Shingling Text Files

Overview

This project requires you to implement one of the common first steps in data science applications for comparing text files (TurnItIn, for example). You'll be reading a text file and computing its k-shingles (character shingles of length *k*).

Detailed Instructions

- Create a project named **Project1.** I highly recommend enabling Git on your local machine. I will provide follow up instructions about integrating Git and GitHub for the project.
- Navigate to https://classroom.github.com/a/6fLLSEw0 and accept the assignment. A private repository will be created for you on GitHub. In this case you'll just see two .txt files that you need to download and place in your project.

- NOTE: If you are working in IntelliJ and you want to place the text files in the same folder with your Java source code, you'll open the file as "src/wfu.txt" in your code. Otherwise, you'll need to place the data file in the parent folder of your "src" folder.

- One of the requirements for this project is that your program must read two values from the "command line". Since you are working in an IDE you have to set up these values indirectly. In IntelliJ you do this by choosing **Run → Edit Configurations** and filling the line in the form labeled "Program arguments:" Type the following in the form (without the quotes) "`5 src/wfu.txt`". This will cause args[0] in your program code to take on the value "5" and args[1] in your program code to take on the value "src/wfu.txt". These correspond to the value of *k* to be used in your program and the name of the text file to be read.

- Create your project and a class named **Main** that will contain your code. In your main() method you should check that you can use the values of arg[0] and arg[1] in your program.

- Create a Node class in your project. Each Node will contain a String of length *k* (a shingle) and a count of the number of times the shingle appears in the text file. The starting line of the class definition should read `class Node implements Comparable<Node>{`
  This class will need at least the following:
  - A String instance variable to store a shingle of length *k;* I called mine *data*
  - An int instance variable to keep count of how many times the shingle has appeared in the text; I called mine *count*
  - A constructor method that takes a String and an int as parameters and sets the instance fields.
  - *get* methods for the instance variables
  - a method to increment the count
  - a *compareTo* method to compare the String in a shingle to another String; returns 0 if the two Strings are equal; a positive value if the instance String comes after the other; a negative value if the instance String comes before the other, alphabetically.
  - a *toString* method that returns a String representation of the shingle and its count

- Write a method named testNode() to be called from main()when you are debugging your code. In this method you should create a Node(s) and call each of the methods in the Node class to make sure they work correctly.

- Set up a loop to read each line of the text file. Inside the loop you will need to:
  - *Clean* each line of the input. In detail that means you will:
    - Ignore lines of length 0
    - Convert all characters to lower case
    - Replace all characters other than a-z with a single space
    - Replace all sequences of multiple spaces by a single space
    - Strip white space from the beginning and end of the line
    - If the line is now empty, return an empty string
    - Else return the line with a single space added to the end
  - Identify all the shingles of length *k* in the line and insert them into an ArrayList of Nodes, along with the count of each shingle, maintaining the Nodes in alphabetical order based on the string in the Node. For partial credit you can use sequential search to find a shingle within the ArrayList but for full credit you must use binary search to determine if a shingle is already in the ArrayList or not, and, if not, where the shingle should be inserted to maintain alphabetical order. NOTE: You will be implementing a version of Insertion Sort to keep the Nodes in order. Do not use any other sort method to keep the shingles in order.

- Print a single line summary of the results of processing the file. The line should contain the information illustrated in the following example where the results will vary depending on the input file:

```
The file src/wfu.txt contains 175 total shingles of length 5 including 158 distinct shingles.
```

Expectations

- This program must be developed by you independently of help from classmates or anyone else other than the TA or instructor. It is okay to ask a classmate a question about syntax or a specific error message. However, it is not okay to give guidance on data structures or conceptual approaches to the problem at hand. It is certainly not okay to share code or to copy someone else's code. Plagiarism will be dealt with through the college Honor and Ethics system and may result in sanctions such as an irreplaceable F or suspension for a semester.
- Your Main method MUST CONTAIN the following heading (with your name filled in) or it will not be graded:

```
/**
 * Project 1 - CSC 221 Spring 2020
 *
 * Honor Pledge:
 * The code submitted for this project was developed by
 * YOUR NAME HERE without outside assistance or consultation
 * except as allowed by the instructions for this project.
 *
 */
```

Evaluation

- 60% - compiles, runs, and produces correct results for wfu.txt, Christie.txt and other test files
- 15% - Unit Test for Node class
- 5% - Uses at least two non-trivial *assert* statements that are useful for program debugging
- 20% - good programming style: variables names, structure, readability, comments, etc.

Turn In

You'll be submitting through GitHub. Detailed instructions will be provided. Late submissions will be reduced in point value by 10% per day for up to 5 days. Submissions will not be accepted after 2/9/20.