Laboratory 6: The Unspeller

Objective. To gain experience with String manipulations.

Discussion. Because names can be spelled in a variety of ways (Bailey, Baily, Bailly Bailie, etc.) is useful to come up with an organized approach to reducing a name to its constituent sounds. Once reduced, the distilled form of the word can be used as a key or locator, say in a census. If vowels and double letters are dropped, for example, each of the spellings of Bailey would be found under the entry “bl” (along with Bela and Boyle). The National Archives makes extensive use of the Soundex system. In this system the first letter and the first three consonants form a four symbol key (see Problem ??).

Our approach, in this lab, is to write a method that reduces the word using the following process:

1. all double letters are reduced to one, then
2. if the first letter is a vowel, it is converted to an exclamation point (’!’), then
3. all remaining vowels are removed.

Thus apple becomes !pl; bible, bubble, and babble become bb1; eye and I become !; and label and lable become lb1. This last example demonstrates a potential use for the function: if a word is misspelled, alternatives can be derived from a dictionary by printing all correctly spelled words with similarly reduced forms.

In this lab, you are to write two functions: isVowel and reduce. isVowel is a method that takes a character and returns true if the character is a vowel. You may assume that the character is lower case, and ‘y’ should be considered a vowel. The reduce method takes a word (represented as a lowercase String) and returns a String that has been fully reduced according to the rules above.

Procedure. Perform the following steps while completing this lab:

1. Write the method isVowel as elegantly as possible. (Hint: one technique uses the indexOf method of Strings.)
2. Test isVowel fully before going onward.
3. Write the method reduce. Be aware that if letters are removed from a String, it becomes shorter; this may confuse poorly designed loops. One approach might be to accumulate the result in a second String.
4. Test reduce fully before going onward. Does it work on cases discussed above? How about llama and squill?
5. If you are so inclined, you may download the Undict class from the book’s web site. When included in the project along with a list of words in a text file called dict (any word list will do), the following code may be used to print out words in dict that sound similar to the String s:
ConsoleWindow c = new ConsoleWindow();
Undict d = new Undict();
String s;

do
{
    s = c.input.readString();
    c.out.println(d.like(s));
} while (!s.equals("quit"));

The program stops when the word quit is typed at the console. To verify the correctness of your reduce function further, correctly spelled words from the dict file should appear among the words found in Undict when typed at the keyboard. If they do not, there is a problem with your reduce method.

Thought questions. Consider the following questions as you complete the lab:

1. What words reduce to !ck?
2. Why is the leading vowel rule affective?
3. Outline three improvements to reduce that make use of various sound ambiguities in English. Provide examples of misspellings whose identification depends on your improvement.