Trabalho 2 – Programação em Sockets com TCP

Regras Gerais:

O trabalho deve ser feito em grupos de até três alunos;

A detecção de cópias ZERA todos os trabalhos, independente de quem fez e quem copiou.

Trabalhos com erros de compilação receberão nota ZERO.

Entrega: Relatório em HTML e código fonte documentado.

Data de entrega: 27 de dezembro de 2017

O objetivo é construir, em Java, um Servidor Web multithreaded, com analisador de protocolo HTTP, que processará somente a primeira linha de requisição, para os métodos GET e HEAD. Para desenvolver esse trabalho siga os seguintes passos:

a) **INÍCIO**:
   a. Pegue o código do Servidor Web de exemplo apresentado ao fim deste documento, crie um projeto na IDE Java de sua preferência e faça-o funcionar.
   b. Para testar a aplicação, crie uma página web simples chamada index.html, coloque-a na mesma pasta onde está o projeto do servidor.
   c. Em seguida, após executar o servidor, abra o browser e digite: 127.0.0.1:6789/index.html (caso não use a porta 6789, coloque a porta correta).

b) **LOOP**: coloque um loop infinito no código de forma que ele possa atender a diversas requisições.

c) **BAD REQUEST**: o código inicial não gera a resposta BAD REQUEST para o browser, essa é sua próxima tarefa. Para testar, simule uma requisição com erro de sintaxe com o telnet:

```
telnet 127.0.0.1 6789
BLABLA index.html http/1.0
```

d) **FILE NOT FOUND**: originalmente não há tratamento de *file not found*. Você deve implementá-lo através de um TRY – CATCH. Para testar, abra o browser e digite: 127.0.0.1:6789/inxxx.htm. Lembre-se de mandar para o browser a resposta padrão HTTP para file not found.

e) LOGS: imprima no console todas as ações que acontecem no servidor, por exemplo: start do servidor; atendimento de requisição HTTP; inicialização de uma Thread; etc...

f) ANALISADOR DE PROTOCOLO HTTP: o código inicial somente lê a primeira linha da requisição do browser. Você deve ler todas as linhas e imprimir no console todas as mensagens HTTP request e response do seu Servidor Web.

g) HEAD: implemente também o método HEAD;

h) THREAD: para cada novo pedido de conexão, crie uma nova Thread para o atendimento.

i) TESTES: os testes deverão ser feitos usando como clientes a ferramenta telnet e um navegador. O servidor deve ser capaz de atender simultaneamente diversas requisições.

j) **RELATÓRIO**: o relatório deve ser feito em HTML. Crie uma página inicial do trabalho, que ficará hospedada no próprio servidor que você criar. Coloque links para: uma figura .jpg; um .gif; .um txt; e um .html com o código comentado. O relatório deve conter: introdução, descrição do código, descrição de testes e conclusão.

Código de exemplo:

```
import java.io.*;
import java.net.*;
import java.util.*;

class WebServer {

    public static void main(String args[]) throws Exception {
        String requestMessageLine;
        String fileName;
        ServerSocket listenSocket = new ServerSocket(6789);

        System.out.println("Web server listening to port 6789...");

        Socket connectionSocket = listenSocket.accept();
        System.out.println("Starting to proccess request...");
        BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
        DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
        requestMessageLine = inFromClient.readLine();
        StringTokenizer tokenizedLine = new StringTokenizer(requestMessageLine);

        if (tokenizedLine.nextToken().equals("GET")) {
            fileName = tokenizedLine.nextToken();
            if (fileName.startsWith("/") == true) {
                fileName = fileName.substring(1);
            }
            System.out.println("Request filename= "+fileName);
            File file = new File(fileName);
            int numOfBytes = (int) file.length();
            FileInputStream inFile = new FileInputStream(fileName);
            byte[] fileInBytes = new byte[numOfBytes];
            inFile.read(fileInBytes);
            outToClient.writeBytes("HTTP/1.0 200 Document Follows\r\n");
            if (fileName.endsWith("jpg")) {
                outToClient.writeBytes("Content-Type: image/jpeg\r\n");
            }
            if (fileName.endsWith("gif")) {
                outToClient.writeBytes("Content-Type: image/gif\r\n");
            }

            outToClient.writeBytes("Content-length: " + numOfBytes + "\r\n");
            outToClient.writeBytes("\r\n"); // fim de cabeçalho
            outToClient.write(fileInBytes, 0, numOfBytes);
            connectionSocket.close();
        } else {
            System.out.println("Bad Request Message");
        }
    }
}
```