Resource Link: http://www.dukelearntoprogram.com/course4/index.php

For the following assignments, you will start with the files provided that include several data files.

First there are two classes and an interface provided:

The **IMarkovModel** interface. It has three signatures: the void method setTraining that has one String parameter named text, the void method setRandom that has one int parameter named seed, and the method getRandomText that has one int parameter named numChars and returns a String.

The class **MarkovWordOne**, from the lesson, has been started for you but is not complete. It has two private variables, a String array myText, to hold the words from the training text, and myRandom, a random number generator. It also has a constructor to initialize myRandom and four methods:

- The void method setRandom has one integer parameter named seed. Using this method will allow you to generate the same random text each time, which will help in testing your program.
- The void method setTraining has one String parameter named text. The String text is split into words and stored in myText. The words are used to initialize the training text.
- The getRandomText method has one integer parameter named numWords. This method generates and returns random text that has numWords words. This class generates each word by randomly choosing a word from the training text that follows the current word in the training text. This method has already been written for you. However, it does not work completely until you have completed the getFollows method.
- The getFollows method has one String parameter named key. This method returns an ArrayList of all the single words that immediately follow an instance of key somewhere in the training text. This method has been started. You will need to complete it.

The class **MarkovRunner** has four methods.

- Note there are two runModel methods with different numbers of parameters. One void method runModel has three parameters, an IMarkovModel named markov, a String text that represents the training text, and an integer named size that represents the number of random words to generate. The second runModel method has a fourth parameter, an int named seed. Java knows which method to call based on how many parameters there are in your call.
- The void method runMarkov has no parameters. This method reads in a file the user chooses, creates a MarkovWordOne object, and then calls runModel to generate and print three sets of randomly generated text using the file read in to choose the random words.
- The void method printOut is called by runModel to print out the random text that was generated with around 60 characters per line. DO NOT CHANGE THIS METHOD. You'll need output generated in this format for some of the quiz questions.

For these assignments, you will create Markov classes to generate random text using words that follow other words to predict the next possible word. The more words you use for the prediction, the more likely the phrase will appear in the training text.

Assignment 1: Generating Random Words with Prediction

For this assignment you will first complete the MarkovWordOne class. You will need to write a getFollows method that is given a word and returns all the single words that immediately follow that word in the training text. You'll need to write a helper function indexOf that will work like the String indexOf method, but here it returns the index location of a word in the String array. After testing and completing MarkovWordOne, you will create a MarkovWordTwo class.

Specifically, for this assignment, you will:

- In the MarkovWordOne class, write the private method indexOf that has three parameters, a String array named words, a String named target, and an integer named start. This method starts looking at the start position and returns the first index location in words that matches target. If no word is found, then this method returns -1. The signature for this method should be:

```
private int indexOf(String[] words, String target, int start)
```

- In the MarkovWordOne class, write a public void method named testIndexOf that has no parameters. This method is only for testing the indexOf method. This method should create a simple String array with the words "this is just a test yes this is a simple test" then look for the words: "this" starting at 0, "this" starting at 3, "frog" starting at 0, "frog" starting at 5, "simple" starting at 2 and "test" starting at 5.
- Complete the method named getFollows that has been started for you with String parameter key. This method should return an ArrayList of all the single words that immediately follow the key in the training text. This method should call the indexOf method.
- You can test the getFollows method by adding a print statement in the getRandomText method to print out a key and the result from calling getFollows to see all the words that follow the key. Then write a testing method in the MarkovRunner class. This method should create a simple String instead of using a file for testing and call runModel. For example, for the String above, the word "test" is followed by "yes", the word "is" is followed by "just" and "a".
- Once you have MarkovWordOne working, modify the runMarkov method in the MarkovRunner class to use the random seed set to 175, and generate 120 words. Then run this method on the file confucius.txt. You should get the output (first five lines shown):

```
teacher. 12. He that his doings from the free distribution of

Ling, the people by learning; then those over with this or hate

daring and hates his muttering a belief in war. He refused all

likelihood, between men. Fan Ch'ih did not go with you? The Master

said, At his best pupil, who notifies you may ignore propriety;
```

Assignment 2: MarkovWordTwo

For this assignment you will complete the MarkovWordTwo class.

Specifically, you should do the following:

- Create a new class named MarkovWordTwo. Copy the code from MarkovWordOne into MarkovWordTwo, and then modify the code to work for two consecutive words. You should have two keys: key1 and key2 that are consecutive words in the text, and then getFollows returns the list of single words that follow these two words. For example if the text was "this is just a test yes this is a simple test", then getFollows of "this" "is" would return an ArrayList with "just" and "a", and getFollows of "just" "a" returns an ArrayList with "test". Specifically, we will do it slightly differently than in the videos:

- getFollows should have two String parameters, key1 and key2

- indexOf should have four parameters: a String array words, a String target1, a String target2, and an integer start. It returns the first location of target1 such that target2 immediately follows it, and the search starts looking at index start.

- Be sure to test these methods.

- Write a new method runMarkovTwo in the MarkovRunner class to generate random text with MarkovWordTwo. Run this on confucius.txt with the random number seed 549. The first line of your output should be:

```
the minister know me? Because I was not so great; xix. 21, says
```

Programming Exercise - Word N-Grams.pdf

Mark as completed