

Collaboration policy: This is an assignment to be completed individually. General discussion among students in this class and elsewhere, about how to use Java, how to create zip files, etc. are acceptable. But no discussion of how to solve the problem is allowed. You may post general questions, but no code, to piazza. Questions asking for clarification of the specifics of the problem are allowed.

Movie Review Sentiment Analysis

Sentiment Analysis is a Data Science 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 god
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 t
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 dialo
1 Kung Pow is Oedekerk 's realization of his childhood dream to be in a marti
4 The performances are an absolute joy .
3 Fresnoadillo 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 .
```

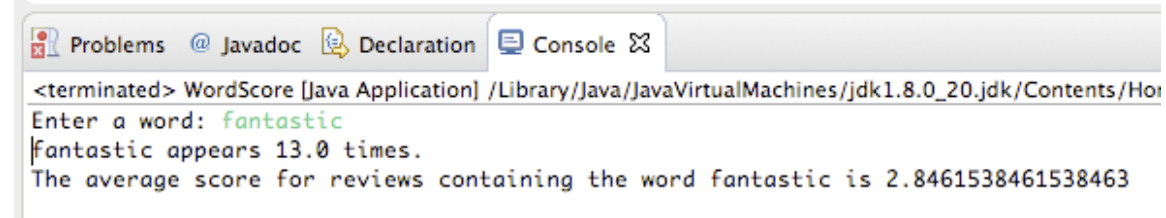
Note that each review starts with a number 0 through 4 with the following meaning:

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

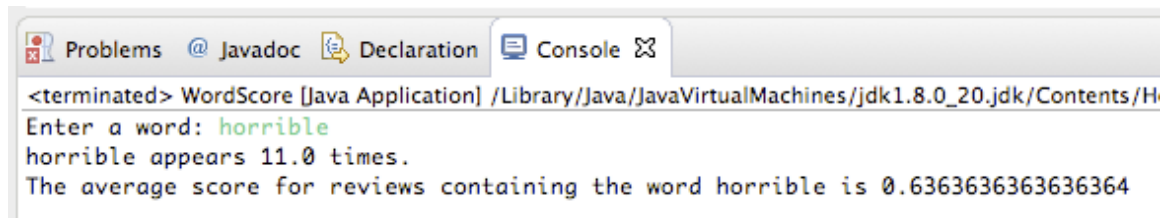
1. (30 points) For the base assignment, you will ask the user to enter a word, and then you will search every movie review for that word. If you find it, add the score for that review to the word’s running score total (i.e., an accumulator variable). You also will need to keep track of how many

appearances the word made so that you can report the average score of reviews containing that word back to the user.

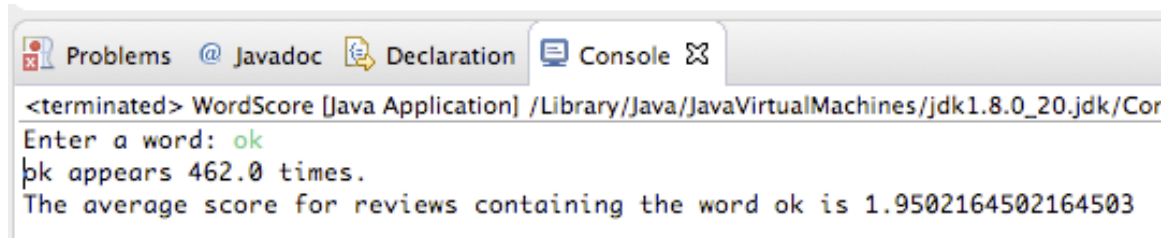
Some sample runs of the program might look like this:



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

The provided template currently asks the user for a word, and reads through the file and displays the numeric rating and textual review for movies containing that word. Modify the program to do the accomplish the rest of what is described above. Also, allow the user to repeatedly enter words until they choose to stop; after each report to them ask them if they wish to enter another word, if they say “yes” in any case-independent way (“yes” or “Yes” or “yeS” or “YES”, etc) , ask them for another word.

2. (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

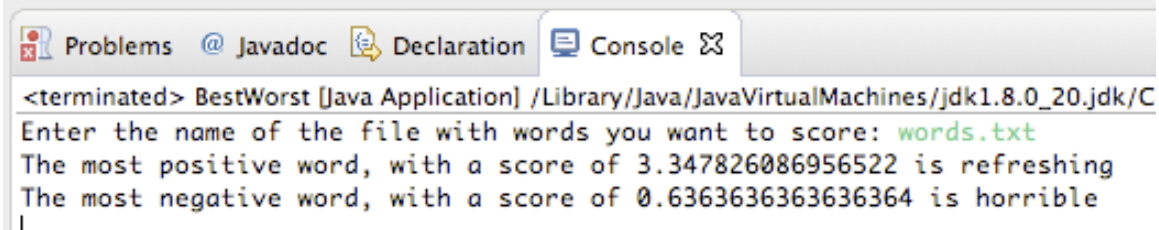
the words in the file. This will allow you to predict the overall sentiment of the phrase represented by words in the file. Consider an average word score above 2.01 as an overall positive sentiment and consider average score below 1.99 to have an overall negative sentiment. As an example, for a file 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) 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

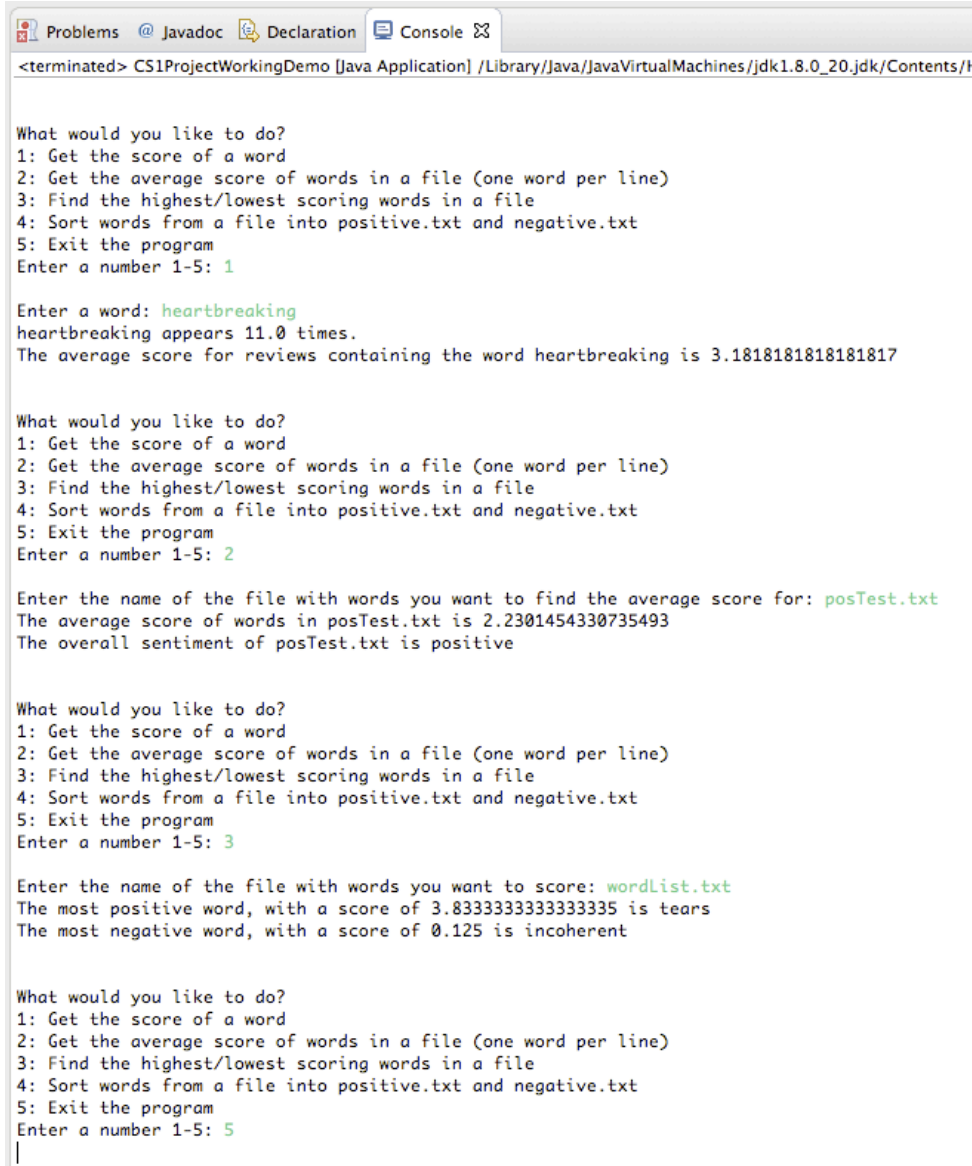


```
<terminated> BestWorst [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/C
Enter the name of the file with words you want to score: words.txt
The most positive word, with a score of 3.347826086956522 is refreshing
The most negative word, with a score of 0.6363636363636364 is horrible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

for a file that looks like this:

```
terrible
horrible
ok
refreshing
formulaic
```

4. (10 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. (Note that the menu also includes the bonus option, discussed next.) (Note that item 4 is for the optional bonus. Please include 4 in your menu, but it is ok that a request for 4 does nothing.) The user should be forced to enter a valid number: 1-5.



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

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 3.18181818181817

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
|
```

5. (BONUS 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).

Program Submission

Create a **ZIP file** that only contains .java files and txt files (no other files types). **DO NOT** include a file hierarchy (i.e. the package structure) – just one or more .java files. For the ZIP file you must use the naming convention: <lastname>HW1.zip e.g., **McCauleyHW1.zip** If the assignment is not submitted in the correct format – **it will not be graded – no exceptions!** Submit the ZIP file via OAKS in the Dropbox that corresponds to the assignment. Resubmit as many times as you like, the newest submission will be the graded submission.

Note – to create such a zip folder:

1. Create an empty folder with the name <lastname>HW1
2. Move your ProgHW1.java file and any text files into it.

3. Zip/compress it into <lastname>HW1.zip
4. Upload to OAKS.

Note to non-BlueJ users – your project folder has a multiple levels of directories, you cannot submit a zipped project folder!!! You must follow the directions above to create the directory structure that will be gradeable.

Note to BlueJ users – if you name your project <lastname>HW1, then you will be able to zip that folder as is when you are done, and submit. You can also remove any non-java or txt files before zipping.

To test if you have created a proper submission file, assuming your last name is Nero:

1. Unzip NeroHW1.zip
2. You should now see a folder named NeroHW1
3. Look in NeroHW1 and see java and text files you wish to submit.
4. If 1, 2, & 3 succeeded, success! Submit NeroHW1.zip

Grading Rubric

20 Points	Style: Comments and Indentation ¹
80 Points	Functionality: <ul style="list-style-type: none">• Program compiles (10)• Program runs (10)• For given inputs (Numbers 1-4), program produces correct output (60)

- If the submitted program does not compile: 0 of 80 points
- If the submitted program compiles but does not run: 10 of 80 points
- If the submitted program compiles and runs: 20 of 80 points
- If the submitted program compiles, runs, and produces correct output: 80 of 80 points

The correctness of your program will be evaluated using test cases developed by the instructor.

Late assignments will not be accepted – no exceptions (please do not email me your assignment after the due date, I will ignore it).

Please feel free to setup an appointment or drop by during office hours to discuss the assigned problem. I'll be more than happy to listen to your approach and make suggestions.

¹ Standards for documentation and indentation will be posted separately.