Program #2 - Word freq

Prof Bill - Feb 2020

Program #2 logistics:

- Due: **Fri Feb 28, 2020** at the beginning of class (3 weeks)
- Worth: **8 points** (8% of your grade)
- Learn: binary search tree, O(log n), JCF (ArrayList, HashMap), Java file I/O, Java string parsing, benchmarking, debugger, Javadoc, and more!

1. Description

Question: How many times do the words “war” and “peace” appear in the novel War and Peace by Tolstoy?

Answer: Let’s find out with Program #2.

In program #2, we’ll read a book and then tally the frequency of the words in the book. We’ll do this using 3 data structures: ArrayList, HashMap, and our own binary search tree (BST). What performance does Big-O predict for each data structure? Let’s run some examples and compare!

Program #2 runs in the console again. (don’t worry, graphics are coming soon).

thanks… yow, bill

2. Design discussion

The book data will be plain text. My book source is: www.gutenberg.org and www.gutenberg.org/browse/scores/top. I'll probably grab a bunch of examples and copy them to the k: drive. I'll also create a tiny example for you to start with initially.

Note: I was thinking of reading the books directly from the Gutenberg website, but it looks like there's extra info before and after the book that we don't want to count. So...more research?

The steps will be something like this (I think):

```
open the book file
init word freq object
while not empty {
  read a line from the file
  for each word in the line {
    strip the word of blank space and punctuation
    increment word count in word freq object
  }
}
print or save your results
close the file
```

He have some fun problems to solve:

- ➢ How do we read a text file in Java? One word at a time.
- ➢ Would it be better (more fun) to read files from the internet?
- ➢ How can we process these words, so that upper/lower case, whitespace, and punctuation are removed?
- ➢ Let's create a Binary Search Tree interface for class. (Bst210?) Then, we can share test code. (cool)
- ➢ What should we be storing in our structures (ArrayList, HashMap, Bst210) to track word frequency? Is this another interface: WordFreq210?
- ➢ How can we compare results amongst ourselves? This is a toughie. Define a file format? Write a Word freq checker? I don't know. Idea: Save the top 100 to a file. Just check those. Or, words used more than 10 times or something. Brainstorm!

- ➢ I'll be running alongside you with a little help: Program #2 Helper (gdoc). Idea: What about one of you running the Helper?

We'll work on some of these problems together in class. Of course, if you get stuck, then you can just...email me!

3. Requirements

Program #2 requirements are:

- ➢ Write your program in **Java**.
- ➢ I will only accept **quality code**: Java coding guidelines
- ➢ Use **VS Code** to edit/debug your program. And use the debugger to help your problem-solving. That's the pow-ah!
- ➢ Code your own **binary search tree**. Use pseudo code from my notes and other sources, not Java code.
- ➢ Add one **creative element** to your solution.

How to succeed (writing any program):

1. Start early!
2. Don't be shy. Ask a question in class. Email me. Come to office hours.
3. Small bites. Divide and conquer your program into small, manageable tasks.
4. ABW. Always be working. Your program should always compile and run. Never leave your work in disarray.

4. Grading

To submit your work, create a **program2** folder on your k: drive.

This folder should contain:

- All your Java source files
- Your executable class files (compile here in the lab)
- Any results files
- A **README.txt** file...that follows my template.

Remember our **plagiarism** guidelines as well. Getting help from google or stackoverflow or a friend is OK, but:

1. You must acknowledge any help you receive with a comment in your code
2. You must understand any code in your solution
3. Get help on program components, not the assignment (the tic tac toe philosophy)
4. If you have any questions in this area, contact me **before** you turn in your work, not after (when it's too late)

thanks… yow, bill