CS 201: Hangman

Scotty Shaw (sks6)

For Hangman, lowerwords.txt provided a large list of words, all of varying lengths. One interesting piece of statistical information is how many different words are of length 4, 5, 6, and so on, up to 20 letters. To estimate this, I modified the original code, which provided a for loop that calculated the number of 4-letter words. In the code provided below, the inner for loop, using the integer k, would attempt to add 1000 unique words to a previously declared hash set named "set" in this method.

```
for (int j = 4; j <= 20; j++) {
	for (int k = 0; k < 1000; k++) {
		set.add(loader.getRandomWord(j));
	}
	System.out.printf("number of " + j + " letter words = %d\n", set.size());
	set = new HashSet<String>();
	}
```

With a hash set, duplicate words do not increase the total of unique words, and running the inner for loop multiple times yields around 815 unique 4-letter words. In order to gather the same information for words of other lengths, I added an outer for loop, using integer j, to test for word lengths of 4 through 20. As a result, integer j was inserted into the argument field of getRandomWord to provide the target length. After the inner for loop added 1000 words, some of which were duplicates, to the hash set, the code outputs the resulting number of unique words for that length before resetting the hash set for the next test length.

The results indicate that attempting to add 1000 unique words of various lengths from lowerwords.txt into a hash set would yield the following amounts for each word length.

4-letter words	~815
5-letter words	~885
6-letter words	~920
7-letter words	~935
8-letter words	~935
9-letter words	~915
10-letter words	~900
11-letter words	~850
12-letter words	~775
13-letter words	~665
14-letter words	~460
15-letter words	~270
16-letter words	103
17-letter words	57
18-letter words	23
19-letter words	3
20-letter words	3

I noticed that as the numbers decreased, the statistical spread decreased as well, leading to greater accuracy in calculations. The list also seems to have more words of lengths 7 and 8 than others, although 5-letter to 11-letter words were all extremely common, with 4-letter and 12-letter words close behind.

Another question I answered was how many unique 4-letter words could be added before the first duplicate. I again created an inner for loop and an outer for loop. The outer for loop was designed to run the inner loop test a double numTests number of times, which I set to 1000.0 for this case. I then created a new hash set named set2 before setting up the inner for loop, which attempted to add up to 1000 unique words of length, designated as integer lettersInWord, or 4.

```
for (int i = 0; i < numTests; i++) {
	HashSet<String> set2 = new HashSet<String>();
	for (int k = 0; k < 1000; k++) {
		added = set2.add(loader.getRandomWord(lettersInWord));
		if (added == false) {
			break;
		}
		sumUniqueWords++;
	}
}

average = (double) (sumUniqueWords) / numTests;
System.out.printf("Average number of unique " + lettersInWord + " letter words
added to set before first duplicate = %f\n", average);
```

A boolean added was necessary to detect when a duplicate was added to set2. The function loader would get a random word of length lettersInWord and add it to set2, and if it was a duplicate, the boolean added would be set to false. That would then trigger the if statement's condition that if added was false, the loop would break, thus ending the test and completing the inner loop once. Until then, each unique word would increment integer sumUniqueWords to keep a total count of unique words added throughout the number of tests.

Finally, the code calculates the double average by setting sumUniqueWords to type double and dividing it by the double numTests, and the last line of code would print a statement telling the user what the average number of unique lettersInWord-letter words, 4 in this case, added to set2 before the first duplicate occurred was. My results usually fell somewhere between 57.3 and 61.1, with the statistical mean approaching 58.9 as I increased the size of numTests.

Although I didn't write code for other questions, it is definitely feasible to calculate the average number of unique words added to a set before the first duplicate for words of other lengths. Other questions that could be asked are what the mean and median word lengths are, which seem to be somewhere around 7 or 8 letters, or what letters appear the most for various word lengths. And finally, tests could also be run to determine which prefixes and suffixes appear the most and at what word lengths.