Programming Assignment 4

Due Nov 2, 2019 by 11:59pm **Points** 60 **Submitting** a file upload **File Types** zip

Programming Assignment 4 will consist of writing multiple short programs using material from modules 5, 7, and 8. **For this assignment, are allowed to use the editor of your choice (e.g. Eclipse, IntelliJ, VS Code, Atom).**

Each problem below should be self contained within its own folder. You will upload these folders to your GitHub site. Each folder must contain the java source code in its proper package structure. All problems should be in the *same* repository. To keep things simple, I suggest that you create a new repository on GitHub called **<last_name>_PA4** and push your projects there.

You will submit your assignment as *both* a link to your GitHub page and an upload of your projects as a *.zip file. Your **projects should be named <last_name>_pN**, where N is the problem number the corresponds to each project. You must **name your zip file <last_name>_PA4.zip**. Failure to adhere to these naming convention may result in your assignment going ungraded. Due to Webcourse@UCF limitations, when you submit your assignment, submit the zip file. Add your GitHub URL as a note during the submission, or as a comment after the submission. If you do not know how to create a zip file, refer to Google (for windows, just look **on microsoft's website** (https://support.microsoft.com/en-us/help/4028088/windows-zip-and-unzip-files)).

If you forget to submit one or both of the required items on time, additional submissions after the deadline will be considered late for the purposes of calculating your grade. **NO EXCEPTIONS WILL BE MADE**.

An example structure of your GitHub repository might be:

```
hollander_PA4
  |--- hollander_p1
  |--- hollander_p2
  |---      :
  |--- hollander_pN
```

Grades for this program will be determined according to the rubic. If your project on GitHub does not contain the proper package structure, it will not be graded. This requirement is in place so that my graders can either pull your code directly from git or unzip it and run it without modification.

Problem 1

Create a class called **DuplicateRemover**. Create an instance method called **remove** that takes a single parameter called **dataFile** (representing the path to a text file) and uses a Set of Strings to eliminate duplicate words from **dataFile**. The unique words should be stored in an instance variable called **uniqueWords**. Create an instance method called **write** that takes a single parameter called **outputFile** (representing the path to a text file) and writes the words contained in **uniqueWords** to the file pointed to by **outputFile**. The output file should be overwritten if it already exists, and created if it does not exist.

Create a separate class called Application that contains a main method which illustrates the use of **DuplicateRemover** by calling both the remove and write methods. Your input file must be called **problem1.txt** and your output file must be called **unique_words.txt**. You will not receive credit if you use different file names, as my graders will not modify your code to make it work with their test files.

Your program should work on any text file. The TA's will provide their own version of problem1.txt when they run your code.

Problem 2

Create a class called **DuplicateCounter**. Create an instance method called **count** that takes a single parameter called **dataFile** (representing the path to a text file) and uses a Map of Strings to count how many times each word occurs in **dataFile**. The counts should be stored in an instance variable called **wordCounter**. Create an instance method called **write** that takes a single parameter called **outputFile** (representing the path to a text file) and writes the contents of **wordCounter** to the file pointed to by outputFile. The output file should be overwritten if it already exists, and created if it does not exist.

Create a separate class called Application that contains a main method which illustrates the use of **DuplicateCounter** by calling both the remove and write methods. Your input file must be called **problem2.txt** and your output file must be called **unique_word_counts.txt**. You will not receive credit if you use different file names, as my graders will not modify your code to make it work with their test files.

Your program should work on any text file. The TA's will provide their own version of problem1.txt when they run your code.

Assignment 4 Rubric

Criteria	Ratings		Pts
Problem 1: Problem 1 shall contain a public class called DuplicateRemover	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: DuplicateRemover shall contain a method called remove	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: Given a text file, dataFile, the remove method shall determine the unique words contained in dataFIle and store those unique words in the associated DuplicateRemover object	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: The remove method shall terminate the program and alert the user when an exceptional IO event occurs	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: The remove method shall clean up any and all resources allocated during method execution	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: DuplicateRemover shall contain a method called write	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: Given a text file, outputFile, the write method shall print the current collection of unique words to outputFile	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: The write method shall terminate the program and alert the user when an exceptional IO event occurs	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: The write method shall clean up any and all resources allocated during method execution	**2 pts Full Marks**	**0 pts No Marks**	2 pts

Criteria	Ratings		Pts
Problem 1: Problem 1 shall contain a public class called Application	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: Application shall contain a main method	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: The main method shall create an instance of a DuplicateRemover called duplicateRemover	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1: The main method shall use the write method of duplicateRemover to output the unique words of "problem1.txt" to a file called "unique_words.txt"	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: Problem 2 shall contain a public class called DuplicateCounter	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: DuplicateCounter shall contain a method called count	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: Given a text file, dataFile, the count method shall determine the number of occurrences of each word contained in dataFIle and store each unique word and its count in the associated DuplicateCounter object	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: The count method shall terminate the program and alert the user when an exceptional IO event occurs	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: The count method shall clean up any and all resources allocated during method execution	**2 pts Full Marks**	**0 pts No Marks**	2 pts

Criteria	Ratings		Pts
Problem 2: DuplicateCounter shall contain a method called write	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: Given a text file, outputFile, the write method shall print the current collection of unique words and their counts to outputFile	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: The write method shall terminate the program and alert the user when an exceptional IO event occurs	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: The write method shall clean up any and all resources allocated during method execution	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: Problem 2 shall contain a public class called Application	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: Application shall contain a main method	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: The main method shall create an instance of a DuplicateCounter called duplicateCounter	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 2: The main method shall use the write method of duplicateCounter to output the unique words of "problem2.txt" to a file called "unique_word_counts.txt"	**2 pts Full Marks**	**0 pts No Marks**	2 pts
Problem 1 has been correctly uploaded to GitHub	**4 pts Full Marks**	**0 pts No Marks**	4 pts

Criteria	Ratings		Pts
Problem 2 has been correctly uploaded to GitHub	**4 pts Full Marks**	**0 pts No Marks**	4 pts
Total Points: 60			