Metro State University of Denver, Department of Mathematics and Computer Science
CS 3700-001: Computer Networks
Spring 2013
Dr. Weiying Zhu
Homework 3, Due Date: 10:00am 02/14/2013
Submission: Upload all source code (.java) files to Blackboard

Write a client process and a server process to implement the following simplified HTTP protocol based on TCP service. Please make sure your program supports multiple clients. The webpage file CS3700.htm is provided. You may choose a port like 5678 and hard code it in both client and server programs.
(Hints: for testing, you may want to put client process and server process at two different directories such that the CS3700.htm saved by the client process won't conflict with the one that the server process reads from.)

- The ending signature of the entity body in the HTTP response message for the case "OK 200":
  - When the HTTP Server process sends a HTTP response message out, it pads four continuous blank lines, i.e., "\r\n\r\n\r\n\r\n", to the end of the .htm file as the ending signature of the entity body.
  - When the HTTP Client process read the HTTP response message line by line, it counts the number of **continuous** lines that are empty strings "" (NOT a null string). Once such number reaches 4, the entity body has been fully received.
  - It is assumed that the .htm file does not include such ending signature (in the real practice, length of the entity body may be included in the head line and used to determine the end of an entity body).
- HTTP Client Process:
  1. Display messages on the standard output to ask the user to input the Host Name (it could be an ip-address) of your HTTP server.
  2. Buildup the TCP connection to your HTTP server with the Host Name input by User at the given port. Catch the exception, terminate the process, and display error messages on the standard output if any.
  3. Display messages on the standard output to ask the user to input the HTTP method type, name of the htm file requested, HTTP Version, and User-Agent, respectively. (hint: all inputs can be strings.)
  4. Use the above inputs from user to construct ONE HTTP request message and send it to the HTTP server process over the TCP connection. Your HTTP request message only needs to include the following lines. (The correctness of the format will be checked by the instructor):
     The request line (hint: the URL should include a '/' in front of the htm file name)
     The header line for the field "Host:"
     The header line for the field "User-Agent:"
     \r\n
  5. Receive and interpret the HTTP response message from the HTTP Server process **line by line**, display the status line and header lines of the HTTP response message on the standard output, and save the data in the entity body to a .htm file to local directory if there is any. (Hint: (a) When one empty string "" (NOT a null string!) is read the FIRST TIME, it indicates the header lines are over and the entity body is going to start next line if the case is "OK 200".)
  6. Display a message on the standard output to ask the User whether to continue. If yes, repeat steps 3 through 6. Otherwise, close all i/o streams, TCP connection, and terminate the Client process.
- HTTP Server Process:
  1. Listen to the given port and wait for a connection request from a HTTP Client.
  2. Create a new thread for every incoming TCP connection request from a HTTP client.
  3. Read, display to the standard output, and interpret incoming HTTP request message line by line
     - If the method given in the request line is NOT "GET", it is a "400 Bad Request" case
     - If the file specified in the URL does not exit/cannot be open, it is a "404 Not Found" case
     - Otherwise, it is a "200 OK" case
  4. According to the case discovered above, construct ONE HTTP response message and send it to the HTTP client process over the TCP connection. Your HTTP response message only needs to include the following lines using the HTTP message format:
     The status line
     The header line for the field "Date:"
     The header line for the field "Server: ", you may use any value of your choice
     \r\n
     Data read from the requested HTML file line by line … (hint: for the 200 OK case only)
     \r\n\r\n\r\n\r\n
  5. Repeat Step 3 through 4 until a null is read.
  6. Close all i/o streams and the TCP socket for THIS Client, and terminate the thread for THIS client.