

Programming Assignment 6

Due: 11:59pm, Saturday, February 13

Overview

The goals of this assignment are to:

1. Use file I/O
2. Use arrays and collections
3. Think about runtime

Setup

Open a new Linux terminal window. In your home directory, create/make a new directory called `HW6` and change directory into that directory:

```
$ mkdir ~/HW6
$ cd ~/HW6
```

There are no starter files for this project, but there are test input files and validation output files.

Part I: Reading (10 pts)

Read the following articles and answer the following questions in a file named `PART1` (note: not `part1.doc` or `part1.txt`. The bundle script will only accept a file named `PART1`):

1. [Hashing](#)
2. https://en.wikipedia.org/wiki/Big_O_notation
3. https://en.wikipedia.org/wiki/Sorting_algorithm

Q1: What are the pros/cons of using arrays?

Q2: Briefly explain what an `ArrayList` is and list their pros/cons.

Q3: Briefly explain what a `HashSet` is and list their pros/cons.

Q4: What is the big-O runtime for insert sort? For QuickSort?

Q5: Why does a sort have to be at least $O(n)$?

Part II: Word cloud (30 pts)

I want to write a poem. One that I know I'd like, so I've decided that the poem should contain words from other poems that I like. I've collected a sample of poems and put them into the file:

```
/home/linux/ieng6/cs11wb/public/HW6/poems.txt
```

I now want you to create a program, `WordCloud.java`, that can read in a text file and print out the most commonly used words in the file. Once done, my job of writing a new poem will be easier because I'll know which words I should definitely use.

Based on [Rick Ord](#)'s problem set 2.

Implementation details:

- Words occurrence count is case insensitive (River, river, rivER would all count as the same word).
- I'm not interested in how many times common words like "the, a, an, ..." (also case insensitive) show up, so your program will also need to read in the text file:
`/home/linux/ieng6/cs11wb/public/HW6/common.txt`
And make sure these common words are not counted and don't appear in the resulting output
- The program will take two arguments: the text file to read in and the number of words (N) to print out.
- The output of the program will be the top N words along with the number of times each word occurred in the text.
- If the user requests the top 100 words, but there are only 50 unique words in the input file, then the top 50 words should be printed.
- You'll need to remove any ",!?:-" from the end of strings that you read, otherwise "time" and "time." will be counted separately.
- If multiple words have the same frequency, you can print them in any order
- If the user does not enter the file name to read, or the number of words to report, the program should state how to use the program (example below):

Example usage:

```
$ java WordCloud
Usage: java WordCloud <file name> <#words>
```

```
$ java WordCloud ~/../public/HW6/poems.txt 8
your : 10
time : 8
against : 5
like : 5
kansas : 4
rising : 4
sun : 3
never : 3
```

Now I know that my poem should include the above words!

Once you get that to work, test your solution on the much larger file,

```
$ java WordCloud ~/../public/HW6/allbooks.txt 200
```

And make sure your program prints out an answer within 30 seconds. If not, then you need to rethink your algorithm and come up with a faster implementation. As a check that your program is working correctly, I've uploaded the files poems_8.out and allbooks_10.out to the public directory.

Based on [Rick Ord](#)'s problem set 2.

POWeek Challenge

Once you get your program to work, you'll know the top words that an author likes to use. What makes this interesting is that we can extend the program just a little and actually produce a TL;DR summary of any given input. Check out <http://smmry.com/>, and give it an example URL to read and summarize. The programmers outlined their algorithm at <http://smmry.com/about>. You'll notice that it relies on counting how many times a word shows up in the text.

Try to implement your own summarizer using the same logic and make it run as fast as possible. In our solution when developing the POWeek challenge, we skipped steps 1 and 4 of smmry's algorithm, but you can take on if you'd like the challenge. Also, instead of specifying how many words to print out, your program should take as input the number of sentences to print. We'll be comparing how well your summary works **and** how fast you get to your solution. You can find the time of your program execution via:

```
$ time java POWeek <file> <#lines>
```

Style Requirements (10 pts)

You will be graded for the style of programming on this assignment.

- Use reasonable comments to make your code clear and readable.
- All methods must have javadoc comments. We will be testing this by running "javadoc filename.java" and ensuring that the resulting documentation pages appear.
- Use reasonable variable names that are meaningful.
- Use static final constants to make your code as general as possible. No hardcoding constant values inline (no magic numbers).
- Judicious use of blank spaces around logical chunks of code makes your code much easier to read and debug.
- Keep all lines less than 80 characters. Make sure each level of indentation lines up evenly.
- Every time you open a new block of code (use a '{'), indent farther by 2 spaces. Go back to the previous level of indenting when you close the block (use a '}').
- Always recompile and run your program right before turning it in, just in case you commented out some code by mistake.

Turnin Instructions – Different from before

Remember the deadline to turn in your assignment is Saturday, February 13 by 11:59pm. We have been running into our quota limits on the ieng6 servers so we have to reduce the amount of data that you turn in. Make sure you do not submit any files other than those requested. If you create additional java classes, those are fine to submit, but if allbooks.txt is in your HW6 directory be sure to remove it before you submit. I'll make a note of this in the rubric as well.

When you are ready to turn in your program in, type in the following command and answer the prompted questions:

Based on [Rick Ord](#)'s problem set 2.

```
$ cd ~/
$ bundleP6
Good; all required files are present:

    HW6

Do you want to go ahead and turnin these files? [y/n]y
OK.  Proceeding.

Performing turnin of approx. 6144 bytes (+/- 10%)
Copying to /home/linux/ieng6/cs11wb/turnin.dest/cs11wb.P6
...
Done.
Total bytes written: 31744
Please check to be sure that's reasonable.
Turnin successful.
```