CSC 207 **Assignment 5: Using Collections** **Fall 2009**

Due: Tuesday 3 March 2009

Topics: I/O, Collections

Collaboration: This homework assignment may be completed individually or in pairs.

Submission: Follow the instructions for submitting programs via P-Web and handing in a printed copy. Be sure to generate test cases and a statement justifying their sufficiency. One only one submission (paper and digital) per group is necessary.

You now have some experience using the Java Collections API, which features many of the data structures we'll be taking a close look at later in the semester. For now, we'll practice *using* them.

1 Soundex

The U.S. Constitution stipulates that census records must be gathered every 10 years. These records are managed by the National Archives. To help researchers using the archives, they have invented an algorithm for matching names by sound, rather than spelling. According to Wikipedia:

> Soundex is a phonetic algorithm for indexing names by sound, as pronounced in English. The goal is for names with the same pronunciation to be encoded to the same representation so that they can be matched despite minor differences in spelling. (`http://en.wikipedia.org/wiki/Soundex`)

While there are many alternatives that improve upon this early algorithm[1], Soundex is relatively simple to implement, so we shall use that.

Algorithm

Here is a description of the algorithm that takes an arbitrary name string and outputs its encoded representation

1. Replace all consonants (except the first letter, which is left as is) with a digit as shown below:

Digit	Letter
1	b, f, p, v
2	c, g, j, k,q, s, x, z
3	d, t
4	l
5	m, n
6	r
7	*all others*

   *Example: Paulsen→P774275, Rebelsky→*R7174227

2. Any repeated adjacent digits are reduced to a single digit.
   Example: P774275→P74275, R7174227→R717427

3. Remove any 7's.
   Example: P74275→P425, R717427→R142

4. Truncate to four characters total (one letter and three digits). If the sequence is already shorter than four characters, append zeros to make the result four characters long.

[1] It was patented in 1918—but we're on safe ground since it has long since expired. If you look at Figure 2 of the patent, it represents exactly what we're saying here! `http://www.pat2pdf.org/patents/pat1261167.pdf`

Assignment

You will write a program that takes a text file with a list of names as an argument, and builds a internal representation of the all the names, organized by their soundex code. Your program should be interactive and allow someone to query by name, being given the soundex, the number of names with the same code, and then a lexicographic list of all the names with that soundex. For instance:

```
Welcome to the Soundex Archive. There are 88799 names with
4607 soundex keys in the archive.

Enter a name: Larsen
LARSEN // Soundex L625
21
LAIRSON LARACUENTE LARASON LARCOM LAREGINA LARGEN LARGENA LARGENT LARGIN LARISON
LARKAN LARKIN LARKINS LARRISON LARSEN LARSON LARSSON LAURSEN LORSON LORSUNG LUERSEN

Enter a name: Paulsen
PAULSEN // Soundex P425
53
PALCZYNSKI PALESANO PALISANO PALLESEN PALUSZYNSKI PAULSEN PAULSON PELIKAN
PELLICONE PELOQUIN PHILOGENE PHILSON PHLEGM
PILKENTON PILKINGTON PILKINS PILKINTON PILSNER PILSON PLACENCIA PLACENCIO
PLACINO PLAGEMAN PLAGENS PLAGMAN PLAGMANN PLAISANCE PLASCENCIA PLASENCIA PLASSMAN
PLASSMANN PLASSMEYER PLEASANT PLEASANTON PLEASANTS PLEASENT PLESANT PLESNARSKI
PLESSINGER PLESSNER PLOSKUNAK POHLSON POLCHINSKI POLCYN POLCZYNSKI POLIQUIN
POLKINGHORN POLSON POLZIN POULSEN POULSON POWELSON PULKKINEN

Enter a name: xxx
XXX // Soundex X200
0

Enter a name: ^C
```

Additionally, as part of your program, you are to implement the following interface:

```
public interface CensusArchives
{
  /** Initializes an archive from an array of surnames */
  public void initNames (String[] names);

  /** Returns the key for a given name */
  public String getKey (String name);

  /** Returns a collection of names matching the key */
  public Collection<String> getNames (String key);

  /** Returns an iterator containing all the archive keys */
  public Iterator<String> getKeyIterator();
}
```

While the interface is general, the idea here is that the "key" in question for your implementation will be the soundex representation.

Note that since `String` objects are immutable, you may wish to make heavy use of the `StringBuilder` class in implementing the Soundex algorithm, though it is not required.

You should choose data structure(s) that support fast access and appropriate representations for the the types of queries that will be performed.

The course web page has a file containing a list of surnames you can use as input to your program.