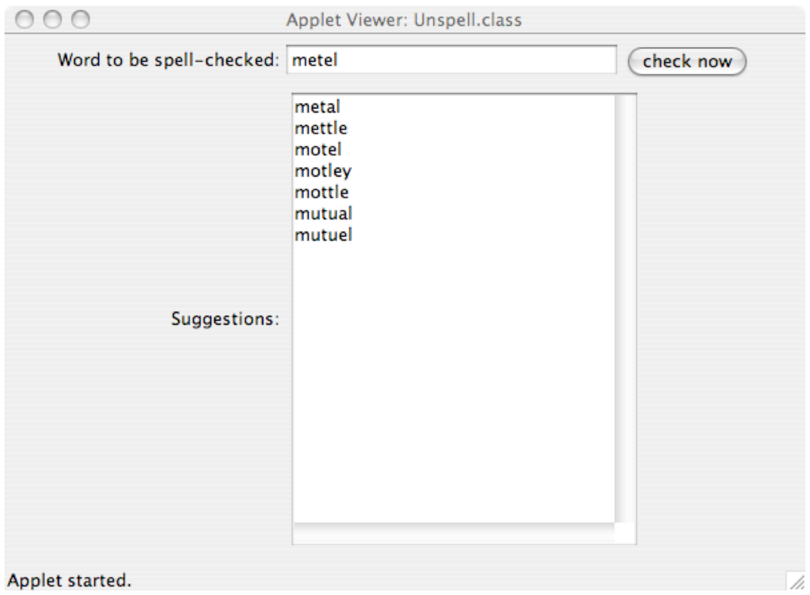
CS 51 Homework Laboratory # 11
A Spelling Checker

Objective: To give you an introduction to strings, and to gain insight into how spelling checkers perform some of their tasks.

You may work in pairs on this project and turn in only one program (and design) with both names on it.

—

Most spelling checkers on the market today not only tell you which words you misspelled, but they also offer alternatives for how you might have spelled the word in question. Detecting which words are misspelled is relatively easy: you need only look up each word in the document in a list of ”correctly spelled” words (called a dictionary). Once you know that a word is misspelled, however, a more difficult task arises finding an alternative. Commercial products do this in a variety of ways, but we will examine only one in this lab.



metel

metal
mettle
motel
motley
mottle
mutual
mutuel

Applet started.

One of the most common ways in which people misspell words is by getting the vowuls wrong. A second comon error occurs when people either use a single consonnant when a double is required or vice versa. Our ”spell checker” will not be fooled by these types of errors!

It works in the following way:
For each word, we ”strip all the vowels and compress all pairs of consonants” to get a new word. When we are asked for alternative spellings, we simply list all words which have the same consonant pattern as our query, as in the example below:

- Suppose that we are given the misspelled word ”bel”.
- When we ”compress” bel, we get ”bl”.
- When we look up ”bl” in our list, we see that a large number of correctly spelled words generate this pattern: ball, bell, boil, and bull, among others. *Important: Be careful about artificially*

creating double consonants that might be deleted. Consider the words "rear" and "rare" and the fact that they should be compressed to "rr".

- Another common spelling arror is to get the leading vowel incorrect. Thus, our compression scheme converts leading vowels to 'a's. Thus both "effect" and "affect" compress to "afct" whereas "fact" compresses to "fct". (For purposes of this assignment, 'y' IS a vowel.)

Your goal in this assignment is to write such a spelling checker. To make it easier for you to interact with it, you should design a class `SpellingDictionary`. I have provided part of the code for the constructor for this class. It will read in all of the words from a dictionary file into a one dimensional array. You will need to add (at least) the following methods:

`String getSize()` *Returns the number of words in the dictionary*

`void reset()` *Resets the dictionary so the next entry returns (see the next method) is the 1st of the dictionary*

`String getNextEntry ()` *Returns the next word in the dictionary. It should return null if there are no entries left.*

`boolean atEnd()` *Return true if no more words in dictionary.*

You will also need to write a method `canonize` that compresses a word according to the algorithm described above.

You should write this assignment in three phases:

1. Create a GUI interface for your program that includes a text field to enter words, a "spell check" button, and a text area to write all of the possible matching words.
2. Write and test your canonize function to make sure it returns the right values. We suggest you use an auxiliary function, IsVowel, which takes a character and returns true if the character is a vowel.
3. Modify the program so that when you enter a word, it prints all of the words in the dictionary that have identical canonized form.
4. Write a program that determines if the word input is spelled correctly. If the word is spelled correctly then print a message to that effect. If it is misspelled, then print a list of all of the words in the dictionary which have identical canonized form.

Be sure to test your canonize function before inserting it into your previous program. Your first try will almost assuredly be wrong, so write a testing program which just reads words and prints out their canonized forms. Only when you are convinced your function works should you move on to the next part. Try words like apply, oops, ballad, baled, you, rear, bubble, etc.

Notes:

1. *Important:* The dictionary that we have given you is held in order by canonized form. Thus if you are searching for a word and you find a dictionary word whose canonized form is larger than that of the word you are searching for, you are guaranteed the word will not occur later and you should stop searching for it.
2. Entries in the dictionary are entirely in lower case. Your program should find matches whether or not the input word contains capitals.
3. A character is not automatically treated as if it were a string. You can convert a character to a string using `Character.toString(c)` where `c` is a character or (*cheating and*) writing `""+c`.

4. There are two dictionary files available for your use in the start-up directory. They are named `word-tab` and `big-word-tab`. When your program starts, a window will pop up and you must navigate to the directory containing those files (the same directory that contains your program).

As usual we ask you to come to lab with a good design for your program. Remember that your designs are to be nearly as detailed as your final program. The ideal is for you to solve all logical design issues before you come into the lab, and use the lab to focus on Java issues and correct any defects in your design. The lab is not designed for you to figure out how to solve the problem, it is to test your design!

Extra Credit Real spell checkers have a suggested correct spelling. Improve your program by selecting one of the matches as the most likely word. The better your selection the more extra credit will be awarded.

Submitting Your Work When your work is complete you should deposit in the appropriate dropoff folder a copy of the entire `Lab11-Unspell` folder. Before you do this, make sure the folder name includes the phrase "Lab 11" and your name. Also make sure to double check your work for correctness, organization and style. This assignment is due Wednesday at 11 PM.

Table 1: Grading Guidelines

Value	Feature
	Design preparation (4 pts total)
1 pt.	SpellDictionary class
2 pt.	canonize method
1 pt.	actionPerformed
	Readability (5 pts total)
2 pts.	Descriptive comments
1 pts.	Good names
1 pts.	Good use of constants
1 pt.	Appropriate formatting
	Code Quality (6 pts total)
1 pt.	conditionals and loops
2 pt.	General correctness/design/efficiency issues
1 pts.	Parameters, variables, and scoping
2 pts.	Good correct use of arrays
	Correctness (5 pts total)
2 pt.	Canonize code
1 pt.	Detecting matches
2 pt.	Reporting correct results