CSC171 — Project 1

WordWrap. Due: Thursday, September 23rd by 1159PM EDT.

This project will give you practice with looping, conditionals, strings. You will implement a program to perform "word wrapping" on unstructured text. Word wrapping is the process of converting very long lines and very short lines into columns of approximately uniform width. This is effectively what happens any time you resize a window containing textual data.

For this project, you are asked to implement a very specific form of word wrapping. You will not hyphenate or break any words. Instead, you will simply accumulate tokens (words) into a buffer and print the buffer whenever adding the next token would cause it to grow past the allowed length. The buffer can be maintained by careful use of `String` variables and the concatenation operator. No arrays or lists are required. Very long tokens (longer than the maximum width) should be printed on a line by themselves. Redundant whitespace between words (e.g., extra spaces or tabs) should be removed. Blank lines (or lines containing only whitespace) should be preserved (as completely blank lines). The user will specify the maximum width as the first line of input, which will be a positive integer. The user will specify the end of input by typing "Stop." alone on a line. Your program will then print the entire word wrapped data. The last page of this document includes an example input/output test case. That's all there is to it!

What You Should Know

We are expecting this project to involve while loops, if statements, and Strings. The only string operations you will need are concatenation, `length()`, and perhaps `substring()`. You will need `Scanner` operations such as `hasNext()`, `hasNextInt()`, `next()`, `nextInt()`, and `nextLine()`. You are not expected to use any collections or libraries which we have not yet discussed either in class or in the zyBook. In fact, you are ***not allowed*** to use any collections or arrays for this assignment. Doing so will result in point deductions.

Collaboration Policy

You may work on this project with a partner; however, collaboration with others should be at the level of "ideas only". Please do not share your source code with anyone other than your partner. Students who wish to tackle this project as a duo must include a block comment in their program which details individual student contributions. Failure to do this will result in points docked for both students. **Further details on submission instructions and academic honesty are in the next two sections.**

Submission Instructions

You will need to create a zipfile containing your program (in a file named "WordWrap.java") as well as a readme file (either in plain text or pdf, please no doc or docx files, or rtf files, etc.). The readme file should describe the status of your program, a high level description of your algorithm, and discuss any known bugs. If your program doesn't work 100% correctly, the readme file will help us to grade your work. Submit the zip containing your source code and readme to blackboard by the deadline.

If you are working with a partner, only ONE of you should submit the project zipfile; however, the other partner must submit a text file indicating who they collaborated with.

Academic Honesty

This project can either be completed solo or as a pair; however, the University of Rochester Academic Honesty Policy still applies. Attempting to find complete solutions either online or from peers is strictly forbidden, and any instances of dishonest submissions will result in a zero on the project and a full letter grade deduction for the course. Subsequent violations will receive stronger penalties.

If you work with a partner, you must both be currently enrolled in CSC 171, and it is your responsibility to ensure that the work you submitted is completed honestly. "My partner did it" is not a valid excuse for submitting plagiarized work. Both partners must be able to describe and explain any portion of the submitted work at any time.

Example Input

```
30
This is a an example of the WordWrap project.
Notice how
different
lines
are combined into a single "paragraph".

Notice also how      spaces are reduced,
but blank lines remain.

Lastly, when there are superlongtokensthatarelongerthanmax
notice how they’re on a single line.

Your program will read until it encounters Stop.

But only stops if Stop. is on a single line.
Stop.
```

Example Output

```
This is a an example of the
WordWrap project. Notice how
different lines are combined
into a single "paragraph".

Notice also how spaces are
reduced, but blank lines
remain.

Lastly, when there are
superlongtokensthatarelongerthanmax
notice how they’re on a single
line.

Your program will read until
it encounters Stop.

But only stops if Stop. is on
a single line.
```