Hangman — Full Writeup

Keng Low
kl78@duke

Goals
1. Understand how classes call and interact in Java
2. Gain familiarity with strings, arrays, loops, and printing in Java
3. Practice a forensic/scientific/analytical approach to data that we'll build on during the semester
4. Understand the differences between primitive types and objects/classes

Part 1a: Statistical/Forensic Analysis

> *We want you to estimate the number of different words there are of 4, 5, 6, and so on up to 20 letters long. For example, to estimate the number of 6-letter words you can repeatedly call HangmanFileLoader.getRandomWord(6), and add the returned word to a set. The size of the set is an estimate of the number of words of length 6.*

Word length	Size of word set (in terms of word count)
4	812
5	896
6	928
7	936
8	939
9	900
10	904
11	867
12	779
13	671
14	451
15	269
16	103
17	57
18	23
19	3
20	3

Part 1b: Your Own Question

> *Pose your own question about the words and the code that generates the random words of a specific length in HangmanFileLoader. Write code to answer your question and include the data in the analytical write-up you turn in. You could, for example, wonder:*
>
> - *On average, how many calls are needed before some word is returned that was previously returned?*
>
> *Remember, to calculate an average you will need to run your code multiple times to collect data. Any question you write is fine, but you must write code to answer the question. You're welcome to come up with 'extra' questions (up to three) that you don't write code for, but which you'd like to be able to answer/write code to answer. Include all questions in your analytic write-up.*

How does `k` affect word count?
In the `for` loop, `for(int k=0; k < 1000; k += 1)`, there is an upper limit initially set at k < 1000. For each iteration of count k, that represents the number of attempts, `set.add(loader.getRandomWord(4));` where in this case the app would get a randomly generated four-letter word in **1000** attempts. In this case, it gives me about 810 occurences.

I slowly increased the value of k in increments of increasing factors. This was achieved by `for (int i = 0; i < 10000; i++)`. I found that **1000** is a small value and the upper limits of the number of 4-letter words is reached between a factor of 10 to 20. In this case, the number of words is twice as large at an averago of **2200**.