Assignment 1

Due date

- 11.59 PM EST, Feb 6th

Git url

- https://classroom.github.com/a/SP_ckZmw

Submit your code as per the provided instructions.

Updates

-

Assignment Goal

A simple Java program.

Team Work

- No team work is allowed. Work individually. You cannot discuss the assignment with ANYONE other than the instructor and TA.

Programming Language

You are required to program using Java.

Compiling and Running Commands

- Compilation: Your code should compile on remote.cs.binghamton.edu with the following command: ant -buildfile wordPlay/src/build.xml all
-
- Running the code: Your code should run on remote.cs.binghamton.edu with the following command: ant -buildfile wordPlay/src/build.xml run -Darg0="student_coursePrefs.txt" -Darg1="courseInfo.txt" -Darg2="registration_results.txt"

Policy on sharing of code

EVERY line of code that you submit in this assignment should be written by you. Do NOT show your code to any other student. Do not copy any code from any online source. Code for File I/O or String operations, if found online, should be clearly cited, and you cannot use more than 5 lines of such online code.

Code downloaded in its entirety from an online repository of code (GitHub, BitBucket, etc.) and submitted as student's own work, even if citied, is considered plagiarism.

Code snippets, for File I/O, if used from an online source should be cited by mentioning it in the README.txt and also in the documentation of every source file in which that code appears.

Post to the piazza if you have any questions about the requirements. Do NOT post your code to the piazza asking for help with debugging.

Project Description

Assignment Goal: Develop a program, using Java, to reverse the words constituting sentences in a file and also to calculate certain metrics.

- An input file contains sentences. Each sentence contains words delimited by <space> character. Each sentence terminates with a period.
- The words in each sentence should be reversed.
- The sentences with the words reversed should be written to an output file.

Note: words in the input file are made of characters [a-zA-Z0-9] Note: The order of words in the sentence should not be changed - just each word should be reversed.
Example ***A good student => A doog tneduts***
Note: The order of sentences should not be changed. Instead, the words in each sentence need to be reversed (see input/output example for better understanding).

METRICS

The following metrics should be calculated and written to a metrics file, the name of which provided via commandline.

- AVG_NUM_WORDS_PER_SENTENCE - Average number of words per sentence. Round to 2 decimal places.
  Format: **AVG_NUM_WORDS_PER_SENTENCE = <value>**
- AVG_NUM_CHARS_PER_SENTENCE - Average number of characters per sentence (include spaces, including period, excluding newline characters).

Round to 2 decimal places.
Format: **AVG_NUM_CHARS_PER_SENTENCE = <value>**

- MAX_FREQ_WORD - Most used word in the file (max frequency). If there is contention between multiple words, then any one of them can be selected.
  Format: **MAX_FREQ = <value>**
- LONGEST_WORD - Word with the most number of characters. If there is contention between multiple words, then any one of them can be selected.
  Format: **LONGEST_WORD = <value>**

The following rules MUST be followed.

1. The input file should be processed one sentence at a time.
2. The program should not read in all the lines and store it in a data structure.
3. You should implement your own function for reversing a string.

INPUT FORMAT

Your program should accept three filenames from the commandline - *input.txt*, *putput.txt*, and *metrics.txt*. Note that *input.txt* will be well formatted.

EXAMPLES

input.txt

```
A new student has registered for design patterns in the spring of 2020.
During the semester the students
are going to learn good design principles and design guidelines to be followed when developing applications.
All programming assignments are to be done in Java.
```

output.txt

```
A wen tneduts sah deretsiger rof ngised snrettap ni eht gnirps fo 0202.
gniruD eht retsemes eht stneduts
era gniog ot nrael doog ngised selpicnirp dna ngised senilediug ot eb dewollof nehw gnipoleved snoitacilppa.
llA gnimmargorp stnemngissa era ot eb enod ni avaJ.
```

metrics.txt

```
AVG_NUMBER_WORDS_PER_SENTENCE = 14.33
AVG_NUM_CHARS_PER_SENTENCE = 87.66
MAX_FREQ_WORD = design
LONGEST_WORD = applications
```

NOTES ON GRADING

- Class participation points will be given to students who piazza with good questions seeking clarification on the assignment, and also to those who respond and participate to help make the discussion interesting and informative.

Sample Input Files sent by students in this course

Please check piazza.

-

Clarifications based on student questions

Compiling and Running Java code

- Your submission must include a readme in markdown format with the name **README.md**.
- Your README.md file should have the following information:
  - instructions on how to compile the code
  - instructions on how to run the code
  - justification for the choice of data structures (in terms of time and/or space complexity).
  - citations for external material utilized.
- You should have the following directory structure (replace username with your github username).
- ./csx42-spring-2020-assign1-username
- ./csx42-spring-2020-assign1-username/wordPlay
- ./csx42-spring-2020-assign1-username/wordPlay/src
- ./csx42-spring-2020-assign1-username/wordPlay/src/wordPlay
- ./csx42-spring-2020-assign1-username/wordPlay/src/wordPlay/util
- ./csx42-spring-2020-assign1-username/wordPlay/src/wordPlay/util/FileDisplayInterface.java
- ./csx42-spring-2020-assign1-username/wordPlay/src/wordPlay/util/FileProcessor.java

- ./csx42-spring-2020-assign1-username/wordPlay/src/wordPlay/util/Results.java
- ./csx42-spring-2020-assign1-username/wordPlay/src/wordPlay/util/StdoutDisplayInterface.java
- ./csx42-spring-2020-assign1-username/wordPlay/src/wordPlay/driver
- ./csx42-spring-2020-assign1-username/wordPlay/src/wordPlay/driver/Driver.java
- ./csx42-spring-2020-assign1-username/wordPlay/src/build.xml
- /README.md
- [Other Java files you may need]
-
-

Code Organization

- Your directory structure should be EXACTLY as given in the code template
  - Use the command on linux/unix to create an archive: *tar -cvzf csx42-spring-2020-assign1-username.tar.gz csx42-spring-2020-assign1-username/*.
  -
  - Use the command on linux/unix to extract the file: *tar -zxvf csx42-spring-2020-assign1-username.tar.gz*.

Submission

- Make sure all class files, object files (.o files), executables, and backup files are deleted before creating a zip or tarball. To create a tarball, you need to "tar" and then "gzip" your top level directory. Create a tarball of the directory csx42-spring-2020-assign1-username. We should be able to compile and execute your code using the commands listed above.
- Instructions to create a tarball
  - Make sure you are one level above the directory csx42-spring-2020-assign1-username.
  - tar -cvzf csx42-spring-2020-assign1-username.tar csx42-spring-2020-assign1-username/
  - gzip csx42-spring-2020-assign1-username.tar
- Upload your assignment to Blackboard, assignment-1.

General Requirements

- Start early and avoid panic during the last couple of days.

- Separate out code appropriately into methods, one for each purpose.
- You should document your code. The comments should not exceed 72 columns in width. Use javadoc style comments if you are coding in Java. Include javadoc style documentation. It is acceptable for this assignment to just have the return type described for each method's documentation.
- Do not use "import XYZ.*" in your code. Instead, import each required type individually.
- All objects, in Java, that may be needed for debugging purposes should have the "toString()" method defined. By default, just place a toString() in every class.
- Every class that has data members, should have corresponding accessors and mutators (unless the data member(s) is/are for use just within the method.).

Design Requirements

Late Submissions

- The policy for late submissions is that you will lose 10% of the grade for each day that your submission is delayed. **There is NO difference in penalty for assignments that are submitted 1 second late or 23 hours late** .

Grading Guidelines

Grading guidelines have been posted here.