Homework 3, Due Date*: 10:00am 02/24/2015, Cutoff Date: 10:00am 02/27/2015**
Submission: Upload all source code (.java) files to Blackboard, and java programs (.class files) to EC2 instances
***Late penalty will apply for past-due late submission; **Submission will NOT be accepted after the cutoff deadline**

Part I: Write a client program and a server program 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 both programs on the same computer, you may want to put client program and server program at two different directories. Do NOT hard code the file name "CS3700.htm"!)

- The ending signature of the entity body in the HTTP response message for the case "OK 200":
  - o When the HTTP Server program 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.
  - o When the HTTP Client program 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.
  - o 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 Program:
  1. Display messages on the standard output to ask the user to input the DNS name/ip of your HTTP server.
  2. Buildup the TCP connection to your HTTP server with the Host Name input by User at the given port, and display the **RTT of establishing this TCP connection** in millisecond (the difference between the moments right before and after creating the socket object). Catch the exception, terminate the program, and display error messages on the standard output if any.
  3. Display message**s** 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 (separately please!)**. (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 program 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 program **line by line**, display the **RTT** (File Transmission Time may be included) **of HTTP query** in millisecond (the difference between the moment right before HTTP request is sent and the moment right after HTTP response is received) as a single line (e.g., RTT = 1.089 ms), 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 program.
- HTTP Server Program:
  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
     - o If the method given in the request line is NOT "GET", it is a "400 Bad Request" case
     - o If the file specified in the URL does not exit/cannot be open, it is a "404 Not Found" case
     - o 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 program over the TCP connection. Your HTTP response message needs 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.

Part II: Test your programs with multiple clients using the Amazon EC2 Cloud.
(The instructor will start and use your instances and her local computer to test your programs and give credits for Parts I and II according to the test results. The latest time of your ***.class*** and ***.txt*** files on your EC2 instances is the submission time of part II.)
1. MAKE a directory "**HW03**" under your home directory on each of your EC2 instances.
2. UPLOAD your ***server*** program to "HW03" on your instance in *N. VA* and your ***client*** program to "HW03" on your instance in *Sydney*.

(continue on next page!!!)

3. TEST the server program together with the client program on a computer and, simultaneously, on your EC2 instance in *Sydney* to test all the possible cases. Here, you need to have one server and at least two clients.
4. SAVE a file named ***testResultsClient.txt*** on your EC2 instance in *Sydney*, which captures the outputs of your ***client*** program when you test it.

⚠ Please STOP your ec2 instances whenever you are NOT working on them. Please do NOT start or stop ec2 instances whose names do not contain your user name. Thank you!