ICS 202 Lab Project – Term 212

**Submission deadline**: **11:59 pm on Sunday, 24th April 2022**

**Instructions**: Submit in your ICS 202 Lab section Blackboard the .java files for the data structure used, the test program, the complete WordPair class, the dictionary.txt file and the report word file in a zip file named in the format:

YourKFUPMID#\_FamilyName\_LabSection#\_LabProject.zip

**Note**:

* Don’t submit .exe or .class files.
* Remove all package statements in your .java files
* Each student is required to do the project on his own. Copying from another student or from the internet any portion of the project will result in a grade of zero for all involved students.
* YOU ARE ONLY ALLOWED TO USE APPROPRIATE DYNAMIC DATA STRUCTURE CLASSES THAT ARE PROVIDED IN ICS 202 LABS OF TERM 212 BUT NOT BINARYHEAP, HASHTABLE, B- OR B+ TREES. Using Java data structure classes such as ArrayList, LinkedList, Vector, TreeSet, TreeMap, HashMap, etc. is not allowed. The only such class that is allowed is the **String** class together with all its methods.
* Make sure to write a report file that briefly explains your code (different classes and their methods). Give the Big-O complexity of each of your methods in the dictionary as a function of *n* where *n* is the number of words in the dictionary. Give any additional notes about your program. You must also state which methods are not working in your project.

Each line of a text-file **dictionary.txt** contains a unique word followed by the meaning or meanings of that word. Each word is a one-word string, the meaning or meanings of a word can be a string of one or more words and it may contain punctuation characters. The dictionary is case-insensitive.

Write a Java program that implements a dictionary by reading, each time the program starts, the text-file and initializing an **efficient** data structure with the words and their meanings. After the initialization all dictionary operations must be performed on the data structure and not on the text-file. Before your program exits, it must update the text-file if the dictionary has been modified in the data structure. This means your dictionary must be persistent.

You may find the following class useful in the project implementation:

**public class WordPair implements Comparable<WordPair> {**

**private String word;**

**private String wordMeanings;**

**// . . . to be completed by students**

**}**

Your data structure must support the following operations that are implemented as a menu driven console program:

|  |  |
| --- | --- |
| method | description |
| public boolean isEmpty() | returns **true** if the dictionary is empty, **false** otherwise. |
| public int getSize() | returns and displays how many words are currently stored in the dictionary. |
| public boolean insert (WordPair wordpair) | inserts word pair into dictionary if not already present. Returns **true** if the insertion is successful; otherwise, it returns **false** if the word exists in the dictionary. |
| public WordPair find(String word) | returns and displays a WordPair if the word is present; otherwise, it returns **null** and displays the message: "Word not in dictionary". |
| public boolean delete(String word) | deletes the wordPair associated with word if the word exists. It returns **true** if the word exists; otherwise it returns **false**. |
| public boolean modifyWord(String word, String newMeanings) | modifies a word to a new meaning, if the word exists in the dictionary. Returns **true** if the word exits; **false** otherwise. |
| public void printAll(String prefix) | prints all words that start with **prefix**, together with their meanings. Prints the message: "No word starts with this prefix" if no such word exists. |
| public void printSorted() | prints the words in the data structure together with their meanings in lexicographic order. |

Your program must use the menu shown below. It must run in a continuous loop until the option 7 is chosen:

1. Insert a new word with its meanings
2. Search for a word
3. Delete a word and its meanings
4. Modify the meanings of a word
5. Print all words with a given **prefix** and their meanings
6. Print the contents of the dictionary sorted in lexicographic order
7. Exit

Provide appropriate required behaviour for each of the above options.