**Project 2**: HTTP download using "Range"  
  
*Type of Project*: Only Individual  
*Deadline*: 2012-02-16, 11:59pm  
*Language*: Java  
*Points*: Max 20 points  
  
***Submission Guidelines***: Submit through nike.cs.uga.edu, as usual. Name the directory project as "**LastNameFirstName**-http\_downloader" (e.g., "PerdisciRoberto-http\_downloader"). Submit ONLY the source code. Name the source code file containing the main as "HttpDownloader.java" (use inner classes, if you need more than one class to develop your program). Copy "HttpDownloader.java" under the directory "**LastNameFirstName**-http\_downloader" and

$ submit LastNameFirstName-http\_downloader cs4760

NOTE: project submissions that do not follow the guidelines risk to be discarded without consideration (i.e., 0 points).  
  
***Project Description***: In this project, you are required to write a program that takes in input (on the command line) the URL of an object to be downloaded, and the number of connections through which different parts of the object will be retrieved using "Range:". The downloaded parts need to be re-stitched to compose the original file.  
  
For example (notice that the line below is an actual example of how your program must be launched)

$ java HttpDownloader http://www.cs.uga.edu/images/template/cs\_template\_r2\_c2.gif 5

will **spawn 5 threads**, each thread will open one TCP connection to www.cs.uga.edu on port 80, and retrieve a part of the .gif file. Each of the 5 parts must be of an approximately equal length. Finally, the program will put the parts together and write the output into a file called "cs\_template\_r2\_c2.gif". You should name the files containing the parts of downloaded content as "part\_i", where i is an index. In the example above, the program will output the parts into 5 different files called "part\_1", "part\_2", ..., "part\_5", along with the reconstructed "cs\_template\_r2\_c2.gif" file. DO NOT delete the "part\_i" fields after you are done recomposing the original file.  
**Save all downloaded files into the same directory from which the program is launched** (do not create any subdirs).  
  
To make sure your software downloads and correctly reassembles objects from the web, you can use *md5sum* to compare your result with the original file downloaded using a browser (or *wget* or *curl*), for example.  
  
NOTE: You don't need to worry about handling Server "errors" (e.g., redirections, unavailable Range option, etc.). I will only test your software on objects retrieved from websites that support the Range option, and for which no special error handling is required. Of course, make sure to use HTTP/1.1 and that your HTTP requests are correctly formatted, otherwise they will fail even the simpler tests.  
HINT: To divide the object to be retrieved into approximately equal parts, you first need to retrieve the length of the object without retrieving the object itself. One way to do this is using a HEAD request and parsing the Content-Length field in the response. An alternative (optional) way to do this is by using a GET request with a particular value in the Range field (I will leave it to you to figure this out, if you decide to use this second option).  
  
TESTING YOUR CODE  
You can use the following URLs to test your code:

http://www.cs.uga.edu/~perdisci/CSCI4760-S12/Project2-TestFiles/topnav-sport2\_r1\_c1.gif  
http://www.cs.uga.edu/~perdisci/CSCI4760-S12/Project2-TestFiles/Uga-VII.jpg  
http://www.cs.uga.edu/~perdisci/CSCI4760-S12/Project2-TestFiles/story\_hairydawg\_UgaVII.jpg

Make sure that when you recompose the output from the different parts, the md5sums match the following ones:

04e1f00315854f382d00311c599a05f8 story\_hairydawg\_UgaVII.jpg  
0592796fa4f06f806e5d979d7e016774 topnav-sport2\_r1\_c1.gif  
9dc5407cc368aaaa33c6cc2c284ab5c4 Uga-VII.jpg

I suggest you to also test your code on other URLs chosen by yourself, and try to determine if there are any websites that support the Range option but cause problems to your code.  
  
MAILING LIST: If you have questions about the projects, the best place to ask is the course mailing list.  
  
PROJECT EVALUATION  
I will run your program on 5 different input URLs and number of requested parts. For each of the 5 runs, you will be assigned 4 points if the md5sum of the final recomposed file matches the md5sum of the original object. Therefore, you will get max points if all the md5s match correctly.  
Notice that I will also verify that the "part\_i" files exist and their size respects the criteria outlined above, otherwise you may be penalized.

TESTING

curl –o web.out http://www…jpg

md5 web.out