Joanna Klukowska
joannakl@cs.nyu.edu

Programming Project 1: Find Words

Due date: September 28, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own. Any sharing or copying of assignments will be considered cheating. If you get significant help from anyone, you should acknowledge it in your submission.

In this project you will work on the program that produces all possible words given a set of letters and a dictionary. The basic version of your program should only produce words that use all the letters from the set. For example, if the letters are

```
atrac
```

your program should print in alphabetical order all different words that can be made out of those letters (assuming that they are present in the dictionary that the program uses):

```
arcat
carat
carta
catar
```

For extra credit, you can modify the program to display all words with lengths 2 up to the number of letters from the set. For example, if the letters are

```
atrac
```

your program should print in alphabetical order all of the following words (again, assuming that they are all in the dictionary):

```
acta
arcat
at
car
carat
carta
cat
catar
rat
ta
```

The Program Input

Input File Your program is given the name of the text file on its command line. The text file contains a dictionary that is used by the program. You may assume that the dictionary contains a sorted list of words, one per line. A word is any sequence of letters.

If the filename is omitted from the command line, it is an error. If the filename if given but it does not exist or cannot be opened for reading by the program, for any reason, it is an error. The program should display an error message if any of these conditions occur, and it should be as specific as possible (it is not enough for it to say "could not open file"). Your program is NOT ALLOWED to hardcode the input filename in its own code.

Your program should read in all the words from the input file and store them in a suitable way for later use. Your program should read the input file only ONCE.

Joanna Klukowska
joannakl@cs.nyu.edu

User Input The user should be prompted to enter a string of 2-10 characters (letters only, no spaces, commas, or any other characters). Your program should accept both upper case and lower case letters. If the user enters any characters other than letters, it is an error. If the user enters too few or too many characters, it is an error. The program should display an error message if any of these conditions occur, and it should be as specific as possible.

If the user enters any uppercase letters, your program should convert them to lowercase before proceeding.

Computational Task

Once the user has entered the letters, the program displays all the words in the dictionary that can be formed as combinations of *all* the letters entered by the user. [If you are implementing the extra credit, your program should display all the words in the dictionary that can be formed as combinations of *any subset* of the letters entered by the user.]

Program Design

Your program must contain at least three classes:

- one to represent the Find Words "game",
- one to represent the dictionary of all the words read in from the input file (Note: depending on your own design you may end up creating multiple dictionary objects with subsets of words in them),
- one to represent the set of letters entered by the user.

The operations that check if a given word is in the dictionary, should be performed by the dictionary class. The operations that "create" different words based on the set of letters should be part of the class that represents the set of letters. The operations related to reading in the input (both file and user) and displaying the results should be part of the Find Words game class itself.

Creating the Possible Words

The task of creating possible words should be achieved recursively using the backtracking technique.

If the user enters n letters there are $n!$ different possible words (note that some of them might repeat, if some of the letters repeat) - but not many of them are going to be found in the dictionary. You should design an algorithm that tries different combinations of letters and checks if they are in the dictionary. But it should do it in a "smart" way (do not try words that cannot possibly be in the dictionary).

Here are a few observations that might help you design the algorithm:

- If you do no implement the extra credit part, then the only words in the dictionary that matter are the words whose length matches the number of letters that the user entered.
- If your letters are: `a t r a c` and your program determines that there are no words in the dictionary that start with `ct`..., then there is no point in trying to create any words with that prefix.

Programming Rules

You should follow the class design rules outlined in lecture 1 notes.

You must document all your code using Javadoc. Your class documentation needs to provide a description of what it is used for and the name of its author. Your methods need to have description, specification of parameters, return values, exceptions thrown and any assumptions that they are making.

You must use recursion and backtracking to produce your list of words.

Joanna Klukowska
joannakl@cs.nyu.edu

You may use ArrayList and String classes and any methods that are provided in them. You may use any classes for reading the input from the file and from the user. You may use any exception related classes.

You may use the sort method provided in the Arrays and/or Collections class.

You must implement your own search method using recursion. You may NOT use any of the search methods implemented in the Arrays and/or Collections classes.

Working on This Assignment

You should start right away! This program does not require you to write much code (well, more than you are used to from cs101, but less than future assignments), but it will take time.

You should modularize your design so that you can test it regularly:

- Start with a program that reads in the input file and the input from the user. Make sure that this works.
- Write a class to represent and store the dictionary. Develop methods that allow you to search in the dictionary (HINT: you will need two search methods: one that looks for words, the other that checks prefixes). Make sure that this works.
- Write a class that represents the set of letters with methods that, given a dictionary object, can produce a list of words (in an ArrayList, for example) that consist of all the letters (DO NOT try to implement the extra credit version right away!). Make sure that this works.
- Test, test, test, test. Use different dictionaries, different letters. Try to break your program.
- Save this version and make a backup of it.
- If there is still time, attempt the extra credit part. If you think it is working, test, test, test, test.
- Submit your code.

You should backup your code after each time you spend some time working on it. Save it to a flash drive, email it to yourself, upload it to your Google drive, anything that gives you a second (or maybe third copy). Computers tend to break just a few days or even hours before the due date - make sure that you have working code if that happens.

Grading

If your program does not compile or crashes (almost) every time it is ran, you will get a zero on the assignment.

15 points	reading, validating, processing the input (both file and user)
15 points	design and implementation of the Find Words ”game” class
25 points	design and implementation of the dictionary class (including the recursive implementation of the method(s) that search the dictionary)
25 points	design and implementation of the set of letters class (including the recursive implementation of the method that constructs potential words)
20 points	proper documentation
15 points	extra credit part

How and What to Submit

Your should submit all your source code files (the ones with .java extensions only) in a single **zip** file to NYU Classes. You should have at least three different files.