CS 284: Homework Assignment 2
Due: February 25, 11:55pm

1 Assignment Policies

Collaboration Policy. Homework will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. Excerpts of code presented in class can be used.

Your code must include a comment with your name and section .

2 Assignment

This assignment will use a dictionary file (ionDictionary.txt) that we created from the book "Ion" by Plato. This assignment requests you to create a program that provides a menu to the user to perform the following functionalities:

- Menu Item 1: print only list of words that appeared in the book (stored in ionDictionary.txt)
- Menu Item 2: take a word as an input from the user and print out how many times the word appeared in the book using the dictionary file
- Menu Item 3: quit the program

Define two classes `DictionaryItem` and `Dictionary` that reads from the file "IonDictionary.txt" which has been created from the book "Ion" by Plato.

DictionaryItem will be a container class where we store the word-count pair information. E.g. the word 'you' appears 135 times in the book. DictionaryItem class will hold this information: 'you' as the word and 135 as the count.

Dictionary class has multiple functionalities, listed below.

2.1 Basic operations

The following operations must be supported for DictionaryItem:

- Two data fields: word and count.
- A constructor `DictionaryItem(String word, int count)` for initializing the data fields: word and count.
- Getter and setter methods for the data fields in DictionaryItem: word and count.

The following operations must be supported for Dictionary:

- Two data fields: wordList arraylist that stores words and dictArrayList that stores DictionaryItems (word-count pairs).
- A constructor `Dictionary()` for initializing two arraylist data fields. For initial capacity, use the number 1300, since we know there exists 1223 words in the dictionary. Call readFile() method from constructor to fill out the arraylists.
- A constructor `Dictionary(String filename)` for initializing two arraylist data fields. For initial capacity, use the number 1300, since we know there exists 1223 words in the dictionary. Call readFile() method from constructor to fill out the arraylists.
- An operation `void printMenu()` for printing the 3 menu items as shown in example output file. In this method, we take a number from the user which indicates the operation user chooses. From this method, call the helper method processMenuItem(). Repeat printing the menu until user enters 3 to quit the program. Check if the user entered values between 1 and 3, if not print an error message and ask for input.
- A helper method `boolean processMenuItem(int menuItem, Scanner scan)` takes two inputs: operation that user chose (menuItem) and Scanner object to read the word from the user for searching a word in dictionary. It calls the appropriate functions for each operation.
- An operation `readFile(String filename)` method to store words in wordList and word-count pair in dictArrayList fields. Throw FileNotFoundException for the missing file in the folder.
- A helper method `splitStoreLine(Scanner scan)` to split the line read from the file and store the word-count pairs in the defined arraylist.
- An operation `printDictionary()` method that prints the word list that we created from the dictionary text file.
- An operation `searchDictionary(String word)` method calls the binarySearch() method to search the word in the wordList and using the index of the word in wordList, returns the count of that word from dictArrayList.
- An operation `binarySearch(String word, int low, int high)` helper method that performs the binary search algorithm on the sorted wordlist arraylist that we created. Do NOT use the built-in binarySearch methods, see the hints below for the pseudocode link.

2.2 Runtime Analysis of Your Code

You are expected to provide another file "RuntimeAnalysis.pdf" that shows the runtime of your algorithm.

You are only asked to calculate the runtime for the following methods:

- printDictionary()
- searchDictionary(): including the runtime of binarySearch() method.

Please provide the runtime information of these two methods in big-o notation and explain how you reached to that result. 1-page explanation is enough. You can use τ notation as a help to explain your analysis. It doesn't have to be very through (e.g. calculating runtime for every statement).

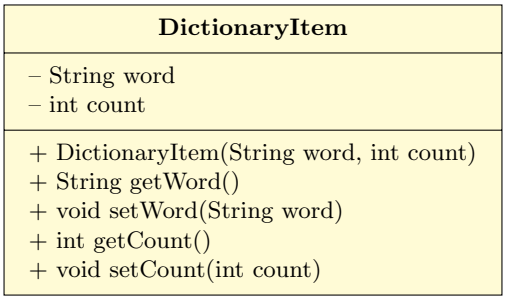
2.3 Hints

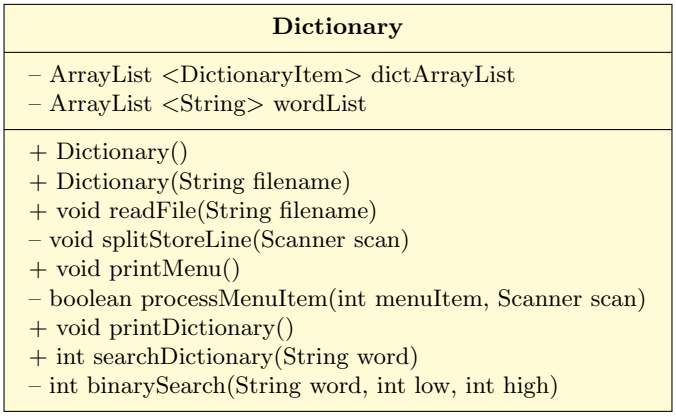
- For reading the file in the `readFile()` method, you will use pipe delimiter to split the sentence into word-count pairs. Since pipe symbol is used as 'or' logical operator, you need to use an escape character. Please check out this link to solve that problem: `https://www.javatpoint.com/how-to-split-a-string-in-java-with-delimiter`.
- Binary Search method needs to be implemented by you, do NOT use the built-in methods (like Arrays.binarySearch() or Collections.binarySearch(). Please see the implementation ideas on this page, you can both use iterative-recursive approaches: `https://www.baeldung.com/java-binary-search`. You will lose points if you use built-in methods.
- When taking input from the user, it might be easier to use nextLine() method, instead of using nextInt() and nextLine() together. You are free to do any implementation you like, but it is easier to take the input as a string.

3 Submission instructions

Submit three files in total. Two Java files named `Dictionary.java and DictionaryItem.java` and runtime analysis report, a pdf file is required to be submitted through Canvas. Your grade will be determined as follows:

- You will get 0 if your code does not compile.
- The code must implement the following UML diagram precisely (see below), but you are allowed to create more helper functions if they are needed.
- We will try to feed erroneous and inconsistent inputs to all methods. All arguments should be checked.
- Partial credit may be given for style, comments and readability.


DictionaryItem
– String word
– int count
+ DictionaryItem(String word, int count)
+ String getWord()
+ void setWord(String word)
+ int getCount()
+ void setCount(int count)


Dictionary
– ArrayList <DictionaryItem> dictArrayList
– ArrayList <String> wordList
+ Dictionary()
+ Dictionary(String filename)
+ void readFile(String filename)
– void splitStoreLine(Scanner scan)
+ void printMenu()
– boolean processMenuItem(int menuItem, Scanner scan)
+ void printDictionary()
+ int searchDictionary(String word)
– int binarySearch(String word, int low, int high)