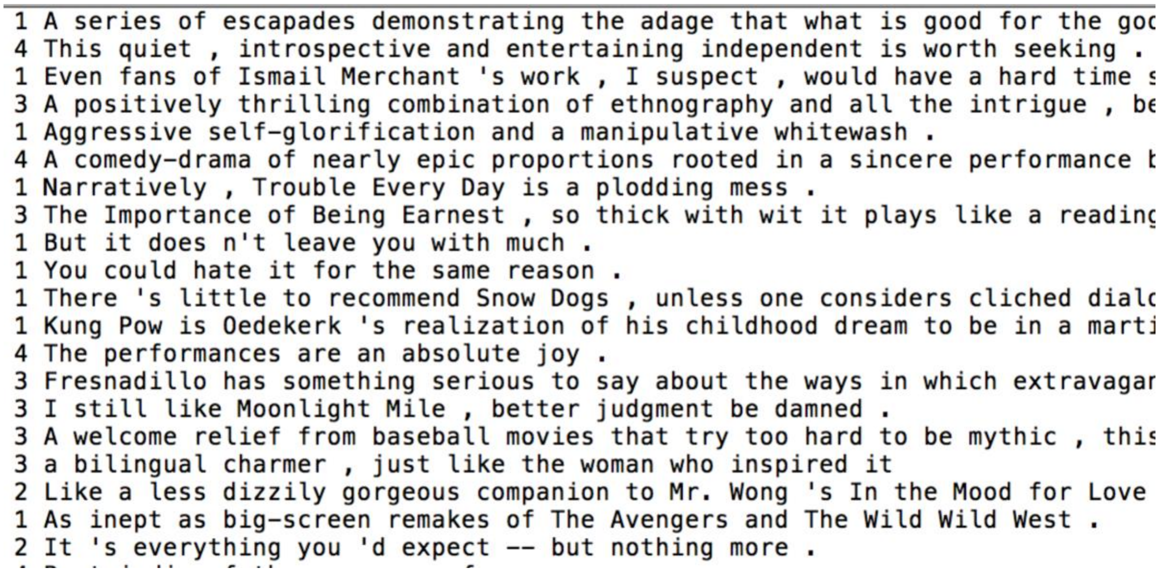
Movie Review Sentiment Analysis (CS1)

Sentiment Analysis is a Big Data problem which seeks to determine the general attitude of a writer given some text they have written. For instance, we would like to have a program that could look at the text "The film was a breath of fresh air" and realize that it was a positive statement while "It made me want to poke out my eye balls" is negative.

One algorithm that we can use for this is to assign a numeric value to any given word based on how positive or negative that word is and then score the statement based on the values of the words. But, how do we come up with our word scores in the first place?

That's the problem that we'll solve in this assignment. You are going to search through a file containing movie reviews from the Rotten Tomatoes website which have both a numeric score as well as text. You'll use this to learn which words are positive and which are negative. The data file looks like this:



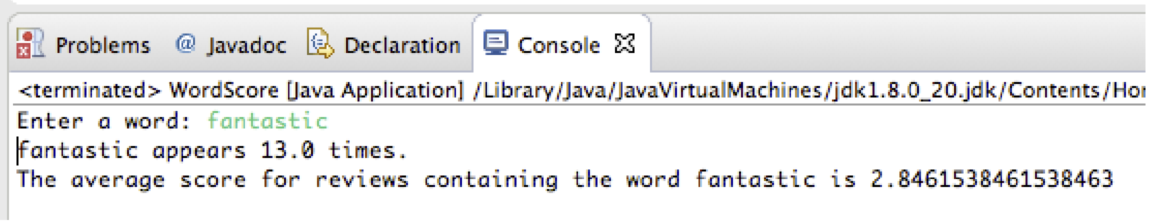
```
1 A series of escapades demonstrating the adage that what is good for the goc
4 This quiet , introspective and entertaining independent is worth seeking .
1 Even fans of Ismail Merchant 's work , I suspect , would have a hard time s
3 A positively thrilling combination of ethnography and all the intrigue , be
1 Aggressive self-glorification and a manipulative whitewash .
4 A comedy-drama of nearly epic proportions rooted in a sincere performance b
1 Narratively , Trouble Every Day is a plodding mess .
3 The Importance of Being Earnest , so thick with wit it plays like a reading
1 But it does n't leave you with much .
1 You could hate it for the same reason .
1 There 's little to recommend Snow Dogs , unless one considers cliched dialc
1 Kung Pow is Oedekerk 's realization of his childhood dream to be in a marti
4 The performances are an absolute joy .
3 Fresnadillo has something serious to say about the ways in which extravagar
3 I still like Moonlight Mile , better judgment be damned .
3 A welcome relief from baseball movies that try too hard to be mythic , this
3 a bilingual charmer , just like the woman who inspired it
2 Like a less dizzily gorgeous companion to Mr. Wong 's In the Mood for Love
1 As inept as big-screen remakes of The Avengers and The Wild Wild West .
2 It 's everything you 'd expect -- but nothing more .
```

Note that each review starts with a number 0 through 4 with the following meaning:

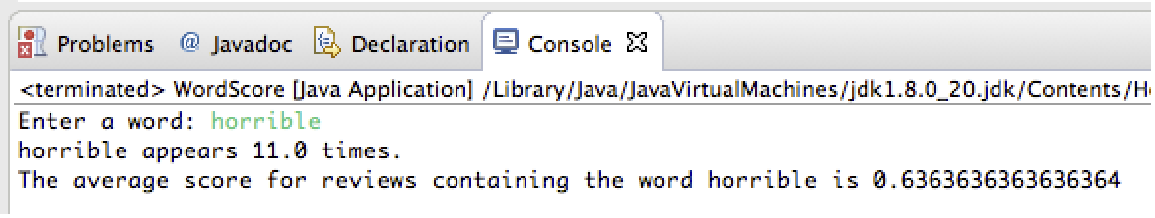
- 0 : negative
- 1 : somewhat negative
- 2 : neutral
- 3 : somewhat positive
- 4 : positive

1. (30 points) For the base assignment, you will ask the user to enter a word, and then you will search every movie review for that word. If you find it, add the score for that review to the word's running score total (i.e., an accumulator variable). You also will need to keep track of how many appearances the word made so that you can report the average score of reviews containing that word back to the user.

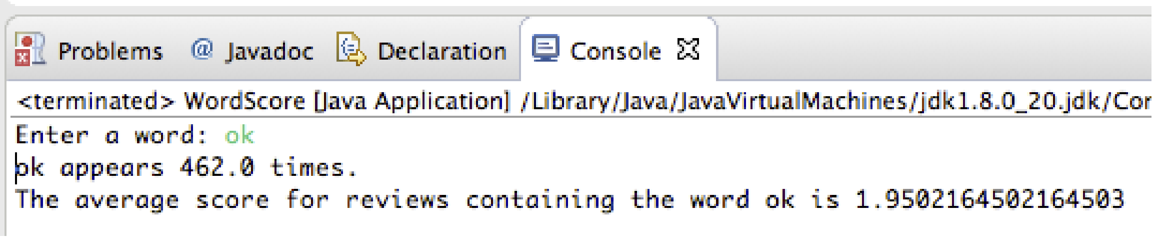
Some sample runs of the program might look like this:



```
<terminated> WordScore [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Hor
Enter a word: fantastic
fantastic appears 13.0 times.
The average score for reviews containing the word fantastic is 2.8461538461538463
```

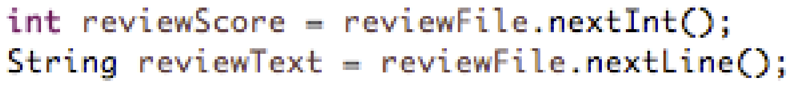


```
<terminated> WordScore [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/H
Enter a word: horrible
horrible appears 11.0 times.
The average score for reviews containing the word horrible is 0.6363636363636364
```



```
<terminated> WordScore [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Cor
Enter a word: ok
ok appears 462.0 times.
The average score for reviews containing the word ok is 1.9502164502164503
```

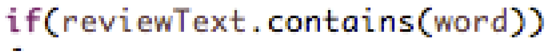
Note that as you search through the lines of text in the file, since the number and the text appear on the same line, you can get them both with statements like



```
int reviewScore = reviewFile.nextInt();
String reviewText = reviewFile.nextLine();
```

This is a case where eating the leftover newline character after a number would be a bad idea because the rest of the line contains the review text.

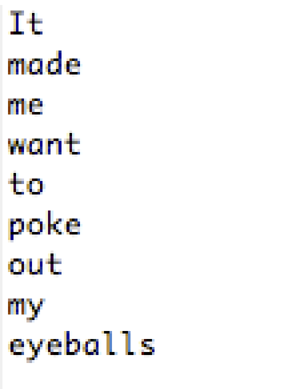
Note also that you can check if a string contains another string as a substring using the `.contains()` String method. If `reviewText` is the variable containing the whole review, and word is the variable containing the word that the user is asking about, then you might check if the review has that word in it, with a statement like this:



```
if(reviewText.contains(word))
```

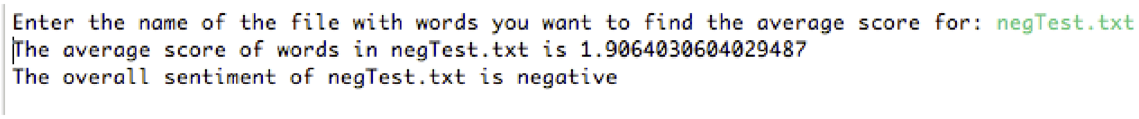
2. (10 points) For an additional 10 points, ask the user to give you the name of a file containing a series of words, one-per-line, and compute the score of every word in the file. Report back to the user the average score of the words in the file. This will allow you to predict the overall sentiment of the phrase represented by words in the file. Consider

an average word score above 2.01 as an overall positive sentiment and consider average score below 1.99 to have an overall negative sentiment. As an example, for a file called `negTest.txt` containing words like this:



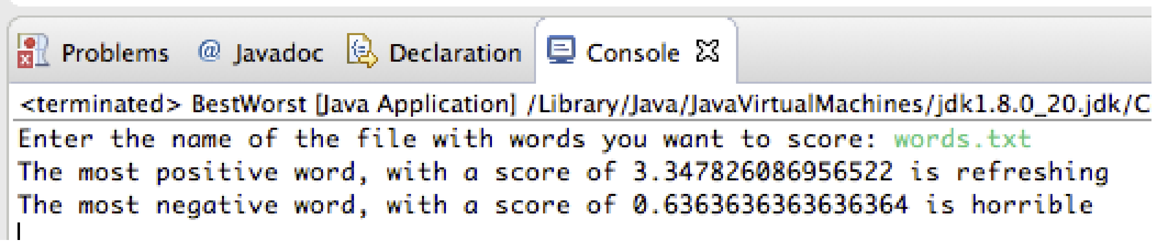
```
It
made
me
want
to
poke
out
my
eyeballs
```

A sample run might look like this:



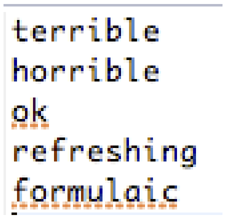
```
Enter the name of the file with words you want to find the average score for: negTest.txt
The average score of words in negTest.txt is 1.9064030604029487
The overall sentiment of negTest.txt is negative
```

3. (10 points) For an additional 10 points, ask the user to give you the name of a file containing a series of words, one-per-line, and compute the score of every word in the file. Report back to the user which word was the most positive and which was the most negative. An example run might look like this:



```
<terminated> BestWorst [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/C
Enter the name of the file with words you want to score: words.txt
The most positive word, with a score of 3.347826086956522 is refreshing
The most negative word, with a score of 0.6363636363636364 is horrible
```

for a file that looks like this:



```
terrible
horrible
ok
refreshing
formulaic
```

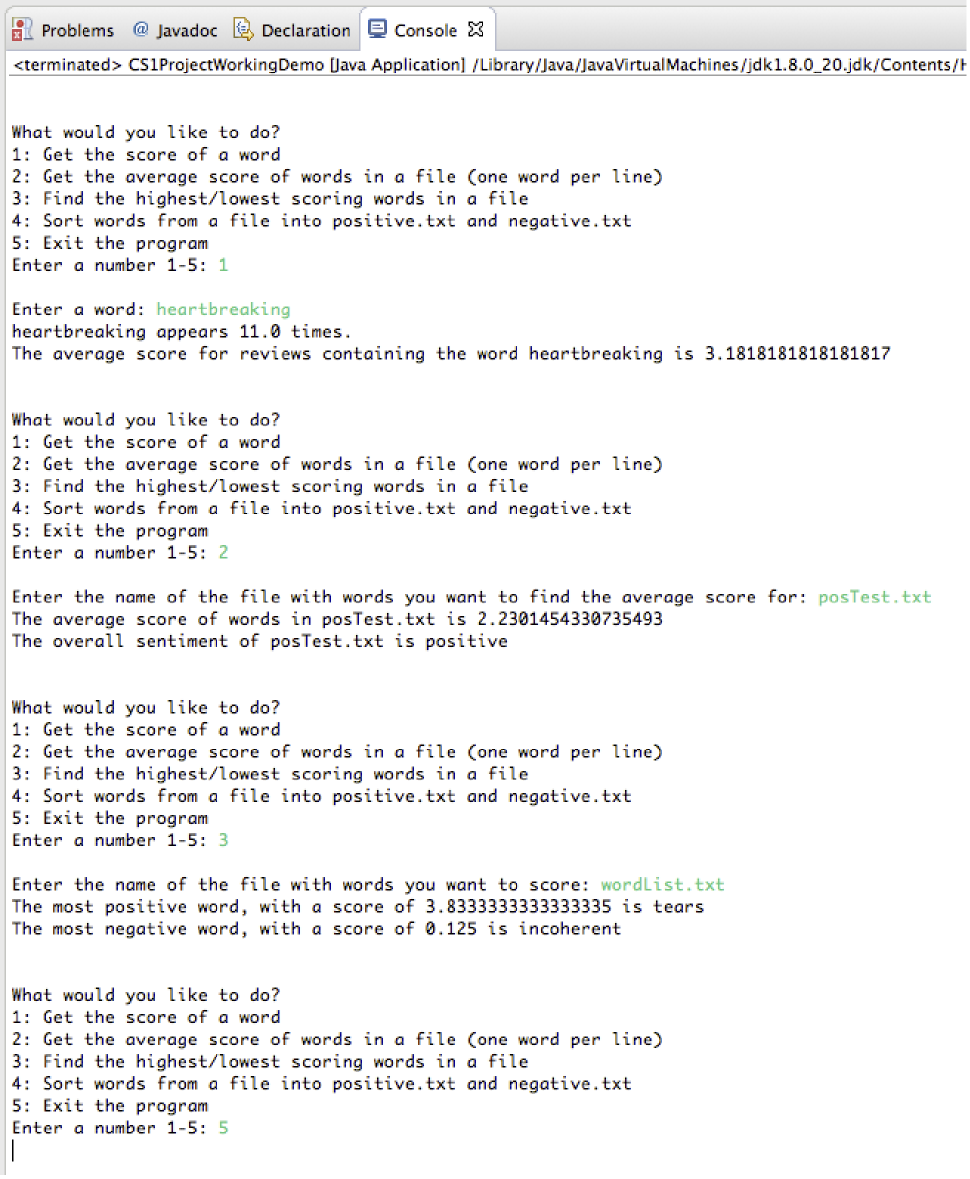
To get these points, you may either just put the code after the code from the base assignment, so it would ask for the filename right after telling you the other word score, or you can give the user a menu offering them choices about what they'd like to do.

4. (10 points) For an additional 10 points, add functionality that will ask the user to enter a word file like in the previous step, but instead of reporting the best and the worst word, create two files called positive.txt and negative.txt, sorting words that have scores below 1.9 into negative.txt, and words that have scores above 2.1 into positive.txt (and just leave out words in between).

To get these points, you may either just put the code after the code from the previous part or you can give the user a menu offering them choices about what they'd like to do.

5. (5 points) Put the code from the above three parts (or two or one, depending on how many you attempted) into their own methods and call them as appropriate.

6. (5 points) Create a menu that allows the user to pick the functionality that they want from the choices. When finished with it, present the menu again until the user chooses to exit. A sample run of the menu might look like this.



```
<terminated> CS1ProjectWorkingDemo [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/


What would you like to do?
1: Get the score of a word
2: Get the average score of words in a file (one word per line)
3: Find the highest/lowest scoring words in a file
4: Sort words from a file into positive.txt and negative.txt
5: Exit the program
Enter a number 1-5: 1

Enter a word: heartbreaking
heartbreaking appears 11.0 times.
The average score for reviews containing the word heartbreaking is 3.1818181818181817


What would you like to do?
1: Get the score of a word
2: Get the average score of words in a file (one word per line)
3: Find the highest/lowest scoring words in a file
4: Sort words from a file into positive.txt and negative.txt
5: Exit the program
Enter a number 1-5: 2

Enter the name of the file with words you want to find the average score for: posTest.txt
The average score of words in posTest.txt is 2.2301454330735493
The overall sentiment of posTest.txt is positive


What would you like to do?
1: Get the score of a word
2: Get the average score of words in a file (one word per line)
3: Find the highest/lowest scoring words in a file
4: Sort words from a file into positive.txt and negative.txt
5: Exit the program
Enter a number 1-5: 3

Enter the name of the file with words you want to score: wordList.txt
The most positive word, with a score of 3.8333333333333335 is tears
The most negative word, with a score of 0.125 is incoherent


What would you like to do?
1: Get the score of a word
2: Get the average score of words in a file (one word per line)
3: Find the highest/lowest scoring words in a file
4: Sort words from a file into positive.txt and negative.txt
5: Exit the program
Enter a number 1-5: 5
```