Assignment 2.2: ArrayLists, For-Each loops, and File I/O

Due: 8am, Wednesday, November 19, 2008

Directions

The problems for this assignment are described below. You will submit this assignment by emailing it do Mr. Wulsin. The file will be named "Assignment 2.2-LastName.zip", where [LastName] is obviously your last name. All of your submitted assignments will be named in this fashion. The Zip file will contain a folder of the same name (so, "Assignment 2.2-LastName"), which will contain a folder for each of the subsequent problems. Each of these subfolders will have the name of its particular problem. Each of these problem subfolders will contain all of the necessary files for that particular problem. It's very important that you follow all of these naming conventions exactly as specified.

Problems

1) Cipher

You will make a substitution cipher decoder. Substitution ciphers are one of the oldest forms of encryption, where certain letters are substituted for other letters. Thus, the sentence

```
I love coding it up in Landon's Media Center B!
```

could be encoded into

```
M fglj rgpmkd mb ze mk Fokpgk'u Vjpmo Rjkbjh A!
```

As it turns out, such ciphers are quite straightforward to decode (as Mary Queen of Scots fatally discovered when her secret codes were decoded by Elizabeth's security minister Walsingham in what's known as the Babington Plot). Nevertheless, it does take a good bit of guess and check, and deciphering a long passage by hand would prove quite tedious.

Thus, you will develop a program that can efficiently decode an encoded text file. You will use character frequency analysis in this task. For example, if you know that the character `e` occurs most frequency in the english language and the letter `q` occurs most frequently in your encoded text, you can probably assume that the encoded character q corresponds to the decoded character e. We can continue doing this for the second, third, fourth, etc most frequent characters of the decoded and encoded text.

To represent the frequencies that correspond to each character, you will use the `CharData` class, already provided for you. This class is a very simple wrapper of a `char` variable representing a specifc character and a `double` variable representing the frequency of that character, both of which passed in and set in the constructor. You will use the CharData class to keep track of your accepted and calculated character frequencies.

We will use a clever way of representing our corresponding decoded and encoded characters: a `char` array. Since characters in programing are really represented by integers (which correspond to specific ASCII characters), we will use the integer value of our decoded text to be the index in the `char` array, which contains the encoded characters. So for example, if our decoded character `e` has an ASCII value of 101 and our char array is called `charMatches`, `charMatches[101] = 'q'`.

You will have a good deal of code to work from in programming your solution to this problem. Here's a short summary of the files included:

CharData.java contains the `CharData` class and is already written

charFrequencies.txt contains some "accepted" character frequencies for each letter that you'll use in your analysis

Cipher.java contains your `public static void main` method and is already written (but of course, feel free to change it as you wish)

Decoder.java contains `Decoder` class, which has the skeleton and a few methods already coded. You fill in the rest!

encoded.txt contains the encoded text for your to decoded!

Encoder.txt contains the `Encoder` class, which was used to encode the **encoded.txt** file

Also, make sure to use at least one or two for-each loops in your code.