Callie Mao

Hangman Writeup

Part 1A

Printout:

```
number of words of length 4 =    2201
number of words of length 5 =    3786
number of words of length 6 =    4991
number of words of length 7 =    5448
number of words of length 8 =    5338
number of words of length 9 =    4932
number of words of length 10 =   4100
number of words of length 11 =   2946
number of words of length 12 =   1867
number of words of length 13 =   1137
number of words of length 14 =   545
number of words of length 15 =   278
number of words of length 16 =   103
number of words of length 17 =   57
number of words of length 18 =   23
number of words of length 19 =   3
number of words of length 20 =   3
```

I used the method numberOfWords() in HangmanStats to estimate the number of words of length "start" through length "end". I call the method numberOfWords() in the main method with the parameters 4 and 20 for "start" and "end" respectively and obtain the number of estimates of the different words there are of 4, 5, 6, … , 20 letters long. In the method, getRandomWord() is iterated 10,000 times from the HangmanFileLoader "loader". Changing the number of iterations to 100,000 gives results (number of words for length 4 through 20) that do no change with each run, which are no longer estimates, but the exact count. Thus, I used 10,000 iterations, which loses some accuracy, but makes the code more efficient to make up for the loss.The hashSet used in this method ensured that duplicates are not stored, and thus the size of the hashSet is effectively the estimate of number of words of x letters.

Part 1B

Question: *How many times on average does the method getRandomWord() need to be called in order to repeat a word of length 10?* There is an estimated 4769 calls needed before this repeat word is returned. I used the method numCalls() to return the number of calls to getRandomWord() that would get a repeat of a word of length 10. I call the method getRandomWord(num) for num = 10 (10 letters) to get the first random word of length 10. This value is then saved. Then, I iterate in a while loop

while keeping a counter, until getRandomWord() returns the same word as that of the first call. The counter tells me how many calls were needed in order to retrieve the initial word again. In the main method, I call the method numCalls() 100 times and calculate the average by summing all the results and dividing by 100. This method can be repeated for all lengths of words from 2 to 20 to see how many calls are needed to repeat a word of different lengths and can answer the questions *how many times on average does the method getRandomWord() need to be called in order to repeat a word of length 2, 3, 4…., or 20?*

Printout:

```
Part 1B (please allow time for result to calculate)
Average calls to get a repeat of a word of length 10: 4694
```

Code:

```
public int numCalls(HangmanFileLoader loader, int num){
      HangmanFileLoader data = new HangmanFileLoader();
      data.readFile("lowerwords.txt");
      String randomWord = data.getRandomWord(num);

      String newWord="";
      int counter = 0;
      while (!newWord.equals(randomWord)){
            newWord = data.getRandomWord(num);
            counter ++;
      }
      return counter;
}
```

Extra Credit

I added an initial question to the method play() in HangmanGame.java that asks the user for what type of category of words the user wishes to guess from. “Actors” and “books” are new categories from the new text files “actors.txt” and “books.txt”, while “dictionary” asks standard words from the original “lowerwords.txt” file. Since books and actors often contain spaces and occasionally punctuation (for books), I edited the HangmanFileLoad class to trim out all spaces and comma’s when saving the length of the words in order to ensure that spaces and comma’s were not counted in a title or name’s length. Since both are also proper nouns and the text file had capitalized letters, I had to include a Character.toLowerCase() to convert all letters to lower cases.