Programming Exercise: Known Language, Unknown Key Length

Assignment: English Language, Unknown Key Length

Now that you have broken Vigenère ciphers with a known key length, it is time to expand your program's functionality to break Vigenère ciphers of unknown key length. You will do this by adding three methods to your **VigenereBreaker** class, and then modifying your **breakVigenere** method.

Specifically, you should do the following:

- In the **VigenereBreaker** class, write the public method **readDictionary**, which has one parameter—a FileResource **fr**. This method should first make a new HashSet of Strings, then read each line in **fr** (which should contain exactly one word per line), convert that line to lowercase, and put that line into the HashSet that you created. The method should then return the HashSet representing the words in a dictionary.
- In the **VigenereBreaker** class, write the public method **countWords**, which has two parameters—a String **message**, and a HashSet of Strings **dictionary**. This method should split the message into words (use `.split("\\W")`, which returns a String array), iterate over those words, and see how many of them are "real words"—that is, how many appear in the dictionary. This method should return the integer count of how many valid words it found.
- In the **VigenereBreaker** class, write the public method **breakForLanguage**, which has two parameters—a String **encrypted**, and a HashSet of Strings **dictionary**. This method should try all key lengths from 1 to 100 (use your **tryKeyLength** method to try one particular key length) to obtain the best decryption for each key length in that range. For each key length, your method should decrypt the message (using **VigenereCipher**'s

decrypt method as before), and count how many of the "words" in it are real words in English, based on the **dictionary** passed in (use the **countWords** method you just wrote). This method should figure out which decryption gives the largest count of real words, and return that String decryption. Note that there is nothing special about 100; we will just give you messages with key lengths in the range 1–100. If you did not have this information, you could iterate all the way to **encrypted.length()**. Your program would just take a bit longer to run.

- Finally, you need to modify your **breakVigenere** method to make use of your new code. The steps below describe what your **breakVigenere** method should do. They are similar to the steps from the previous portion of this project but have been updated: new items are in italics, and old items that you should no longer do are stricken through.
    a. Create a new FileResource using its default constructor (which displays a dialog for you to select a file to decrypt).
    b. Use that FileResource's .asString() method to read the entire contents of the file into a String.
    c. *You should make a new FileResource to read from the English dictionary file that we have provided.* ~~Use the tryKeyLength method that you just wrote to find the key for the message you read in. For now, you should just pass 'c' for mostCommon.~~
    d. *You should use your **readDictionary** method to read the contents of that file into a HashSet of Strings.* ~~You should create a new VigenereCipher, passing in the key that tryKeyLength found for you.~~

e. *You should use the **breakForLanguage** method that you just wrote to decrypt the encrypted message.* ~~You should use the VigenereCipher to .decrypt() the encrypted message.~~

f. Finally, you should print out the decrypted message!

Test this method on the text file **athens_keyflute.txt**. The first line should be “SCENE II. Athens. QUINCE'S house”, and the key is “flute”, or {5, 11, 20, 19, 4}.