**Exercises:**

**File I/O Part 1**

Make a new Eclipse project and copy the enable1 word list from the file-io project to the top-level folder

of your new project. Except for problem #3, you may assume that the enable1 word list is all in lower

case. Recall from the lecture that the usual best real-life approach is to break your file reading into two

pieces: one reusable method that takes a Stream<String> and a second method that reads the file and

passes the Stream to the first method. But, since we have not yet explained how to do this breakup with-

out repeating code, for these exercises just make a separate class for each problem, put everything in

main, and have main throw Exception. This is the approach shown in the lecture. A similar approach is

to make separate static methods for each of the problems and have the static methods as well as main

throw Exception.

Problems 1-5 are the main file-reading problems. Problems 6-8 are only if you want to experiment with

some of the less important topics of the lecture.

**1.** Print the first 10-letter word found in the file.

**2.** Print the first 8-letter word that contains “a”, “b”, and “c”.

**3.** Repeat the previous problem, but handle the possibility of mixed-case words in the file. Hint: do

something shorter than merely modifying your filter tests to include “A”, “B”, and “C”.

**4.** Print the longest English word that contains neither “a” nor “e”.

**5.** Print the shortest English word that contains a “q”.

**6.** Make a file called “twitter-words.txt” that contains all words from the enable1 list that contain

“wow” or “cool”. The words should be sorted, in uppercase, and have an exclamation point at the

end. (E.g., “COOLER!”).

**7.** Print out the number of files in your Eclipse project. Folders count as files.

**8.** Create a file containing 17 random doubles between 0 and 100, each with exactly three digits after

the decimal point. Note that the file will not actually be written until you close the PrintWriter. So,

although you probably have mostly been ignoring exceptions, for this problem you might as well

declare the PrintStream using the try-with-resources approach shown in the lecture. Not only will

this handle the exceptions explicitly, but it will also automatically close the PrintStream at the end.