Tianyi Mu
CS 201
Hangman

Part 1a

The method I used to get a fairly good estimate for word counts is to use HashSet to keep track of the recorded word count for a specific length of words and then see if the newly recorded word count matches one of the recorded word count in the Hashset.
Because every 10,000 call to the HangmanLoader.getRandomWord will return a different count for a specific length of words, and because it is possible to get the same count when 10,000 call to the HangmanLoader.getRandomWord is ran twice, my method will add the first 5 counts from 5 runs of 10,000 HangmanLoader.getRandomWord calls to a HashSet regardless of their values.
After, the sixth count generated by 10,000 HangmanLoader.getRandomWord calls is compared with the existing counts in the HashSet.
If the sixth count does not match any count in the HashSet, the iteration (originally 10,000) is added by 20,000 for every time the matching test fails.
If the a new count finally matches one of the old counts, then the calculation stops there for that specific length of words and the estimate word count for that specific length of words is then returned.

```
HashMap <Integer, Integer> time = new HashMap<Integer, Integer>(21-4);
                for(int i =4;i<=20;i++)
                {
                        int iter =10000;
                HashSet<Integer> count = new HashSet<Integer>();
                HashSet<String> set1 = new HashSet<String>();
                for(int j=0;j<5;j++)
                {
                set1.clear();
                for(int k=0; k < iter; k += 1)
                {
                        set1.add(loader.getRandomWord(i));
                        count.add(set1.size());
                }
                }
                for(int k=0; k < iter; k += 1)
                {
                        set1.add(loader.getRandomWord(i));
                }
                while(!count.contains(set1.size()))
                {
                        iter=iter+20000;
                        set1.clear();
                        for(int k=0; k < iter; k += 1)
                        {
                                set1.add(loader.getRandomWord(i));
                        }
                        count.add(set1.size());

                }
```

```
                time.put(i, set1.size());
                }
```

Using this method, the estimates are very close to the actual counts.
Print out of estimates:

```
number of 4 letter words is 2235
number of 5 letter words is 4165
number of 6 letter words is 6125
number of 7 letter words is 7238
number of 8 letter words is 6990
number of 9 letter words is 6035
number of 10 letter words is 4586
number of 11 letter words is 3069
number of 12 letter words is 1880
number of 13 letter words is 1137
number of 14 letter words is 545
number of 15 letter words is 278
number of 16 letter words is 103
number of 17 letter words is 57
number of 18 letter words is 23
number of 19 letter words is 3
number of 20 letter words is 3
```

Calling the HangmanLoader.getRandomWord 100,000 times to get the actual counts.

```
for(int i =4;i<=20;i++)
                {
                        HashSet<String> set1 = new HashSet<String>();
                        for(int k=0; k < 100000; k += 1)
                        {
                                set1.add(loader.getRandomWord(i));
                        }
                        System.out.println("number of "+i+" letter words is "+set1.size());
                }
```

Print out of actual counts:

```
number of 4 letter words is 2235
number of 5 letter words is 4170
number of 6 letter words is 6166
number of 7 letter words is 7359
number of 8 letter words is 7070
number of 9 letter words is 6079
number of 10 letter words is 4591
number of 11 letter words is 3069
number of 12 letter words is 1880
number of 13 letter words is 1137
number of 14 letter words is 545
number of 15 letter words is 278
number of 16 letter words is 103
number of 17 letter words is 57
number of 18 letter words is 23
number of 19 letter words is 3
number of 20 letter words is 3
```

Part 1b

The question is “What is the most occurred word?”
First, the method of estimating word counts is the same as in part 1a. For comparing word count, the method will compare the word count of n+1 characters word to that of n characters word. If the word count of n+1 characters word is larger, then the count is saved for later comparison and its character length is also recorded. At the end, after all comparisons, the method will print out the character length of the most occurred word and its estimated word count.

```
	public void statisticalQuestion(HangmanFileLoader loader){
		int length=0;
		int size=0;
		HashSet<Integer> count = new HashSet<Integer>();
		HashSet<String> set1 = new HashSet<String>();
		for(int i =4;i<=20;i++)
		{
			int iter =10000;
		if(set1.size()>size)
		{
			size=set1.size();
			length =i-1;
		}
		set1.clear();
		count.clear();
		for(int j=0;j<5;j++)
		{
		set1.clear();
		for(int k=0; k < iter; k += 1)
		{
			set1.add(loader.getRandomWord(i));
			count.add(set1.size());
		}
		}
		for(int k=0; k < iter; k += 1)
		{
			set1.add(loader.getRandomWord(i));
		}
		while(!count.contains(set1.size()))
		{
			iter=iter+20000;
			set1.clear();
			for(int k=0; k < iter; k += 1)
			{
				set1.add(loader.getRandomWord(i));
			}
			count.add(set1.size());
		}

		}
		System.out.println("Most occured words are "+length+" characters long.
There are "+size+".");
	}
```

The print out after running this code is:

```
Most occured words are 7 characters long. There are 7236.
```

Judging by the previous estimates and the actual counts, the result is correct.

Extra Credit

I added the option for users to play hangman with a food wordlist or a name wordlist in addition to the default wordlist provided with the project.
I got the wordlists online and saved them as .txt files.
When the game starts, it will ask the user to choose one of the three wordlists he wants to play with. Once the user enters the selection, the program will identify it and load the corresponding wordlist for the user to play hangman with.