COMP251 - Assignment 1
Deadline: Oct 15, 2018

Goal:

In this assignment you will design your very first ADT and implement it. You need to respect the *abstraction* principle in your implementation. Using the ADT you designed you will implement a dictionary of words[1].

Problem:

Each entry in the dictionary is a pair: (word, meaning). Word is a one-word string, meaning can be a string of one or more words. The dictionary is case-insensitive. It means "Book", "BOOK", "book" are all the same[2].

Your dictionary application must provide its operations through the following menu (make sure that you follow this format):

```
Possible operations:
f) find the meaning of a word.
i) insert an entry (a new pair of word, meaning).
l) load the dictionary from an input file.
r) remove an entry.
s) save the contents of the dictionary in an output file.
x) exit
Please choose an option (f, i, l, r, s, or x):
```

The user should enter one of the options (just one character f, i, l, r, s or x). Your program should act based on the user's option:

Find option:

The program asks the user to enter a word, then searches the dictionary and if it exists, prints the meaning of the word, otherwise prints:

```
Word not found.
```

[1] Later in this course we will see there are better ways to implement a dictionary of words.

[2] Simply convert all input strings to lowercase at first then work with them.

Insert option:

The program asks the user to enter a word. The program searches the word in the dictionary, and if it does not exist, it asks the user to enter the meaning of the word and it should insert the entry (word, meaning) into the dictionary.
Remark: the insert method must check that the word does not exist in the dictionary, if the word exists then it must ask if the user want to update the meaning or insert a new word. (for example print a message like this:)

```
The word you entered already exists in the dictionary.
Would you like to update the meaning (y/n)?
```

Load option:

The program asks the user to enter the address of a file containing a set of entries (e.g. “dictionary.txt”). It should open the file, read the entries and insert them in the dictionary.
Format of the file: each line is one entry. The first word in the line is the word and the rest of the line is the meaning (it can be just one word or more). Note that the words in the input file are random (not necessarily in alphabetical ordered).

Remove option:

The program asks the user to enter a word and the program searches the word in the dictionary and remove it, otherwise prints:

```
Word not found.
```

Save option:

The program asks the user to enter the address of a file to store the entries. The program should write the entries in the dictionary**.** Each line of the file contains one entry of the dictionary.

```
abide follow
boss head, chief
destroy demolish
domestic family
```

…

Exit option:

Terminates the program.

Requirements:

Your implementation must follow these requirements:

Sorted List ADT

You must have a class to implement "sorted list ADT", you should implement it using array as internal data structure with name

- ArrayBasedSortedList[3]

The implementation of your ADT should provide these operations:

- `void add(Comparable x)`: inserts the comparable object `x` at the appropriate position in the sorted list
- `boolean remove(Comparable x)`: finds an element in the list which is equal to the comparable object `x` and remove it from the list. The method return true if the items is removed from the sorted list and false otherwise (if the item does not exist to be removed).
- `Comparable get(int i)`: returns the object at the `i`-th position in the list.
- `Object find (Comparable x)` returns an object of the list which is equal to the comparable object `x`. The list is sorted so you can use binary search to implement this method.
- `int size()` : return the number of elements in the list

You might have other methods in the implementation of sorted list ADT, but you can only interact with the list through the operations above.

Guideline:

In addition to sorted list ADT, you need at least three other classes: one class for word entries (e.g `Word`), a class for the dictionary (e.g. `Dictionary`) and a test class containing a `main` method that just tests your dictionary application.
In class `Dictionary` you might have an object of sorted list ADT (e.g. `wordList`).

For example when you want to look for a word in your dictionary, you can do it as follows[4]: (Note that `searchWord()` is a method of class `Dictionary` which search in the list of

[3] The `ArrayBasedSortedList` is a dynamic structure (i.e the size of the list is not fixed and it can grow if needed).
[4] It is just a hint, you are not required to follow it.

words (`wordList` which is an object of `ArrayBasedSortedList`) to find a word and display the meaning of that word)

```
void searchWord(String word){
     Word temp_word = new Word(word, "");
     Word result = wordList.find(tem_word);
     if (result == null)
          System.out.println("Word not found.");
     else
          System.out.println(result);
}
```

Make sure you define methods `toString()`, `equals()` and `CompareTo()` for your classes as required.

Important Note: showing the menu of the options,... is a part of your dictionary application not testing! It is better to have a method in class `Dictionary` (eg. `run()`), that after creating an object of dictionary by calling `run()` your application starts working.

Specific requirements

Make sure to add a readme file that explains your code (different classes and their methods) and any additional notes about your assignment.

Important Remarks

It's very important that you make sure your program compiles without any problem (you may get 0 if your program doesn't compile). Your input and output file format should exactly follow the requirements stated previously.

Submission

Create a directory called src and copy all your source files and any readme file in it, zip it and submit it to the blackboard.