**Phase 2 Submission Protocol**

This document follows and makes reference to the Phase 2 Requirements document. For submission and assessment purposes, This phase will be described by 10 tasks and each of these tasks has a separate entry under the Phase 1 subdirectory of the Projects subdirectory under the Assignments tab folder in the Blackboard system.

**SE.P2.Task#1) Input File.** The program must have an input text file containing the following set of data that will be processed by this phase:

Your input file will be a listing of URLs each ending in a different ending from the previous paragraph: minimally,

one url ending in .htm or .html

one url ending in .txt

one url ending in .jpeg, .jpg or .gif

//First Note: some sites will not return these files due to copyright so choose URLs that will allow //these files to be returned to your project.

//Second Note: Some .gif file have sophisticated encoded animations that may not be compatible //with the Java methods. If this is the case for your .gif file, then you will need to obtain such a .gif //file byte by byte using bytestreams. These are explained briefly in the corresponding //Requirements document.

one url ending in .pdf

one url ending in .docx

Upload this file to Blackboard system for **SE.P2.Task#1**.

**SE.P2.Task#2) Command Line Processing.** Your program parses flags and arguments from command line. Minimally, the name of an input file and the name of an output file and the directory path where the input and output files are found. Collect all the .java files necessary for this task into a single temporary directory on your computer system. Compress/zip the directory all as one file and upload it to the Blackboard system for **SE.P2.Task#2.**

**SE.P2.Task#3) File Processing.** After the project code obtains the files from the URLs provided for SE.P2.Task#1, this phase is supposed to process these files as follows (as explained in the corresponding Requirements document):

If the URL is a .html or .htm or .txt file, then read all the lines of the file and save it to a separate file on your system with the Same Name as the URL file name (i.e. last portion of actual URL). Then, output the number of lines read in and the name of the file to the output file (again this is referred to by the -o parameter on the command line). If the URL is an image file, .jpeg or .jpg or .gif, then save the image file to your computer with the Same Name as the URL file name. Then, output the name of the file to the output file. If the URL is .pdf or .docx then save the file to your computer with the Same Name as the file name found in the URL and output the name of the file to the output file. In the case of .pdf and/or .docx you will copy byte by byte instead of line by line because parts of .pdf files are stored in binary encodings and not ascii. See <https://docs.oracle.com/javase/tutorial/essential/io/bytestreams.html> for a template of Java code that will help accomplish this. You will need to integrate this kind of code with the code discussed in class.

Collect all the .java files necessary for this task into a single temporary directory on your computer system. Compress/zip the directory all as one file and upload it to the Blackboard system for **SE.P2.Task#3.**

**SE.P2.Task#4) Output Files (part 1): Actual files obtained.** The previous task explored obtaining data files found on various websites based on their file types. Collect all the “output” files produced for this task into a single temporary directory on your computer system. Compress/zip the directory all as one file and upload it to the Blackboard system for **SE.P2.Task#4.**

**SE.P2.Task#5) Output Files (part 2): URL info about files obtained.** For each URL you are processing, obtain URLInfo (as we discussed in class; if find other info that you can obtain, please do tell me about them). Output the name of the URL and all of its info about the URL to the output file (whose name will be the -o parameter on the command line you pass to java; again. –i provides the name of the input file, which contains a list of all of the URLs tested). Included in the processing (output), is determining the number of bytes in the file pointed to by the URL. See <https://www.mkyong.com/java/how-to-get-file-size-in-java/> for code samples on how to accomplish this. Collect all the output files produced for this task into a single temporary directory on your computer system. Compress/zip the directory all as one file and upload it to the Blackboard system for **SE.P2.Task#5.**

**SE.P2.Task#6) Coding Statistics (Part1).** How many classes in total did you code in your submitted project? How many methods in total did you code in your submitted project? This report should be typed into a text (.txt) file which you will upload to Blackboard system for **SE.P3.Task#6**.

**SE.P2.Task#7) Coding Statistics (Part2).** Estimate how many hours of coding you put into each of the three phases. This report should be typed into a text (.txt) file which you will upload to Blackboard system for **SE.P3.Task#7**.

**SE.P2.Task#8) Overall Phase Assessment.** Read over the requirements document for this phase. Type into a text (.txt) file a report on what was accomplished and what was not accomplished in your project submission (see SE.P2.Task#9 below). What functionality that you submitted works and which functionality in your code does not work. For the latter, what do you think is causing the problem? This report should be typed into a text (.txt) file which you will upload to Blackboard system for **SE.P2.Task#8**.

**SE.P2.Task#9) Complete Java Code**. Collect all the .java files necessary for your system developed so far into a single temporary directory on your computer system. Compress/zip the directory all as one file and upload it to the Blackboard system for **SE.P2.Task#9**. If you did this correctly and did not add any extraneous project or data files and just the .java files, then the compressed file to be uploaded should not be particularly large and you will be able to upload as is (and NOT as a shared or cloud file or via file link.)

**SE.P2.Task#10)** **Commenting the Java Code**. Comment your code appropriately and use meaningful names for classes, methods, variables and constants. You are expected to report on the number of lines of code using cloc found at <https://github.com/AlDanial/cloc/releases/tag/v1.92> This command must be run on the command line (cmd.exe). When you collect all .java files into the same temporary directory (see SE.P2.Task#9 above), then the following command (shown below) will report on the contents of each .java file. The report generated by cloc program will report on the number of actual java code lines, blank lines, comment lines for each of the .java files that is part of your project. The command to obtain the report data is as follows (assuming you are using the above version):

cloc-1.92.exe --by-file \*.java

This report should be copied into a text (.txt) file which you will upload to Blackboard system for **SE.P2.Task#10.**

Sincerely, Professor