Project #3

Late Policy: 24-hour late period, then zero

Instructions: This is an individual assignment. All work should be your own.

Objectives:

Use a hash table to solve a word puzzle.

Description:

The word puzzle is described in section 1.1 in the textbook and discussed

further at the end of chapter 5.

A grid consisting of letters is to be checked against a dictionary of words

to see if the grid contains any of the words.

The user can input a value for the rows and columns of the grid and the program

will create a grid of random characters. The number of rows and columns can

range from 10 to 40, and the grid does not have to have the same number of rows

and columns.

The program will read in a dictionary file (provided) and use an algorithm to

solve the word puzzle. You can keep the largest word size for use in solving

the puzzle. Words containing punctuation can be omitted.

The hash table should be the QuadraticProbingHashTable from the textbook without

any modifications to it.

The program should use the algorithm described as the "second" algorithm in 1.1,

which checks each string in the grid for presence in the dictionary.

The user should also have the option of using the following enhancement:

When reading the input file of words, store each prefix of the word as well.

For example, if the word is "apple", store "a", "ap", "app", "appl", "apple".

In the algorithm, if a prefix is not found, the rest of this string can be

treated as "not found". For example, if the string is "apbum", and after

checking and finding "a" and "ap" I find that "abp" is not in my dictionary,

then there is no point in checking further in this direction. For this you

should use Java's HashMap class to map the string to its type.

The program should output the elapsed time to find the words and a sorted list of

found words (omitting duplicates and single-letter words). Do not include the

time to load the table.

Submit to eLearning:

WordPuzzle.java