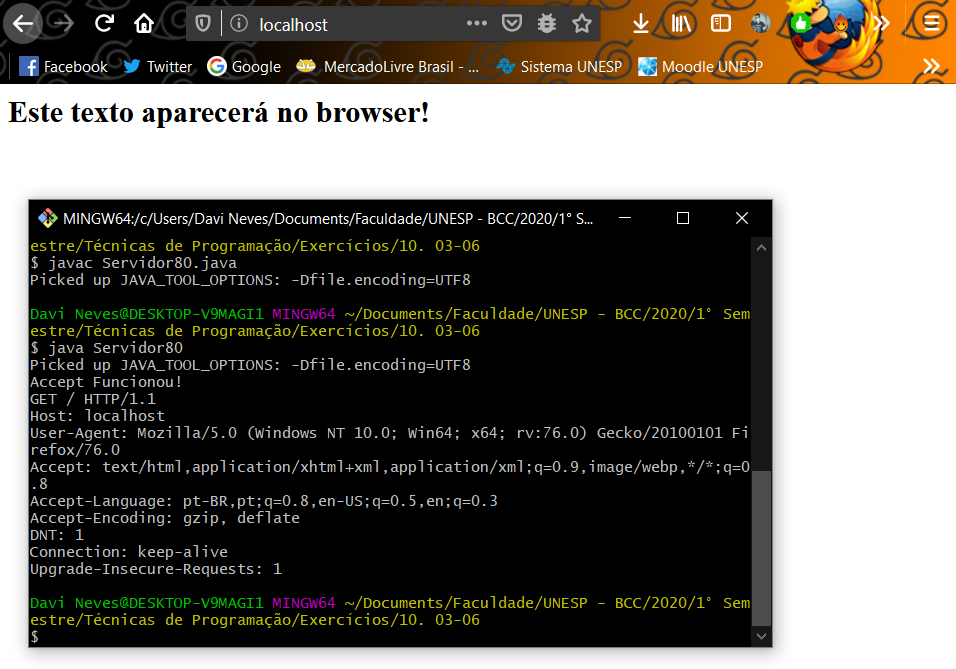
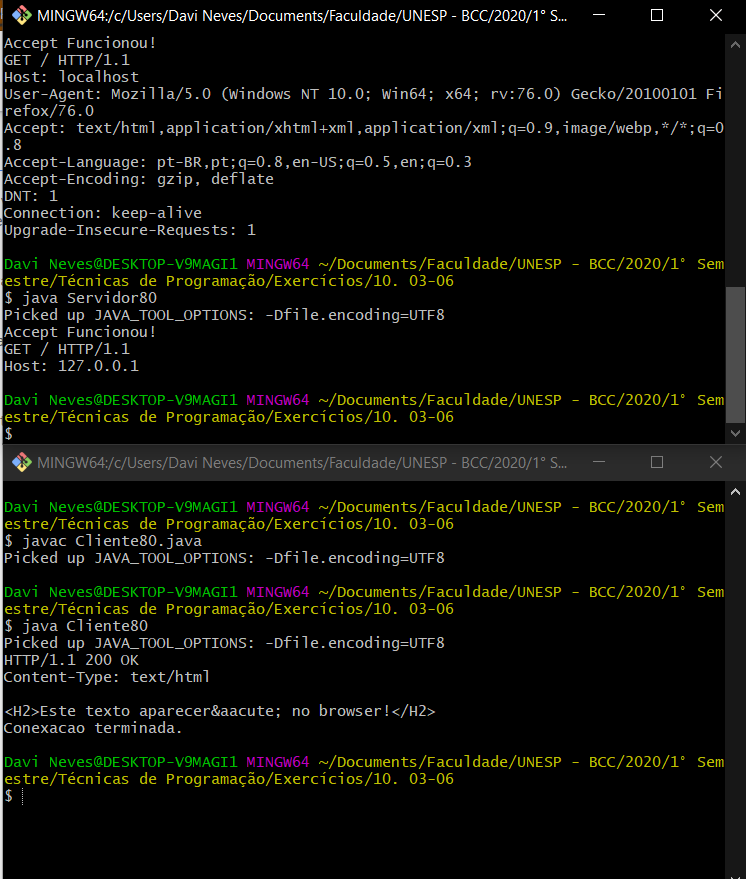
**Resolução – 8° Lista de Exercícios – Redes e Sockets**

**Grupo 1 de Programas (Servidor80 e Cliente80)**

**1)**

Ao conectar no endereço “localhost”, o seguinte texto é emitido no navegador pela linha 36, além de mostrar os dados de entrada da conexão entre o meu navegador com o Servidor80 (porta 80) na linha de comando. Esses dados, conforme a imagem abaixo, exprimem desde o cliente aceito pelo servior (neste caso, o Firefox) até a entrada de dados desse cliente como a encodificação e o idioma. Já os dados de saída são representados pela mensagem no navegador, vinda direta do servidor.

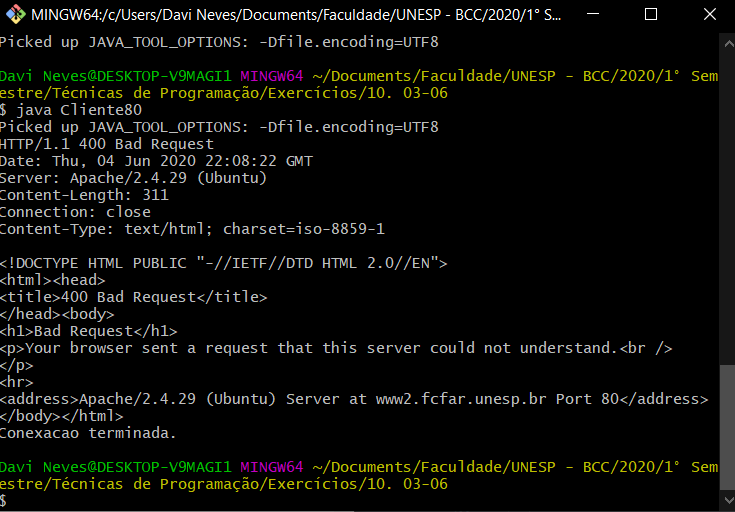


**2)** 

**3)**

Ocorrem os seguintes processos:

1. O cliente solicita a conexão no host (neste caso, o endereço “www.fc.unesp.br”) que é recusada (400 Bad Request).
2. Na linha 28 do código é exprimido a saída que está na primeira linha após a execução na linha de comando (ver imagem abaixo, após a encodificação em UTF-8).
3. Os dados mandados pelo host para o cliente (contidos na variável “in” após iniciar a conexão) são exibidos abaixo de “HTTP/1.1 400 Bad Request”. Neste caso, os dados vem de um servidor Apache e simbolizam o código em HTML (caso fosse exibido em um navegador, apareceria a mensagem “Your browser sent a request that this server could not understand.”).



**Grupo 2 de Programas (Servidor, Cliente, ClienteFrame)**

**1)**

Esse método escuta uma conexão e aceita quando for encontrada. Neste caso, o servidor aguarda até três conexões para prosseguir com o programa.

No exemplo da pergunta, o servidor irá esperar a conexão dos dois “Cliente” e de um “ClienteFrame” para prosseguir com o código. Além disso, esse método retorna um objeto do tipo Socket.

**2)**

A primeira instrução está relacionada com a entrada de dados de um “Cliente” no servidor, sendo que a segunda instrução está relacionado com a saída de dados desse servidor.

Dessa forma, a primeira instrução irá ler os dados de quaisquer um dos três clientes (aquele que enviar primeiro ao servidor) ao passo que a segunda irá enviar os dados de saída do servidor (neste caso, as mensagens vindas pelos clientes) para os três clientes (por conta do uso da Thread).

**3)**

Por meio da utilização de **Thread’s** que garantam o multiprocessamento, isto é, permite que as mensagens possam ser mostradas ao mesmo tempo, tendo cada cliente sendo manipulado no servidor por meio de uma Thread. Além disso, são usadas variáveis **estáticas** que mantém o seu valor durante a execução de uma nova Thread (por exemplo, um cliente é associado ao “cont” com valor 0 e o outro com 1).

**4)**

O “ClienteFrame” retrata uma aplicação GUI para o que o “Cliente” realiza. Além disso, textos com pontuação são mostrados corretamente (ver imagem abaixo da diferenças de textos exibidos entre “Cliente” e “ClienteFrame”).

