Comp-122 Assignment 2

Andrew Lawler

Introduction

For this assignment I was tasked with creating a cipher text encryption and decryption program.

For the first part of the task I was required to create an interface and then implement the interface in part B.

For part b, I had to implement the methods in a class called Caesar. Within this class I had the methods (rotate, frequencies, chiSquared and decipher). I implemented these methods using my knowledge of cipher text and the suitable examples shown to us.

Note: I asked Patrick if I could use a code snippet and he said it was fine. I used this snippet, which I heavily edited, to handle the rotating of lowercase and uppercase letters. I showed in the code which if statement this was. All code around it (eg: is letter) and other parts, are all my own. I also edited the code snippet as I wanted the code to be easy to edit for other inputted languages. You can see with the link provided in the code that the code itself has been changed a lot.

For the other two parts of the assignment we had to create applications which in turn handled the inputs from the command line. We than had to take those inputs and crease a Caesar method which used the desired method. For example, the rotate application used the rotate method.

Next Parts

c. Testing input for Rotate.java

I implemented a system that will print wrong input length when there are not two inputs. It then tests the inputs to make sure they are in the correct format.

Input	Expected Output	Actual Output	Comments
	Too few parameters! Usage: java Rotate n 'cipher text'	Too few parameters! Usage: java Rotate n 'cipher text'	This shows that the program ends with an error message when there is nothing entered.
.	Too few parameters! Usage: java Rotate n 'cipher text'	Too few parameters! Usage: java Rotate n 'cipher text'	This result tells us that the program will revoke all types of input when there is not the correct amount of inputs.

3	Too few parameters! Usage: java Rotate n 'cipher text'	Too few parameters! Usage: java Rotate n 'cipher text'	This example shows the error message once again.
Andy	Too few parameters! Usage: java Rotate n 'cipher text'	Too few parameters! Usage: java Rotate n 'cipher text'	This shows that the program handles the one string input and provides the same error message.
Andy 3	No integer first, no string second, wrong input	You did not enter an integer first! You did not enter a string second! Wrong inputs!	The output here shows that order matters. It tested the first input and found it was not an integer and I also found the second input was not a string. These are the wrong inputs.
3 Andy	Dqgb	Dqgb	Here you can see if we enter in the correct format it will work.

d. Testing input for BreakCaesar.java

I implemented the same system as for part c but with one input this time. It will check if the input size is equal to one and will then check if that input is a string. If either case is false, we will get a specific error message.

Input	Expected Output	Actual Output	Comments
	Too few parameters! Usage: java BreakCaesar 'cipher text'	Too few parameters! Usage: java BreakCaesar 'cipher text'	This shows that the program requires inputs and will not let you enter nothing.
3	You didn't enter a string	You did not enter a string!	This shows that you need to enter a string because the input is tested.
.	.	.	This was to test that the function did in fact let this in. A dot is allowed as we

			may form sentences to decrypt.
xf bsf ifsf	we are here	we are here	This was to test that the program would work.
olssv	hello	ebiil	This test was to show that the program will not always decipher correctly. If the text input is not long enough or does not contain enough repeating characters, we may not get the correct output. As you can see from this example, I have proved that the chiSquared method is not always going to work.

e. UML Diagram

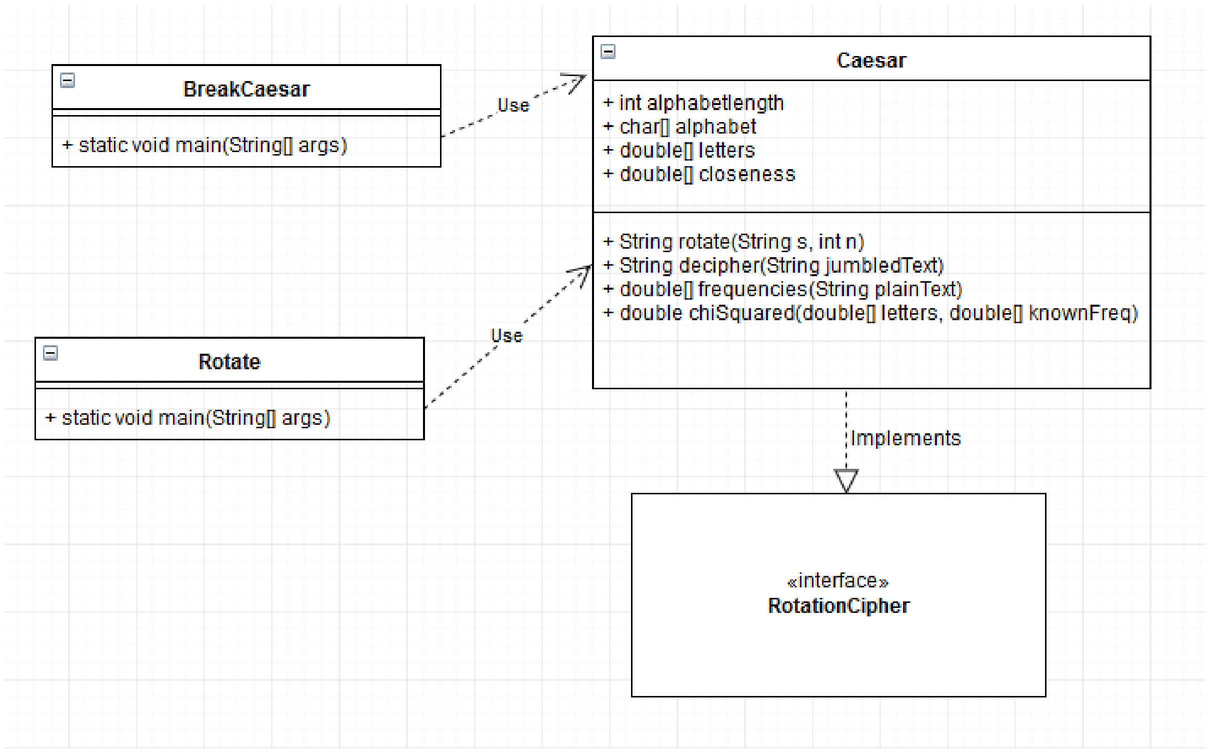
Class Description

I first initialized the interface for the assignment. This interface is called RotationCipher and its job is to simply state the methods for the subclass to implement. Interfaces state methods as public but it is known that they are in fact abstract also. Once I had implemented my interface, I moved onto my Caesar subclass. This subclass implements the interface itself. I wrote the code for the methods and made sure they followed the same input parameters and same name as the ones in the interface, it in turn overrides them.

I then was tasked with implementing the BreakCaesar and Rotate applications. For this task I took the inputs from the user and then also used try catch blocks to make sure the inputs where as they should be. When I had the desired inputs, I created a Caesar object which then calls the rotate/decipher method. Once I have used the method, I then return the answer and print it to the user. This was the quickest and most neat way to do this.

I think the way I designed my classes is good because they all link together well, and it keeps the code neat and efficient.

UML Diagram


BreakCaesar
+ static void main(String[] args)
Use
Caesar
+ int alphabetlength
+ char[] alphabet
+ double[] letters
+ double[] closeness
+ String rotate(String s, int n)
+ String decipher(String jumbledText)
+ double[] frequencies(String plainText)
+ double chiSquared(double[] letters, double[] knownFreq)
Use
Rotate
+ static void main(String[] args)
Implements
«interface»
RotationCipher

f. JavaDocs

I was asked to use JavaDoc to add comments to my methods, interfaces and classes. I added these throughout my project and I also added some extra background information as I went along.

As JavaDocs creates a lot of un-needed stuff, I have put the relevant HTML files inside a subfolder in 'src' called 'RelevantDocs'. Hopefully this makes your life a little bit easier.

g.

Question: Caesar would have written in Latin instead of English. What would we do differently if we know the language, we're examining isn't English but some other language?

This scenario would be quite easy to implement. First, I would need to edit the global variables I provided at the top of the Caesar program. I coded the entire assignment to suit this scenario. You would simply change the alphabetlength integer to the length of your new languages alphabet and then also type the alphabet as a string into the alphabet char array part below. This would then implement these changes throughout the entire program as the entire program runs off of the values implemented in these variables. You would then only need to make a few more changes. You would edit the knownFreq double array with the frequencies for every letter in your language. Lastly, I might need to change my character set also. ASCII cannot handle all languages so this would need to be changed. This means that the Rotate method would need to be changed so that it can handle this new language. Once I have changed all of this, my code would run perfectly.

h.

Question: Suppose we (somehow) know that the person doing the encryption uses one shift value for lower case letters, and a different shift value for upper case letters. What would we have to do differently? How would that affect our calculations, or how would we have to alter our program/calculations to account for this?

This scenario would actually be incredibly hard. We would need to first distinguish which letters are capitals and also which are not. Once I know which letters are capitals and which are not, I would run the decipher method on the capital letters. I would then compare the results with the specific letters and only those letters. That way, the chi value closest to zero should be the English rotation. I would then complete this step for the lowercase letters too. I would rotate them for every possible rotation and then compare their frequency with the chi squared frequency. Something I forgot to add, I would have to change my frequency method to only check specific letters. I wouldn't want to calculate the frequency for every single letter. The only way I could do this was if I completely edited my chiSquared method. I would have to distinguish the letter positions as capital, lowercase and I would have to only loop through the specific positions. Overall, after thinking about this it would be possible.