Xing Su (xs17)
CompSci 201
1/24/13

Hangman Assignment

Part 1a: I used the given code as a guide to write a nested loop that goes from i = 4 to 20 to go through the different lengths of words, and make 10000 calls of the getRandomWord() method for each given word length. Since each word that is returned from the getRandomWord() method is stored in a HashSet (which doesn't allow duplicates), we can get an estimate of how many unique words there are for each given length from 4 to 20. Once the 10000 calls of getRandomWord() are made for a given length, I used printf to print out the result and then proceeded to clear the set in order to start over for the next length. Below is the code:

```
HangmanFileLoader loader = new HangmanFileLoader();
loader.readFile("lowerwords.txt");
HashSet<String> set = new HashSet<String>();

// Part 1a
for (int i = 4; i <= 20; i++) {
      for (int k = 0; k < 10000; k += 1) {
            set.add(loader.getRandomWord(i));
      }
      System.out.printf("number of %d letter words = %d\n", i,
      set.size());
      set.clear();
}
```

Here's a sample out put from the above code.

```
number of 4 letter words = 2210
number of 5 letter words = 3818
number of 6 letter words = 4958
number of 7 letter words = 5438
number of 8 letter words = 5320
number of 9 letter words = 4943
number of 10 letter words = 4074
number of 11 letter words = 2961
number of 12 letter words = 1873
number of 13 letter words = 1137
number of 14 letter words = 545
number of 15 letter words = 278
number of 16 letter words = 103
number of 17 letter words = 57
```

```
number of 18 letter words = 23
number of 19 letter words = 3
number of 20 letter words = 3
```

The results are tabulated as below:

Word Length	Number of Words
4	2210
5	3818
6	4958
7	5438
8	5320
9	4943
10	4074
11	2961
12	1873
13	1137
14	545
15	278
16	103
17	57
18	23
19	3
20	3

Part 1b: I chose to answer the question “On average, how many calls are needed before the same word is returned twice in a row for a given word length?” and chose to use a nested loop again. I began by instantiating two empty strings and two ints. Here I set the count to start at one instead of zero because in order to return the same word twice, I have to make at least two calls. Thus, initializing count at 1 is essentially counting the first getRandomWord() call. Ignoring the outside loop for now, the String benchmark is set to a random string of length j from lowercase.txt(4 in the very first run through) to begin with. Next a while loop is called to set String current as another random string of the equal length from the same text file. Then, the two strings are compared to each other: if they are the same, meaning that the same string has been returned twice in a row, index or the count of successes increases by one; if the strings are different, the benchmark string is set as the current string and the count (number of calls made) increases by one. This loop will continue until index is 49, meaning that 50 successes or words returned twice in a row are accounted for. Since count is never reset within the while loop, the final count would be the total number of calls to getRandomWord() made to get 50 successes. Total calls divided by 50 would equal to the average number of calls needed to return one word twice. After all that is done, count is set back to 1 and index is set back to 0 to get ready for the next word length. Below is the code used:

```
String benchmark = "";
String current = "";
int count = 1;
int index = 0;
for (int j = 4; j <= 20; j++) {
      benchmark = loader.getRandomWord(j);
      while(index < 50){
            current = loader.getRandomWord(j);
            if(current.equals(benchmark)){
                  index++;
            }
            benchmark = current;
            count ++;
      }
      System.out.printf("\n%d-letter word average: %f", j,
            count/50.0);
      count = 1;
      index = 0;
}
```

Here’s a sample output from the code above:

```
4-letter word average: 2010.760000
5-letter word average: 4697.100000
```

```
6-letter word average: 5133.880000
7-letter word average: 6919.580000
8-letter word average: 6867.840000
9-letter word average: 5691.400000
10-letter word average: 3765.840000
11-letter word average: 3334.560000
12-letter word average: 1274.860000
13-letter word average: 1085.640000
14-letter word average: 539.460000
15-letter word average: 229.100000
16-letter word average: 97.940000
17-letter word average: 66.560000
18-letter word average: 23.840000
19-letter word average: 3.580000
20-letter word average: 2.880000
```

The results are tabulated below:

Word Length	Number of Words
4	2010.76
5	4697.1
6	5133.88
7	6919.58
8	6867.84
9	5691.4
10	3765.84
11	3334.56
12	1274.86
13	1085.64
14	539.46
15	229.1
16	97.94
17	66.56
18	23.84
19	3.58
20	2.88