Substitution Cipher

Background

Substitution ciphers[1] are a family of text encryption ciphers in which each letter of your message (called the plain text) is replaced by some other letter to form the encrypted message (called the cypher text). Which letter is used to replace another is defined by the encryption key.

For example, if the encryption key says to replace "e" with "x", "t" with "b", "h" with "m", and "s" with "r" (within a longer key) then the plaintext "these" would become the cyphertext "bmxrx".

Decrypting a substitution cipher then reverses the operation. You use the encryption key backward: seeing what encrypted letter came from which original letter and then making the substitutions to the cyphertext message.

In a substitution cipher, any mapping from the original letters to the encrypted letters is possible.

A characteristic of text encrypted with a substitution cipher is that the letter frequencies[2] of the original message and of the encrypted message are the same except that the frequencies are attributed to different letters. Consequently, a first step in decoding an encrypted message without the encryption key (a process called breaking the code) has you compute the letter frequencies in the ciphertext and matching the letters to the known letter frequencies in the language that you expect the message to be in. For example, the top 5 letter frequencies in English are "e" at 12.7%, "t" at 9.1%, "a" at 8.2%, "o" at 7.5%, and "n" at 6.7%. Then if a cyphertext had the 5 more frequent letters being w, c, h, m, and q then we would start with an encryption key that had mapped "e" to "w", mapped "t" to "c", mapped "a" to "h", mapped "o" to "m", and mapped "n" to "m".

Each language has different letter frequencies, so you can also get a guess at the language of the message by comparing the letter frequency distribution of your ciphertext message with the distribution for each language; if one language has a much smaller difference in frequency distribution with your ciphertext then you would begin by assuming the message originated in that language.

In our encryption or decryption process, you leave punctuation and white space as-is. You do, however, preserve the upper/lower case of the letter.

---

[1] https://en.wikipedia.org/wiki/Substitution_cipher

[2] https://en.wikipedia.org/wiki/Letter_frequency

Goal

Our ultimate goal is to have a Java class that helps us to decrypt a message that has been encrypted using a substitution cipher. For now, we are developing the support classes that will provide functionality to this overarching class.

We have four support classes (see Figure 1):

- Corpus
- LetterFrequency
- LetterCount
- SubstitutionKey

The classes operate as follows

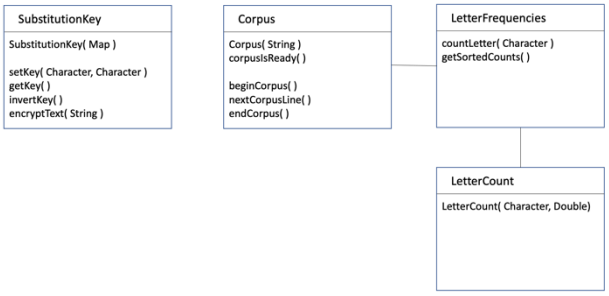

Figure 1 Interaction for support classes

Corpus

The Corpus class is a container for some body of text. In the envisioned big picture, this body of text could either be the text that we want to encode/decode or it could be some sample text. In the case of sample text, the notion would be to use the sample text to deduce the letter frequencies of that language so that we can then match the message to be decrypted with the nearest language frequencies.

The methods in the Cipher class create the object with data (the constructor), let us determine if the data was properly loaded (corpusIsReady), or provides us with a set of methods to iterate through the lines of the text one at a time (beginCorpus, nextCorpusLine, and endCorpus) much like you would in working through the content of a file.

beginCorpus signals an error condition by returning a value of false. nextCorpusLine signals the end of the file or an error condition by returning a null value.

The class stores the contents of the body of text in an ArrayList and stores the frequency of letters in an object of the LetterFrequencies class.

LetterFrequency

The LetterFrequency class tracks the use of each letter in some body of text. Each time a letter in the body of text is encountered, its occurrence is recorded using the countLetter() method. Once all of the letters are recorded, you can retrieve the frequency of all the letters using the getSortedCount() method, which returns a list of LetterCount objects sorted in decreasing order of frequency of letter occurrence. If two letters have the same frequency of occurrence then the alphabetically smaller letter comes first.

The letters are stored as lower case letters only; upper case letters are converted to lower case letters before recording their occurrence.

The class does not store non-letter characters like punctuation.

LetterCount

The LetterCount class is a helper class that keeps key-value pairs together. In this instance, it stores the pairing of a letter with the frequency of occurrence of the letter.

The frequency is stored as a Double so that the same class can either carry the count of occurrences of the letter or can carry a percentage value that represents what percentage of the text uses the given letter.

SubstitutionKey

The SubstitutionKey class stores an encryption key and allows a user to encrypt a string of text using that key as the key for a substitution cipher.

The encryption key is either provided as a Map (plaintext character and then ciphertext character) in the constructor or is built one plaintext / ciphertext character pairing at a time using setKey. The class allows you to retrieve the current key using the getKey() method so you can see what is stored.

The encryptText method then encrypts a string using the given substitution cipher and returns the encrypted key as the methods return value. If an error occurs, such as finding a character in the string to encrypt for which we do not have a substitution value in the encryption key, then the method returns null.

Last, the class provides the invertKey method that returns another SubstitutionKey object whose key is the decryption key for the current substitution cipher.