Data Structures
COMP 2100
Fall 2022

Project 1: Haiku Very Much

Haiku is a traditional form of poetry that originated in Japan. Wikipedia describes its structure like this:

> "Traditional Japanese haiku consist of three phrases that contain a *kireji*, or "cutting word", 17 *on* (a type of Japanese phoneme) in a 5, 7, 5 pattern, and a *kigo*, or seasonal reference. However, modern haiku vary widely on how closely they follow these traditional elements."

While it is common for English haiku to implement the 5-7-5 pattern as a syllable count, this is not an exact translation of the Japanese *on*. For this reason, it is reasonable in English haiku to deviate from this pattern and have other forms (not based on syllable count) as templates for haiku. For example:

> A little wave…
> The firefly has risen near the shade
> Hidden broken pine

Admittedly, this is not a good poem. That is mainly because it was automatically generated by a computer.

Specification

For this lab project, you are to write an automatic haiku generator. The haiku generator will produce a poem that matches one of several pre-defined patterns, using words that are supplied in a vocabulary input file.

The vocabulary will consist of words from five parts of speech: articles, adjectives, nouns, verbs, and prepositions. A sample input file is provided for download on the course web page. Its format will consist of, for each part of speech, a section header, which is a line beginning with * and the part of speech label, followed by any number of lines each containing one vocabulary word or phrase, as seen here:

```
* Nouns
MOUSE
HOUSE
* Verbs
ATE
CHASED
* Prepositions
WITH

etc.
```

The haiku patterns will consist of sequences of parts of speech (kind of like in Mad-Libs, but with all the words missing) and can be hard-coded as follows:

1. Article-adjective-noun...
   Article-noun-verb-preposition-article-noun
   Adjective-adjective-noun
2. Noun-preposition-article-noun
   Article-adjective-noun-preposition-article-noun
   Adjective-noun
3. Article-adjective-noun
   Preposition-article-adjective-noun
   Article-noun-verb
4. Article-adjective-noun-verb
   Article-adjective-noun
   Preposition-article-adjective-noun

When your program is run the name of the vocabulary file should be passed as a command line argument. The program will then randomly select one of these four patterns, and produce a poem by randomly selecting the appropriate words from the vocabulary lists that were read from the file. The program should offer to produce more poems until the user declines. Here is a sample session:

```
C:\>java Haiku vocab.txt
The hidden breeze has passed
The misty wind
Near an icy snow

Would you like another? Yes
Breeze in the sea
The dry morning over a moon
Misty lake

Would you like another? No
C:\>
```

Your solution should minimally consist of two Java source files: one that is the main application class that contains a main method and the user interaction event loop illustrated above; the other should be a class that represents the vocabulary data in the input file. This constructor for this second class should be passed the name of the input file when it is instantiated from the main application. The full API for this class is shown below.

Class `Vocabulary`

Constructor	`public Vocabulary(String filename)`
	Requires: file specified by filename exists Ensures: instantiates a Vocabulary object that is populated with the data contained in the file
verb	`public String verb(boolean unique)`
	Ensures: returns a random verb from the vocabulary; if unique is true the returned value will be distinct from the most recently produced verb
noun	`public String noun(boolean unique)`
	Ensures: returns a random noun from the vocabulary; if unique is true the returned value will be distinct from the most recently produced noun
article	`public String article()`
	Ensures: returns a random article from the vocabulary
adjective	`public String adjective(boolean unique)`
	Ensures: returns a random adjective from the vocabulary; if unique is true the returned value will be distinct from the most recently produced adjective
preposition	`public String preposition(boolean unique)`
	Ensures: returns a random preposition from the vocabulary; if unique is true the returned value will be distinct from the most recently produced preposition

Considerations

- The Vocabulary class is simply representing what it finds in the input file. It is agnostic about upper and lower case. The client class should do whatever case conversions are necessary. For example, you will notice that in the templates the first word of each line should be capitalized, but nothing else. (Note that I am suggesting here that the template is encoding information about case.)
- You should make sure that any time the article 'A' is returned by vocabulary that it be in agreement with the next word. In other words, if the word following the article starts with a vowel change the article to 'AN'. I would recommend that you implement a method to take care of this.
- The simplest way to select a template is to pick a random integer between 1 and 4 and use a switch statement that hard codes each template. However, I will offer up to 15% extra credit on the project if you instead allow the templates to be provided by the user in a second input file on the command line. I would recommend doing it the easy way first, and then try for the extra credit in a copy of the project. This way if you don't get that part completed you'll still have a working solution.

All work must be done within assigned teams. You may discuss general concepts with your classmates, but it is never acceptable for you to look at another team's code. Please refer to the course policies if you have any questions about academic integrity. If you have trouble with the assignment, I am always available for assistance.

All work is due before Friday, Sept. 16, 2020 at 11:59 p.m.

To Submit: Zip your project and email it as an attachment to dstucki@otterbein.edu