C S 272/463 Introduction to Data Structures

Lab 1: Eclipse Tutorial and Java File Operations

In this Lab, the students will get familiar with Java file operations and learn how to use Eclipse IDE. If you did not take any prerequisite courses on Java, you may want to read `JavaOverview.pdf` on Canvas for an overview of Java. Please download `Lab1.zip` from Canvas that contains all files mentioned in this lab instruction.

1. **Task 1 (40 points)**
   Follow the steps in the Eclipse tutorial (`EclipseTutorial.pdf` on Canvas) to create a Java project named `Lab1` and do the following steps:

   a) (10 points) Create a Java class `Welcome` with a `main` method that prints out "`Welcome to Java`"

   b) (10 points) Add one line to the `main` method to print out the current system properties to screen. The function for getting the current system properties is `System.getProperties()`. Besides printing the results to screen, also use `java.io` to create a file named `eclipse_test.txt` to store the output results. One way is using `FileWriter` to write a string to a file.

   c) (20 points) In class `Welcome`, create a static method named `average()`. This method will read from the keyboard a line of integers that are separated by a space. You will parse the input to get the integers, compute the average of all integers, and write the output result to the `eclipse_test.txt`. Call `average()` method in `main` to test it. For this question, you can use `BufferedReader br = new BufferedReader(new InputStreamReader(System.in))` to read a line from the keyboard. Please check `Intro.pdf` to know how to parse a line. You may also need to use `Integer.parseInt(String s)`.

2. **Task 2 (60 points)**
   You are given a text file named `pg100_small.txt`. Create a Java file named `WordCount.java`. In this Java file:
   a) (25 points) Create a read function to read in the text file and count how many words (case-insensitive) there are in the text file.
   b) (25 points) Create a write function that receives in a word and returns all the unique words in the text file that follow right after this word. For example, if the file has the following content:

      “This is a test document. We will test reading and writing files in Java.”

      and the input word is “test”. The function will return “document”, and “reading” because these words follow right after “test” in the text file. Write all results to a file named `output.txt`. In this file, each line contains a word.
   c) (10 points) Create a main function to properly call the read and write functions that you created. Test the write function with the input word “best”.

Please check `MyTokenizer.java` for an example on how to tokenize text into an array of words. To run `MyTokenizer`, make sure that you have the file `test_doc.txt`. All these files are in the `Lab1.zip` on Canvas.

3. **Submission instructions**
   Submit through canvas a zip file consisting of `Welcome.java`, `eclipse_test.txt`, `WordCount.java`, and `output.txt`. Please do NOT submit the .class files.

4. **Grading criteria**
   - **a.** The score allocation is already put in the questions.
   - **b.** Please make sure that you test your code thoroughly by considering all possible test cases.
   - **c.** 5 points will be deducted if submitted files (including file types, file names, etc.) do not follow the instructions.
   - **d.** At least 20 points will be deducted if your code cannot be run on CS servers.