

# Assignment 02 – Comp 122

## Part 1 – Caesar Cipher

### Summary of the requirements

For this first part of the requirements we first had to make a function that could take a string of characters and shift them a certain amount of places using the Caesar cipher method of encryption/decryption. This function allowed strings of characters to be moved by a specific shift that was input by the user. The next requirement is to compute the letter frequencies for the decoded text. This means counting how many times a character appears in a string and then dividing that value by the length of that string, this returns a decimal value of the frequency of that letter. The third requirement for the program was to compare the frequencies of the string of characters with the known English frequencies. If the frequencies were lower and still decrypted a given string then that means that the decryption is closer to the English known frequencies meaning that the decoded string is likely to be closer to correct English. To measure the closeness of English frequencies we have to use the following formula:

$$\chi^2 = \sum_{a=a}^z \frac{(\text{freq}_a - \text{English}_a)^2}{\text{English}_a}.$$

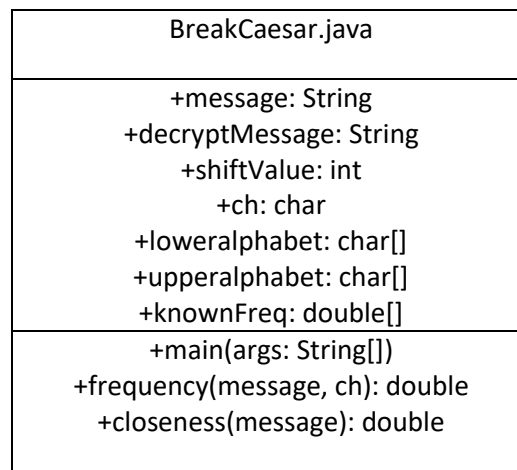
Freq represents the frequency calculated from the given string, English represents the known frequencies for each character. By using this calculation we can complete the final requirement which goes through all 26 possible shifts of the string and uses the calculation to get the lowest frequency value. Once we have done this we do not need any user input for the shift, we can go through all 26 possible shifts and choose the one with the lowest frequency value which should be the English translation.

### Analysis of the problem

Firstly for my program I will be making a function that uses the Caesar cipher method to shift the characters in a string by a value given by user input. Once I have done this I shall be making another function that measures the frequency of each character within a string and returns a decimal value so I can use it later on in my calculations. So if the string was “banana” the frequency of “a” is 3/6 which is equal to 0.5. Once I have made a frequency function I can then make a function that uses the closeness calculation to compare the frequencies of the characters in a string with the known English frequencies. The closeness function will use the frequency function that returns a decimal and will then implement the above calculation into the program. Once this function has been made all I need to do is shift through all known 26 possible shifts and use the lowest value given by the closeness function. This value should then be plain English as it’s the lowest frequency value. This

means I can then use this instead of taking user input of a given shift and the program should automatically decrypt the string based upon the closeness function value.

### Class diagram



### Class attributes

The **message** attribute is the string that is input by the user which is then added to the args, when the user starts the program using the command prompt they will have to add the string of chars to the args when starting the program.

The **decryptMessage** attribute is the message that has been shifted back to normal English, this attribute is output to the user as the decrypted string.

The **shiftValue** attribute is a counter for the amount a string of characters is shifted, the program shifts a total of 26 different times meaning that the shift value attribute acts as a counter for each shift.

The **ch** attribute represents each character within the string and is used when shifting the characters within the message attribute.

The **loweralphabet** attribute is an array of 26 chars. I use this attribute to store all 26 of my lowercase characters.

The **upperalphabet** attribute is an array of 26 chars. I use this attribute to store all 26 of my uppercase characters.

The **knownFreq** attribute is an array of all of the frequencies of characters for known English. I use these frequencies in my closeness function to compare them to the frequencies of letters in my message string.

## Class methods

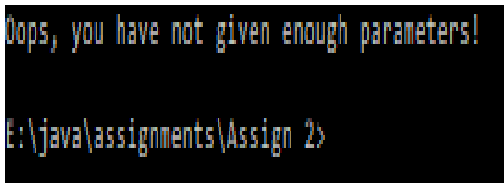
In the **main** I have my function to shift the characters of a string by a specific amount. In my main I also output to the user the decrypted string and I take the user input through the args. At the start of the main I put every character in the alphabet into my loweralphabet array and every uppercase character into the upperalphabet array.

In the **frequency** method I use a for loop that loops through every single instance of individual characters coming up in the message. Every time each character comes up it is added to my charCounter variable which is the frequency of how many times each character appears. For example, In the string "dddr" the character "d" appears 3 times, I then use this value and divide it by the length of the string. As "d" appears 3 times and the string length is 4 it will be  $\frac{3}{4} = 0.75$ . I then return this decimal value so it can be used by my closeness method.

The **closeness** method uses the frequency calculated by the frequency method. It then uses the calculation given to us in the assignment brief to calculate how close the frequencies of the characters in the string are to the frequencies in my knownFreq array. This then returns a frequency value, the lowest frequency value returned means that that string of characters is closer to the known English frequencies in the knownFreq array making it the closest to English.

## Testing – BreakCaesar.java

Test number	Test	Test input	Expected output	Actual output
1	Testing to ensure that the user enters parameters	Not entering an input as I want to ensure my program gives an error message if the user gives no input in the args.	The program should tell the user they have not input enough parameters and the program should close.	"Oops, you have not given enough parameters! "  The program worked correctly and gave an error message as nothing was input by the user
2	Testing to ensure that uppercase characters in a string remain uppercase after they have been shifted	I shall be inputting the string "Vlcha siol mqilx. Qy unnuwe un xuqh."	The expected output should be "Bring your sword. We attack at dawn." The B should be uppercase and the W should be uppercase.	"Bring your sword. We attack at dawn."  The program output the uppercase letters correctly
3	Testing to ensure the program consistently outputs the correct decryption of a string	I shall be inputting the string "htcs aplntgh, vjch, pcs bdctn"	The expected output should be "send lawyers, guns, and money"	"send lawyers, guns, and money"  The program decrypted correctly, I feel that more test cases are needed to ensure the program works correctly.
4	Testing to ensure the	I shall be inputting the string	The expected output should be	"Mississippi delta blues"

	program consistently outputs the correct decryption of a string (Test case 2)	"Wscscscszs novdk lveoc"	"Mississippi delta blues"	The program decrypted correctly, I feel that I need to perform one more test case to ensure the program works correctly.
5	Testing to ensure the program consistently outputs the correct decryption of a string (Test case 3)	I shall be inputting the string "Vlcha siol mqilx. Qy unnuwe un xuqh."	The expected output should be "Bring your sword. We attack at dawn."	"Bring your sword. We attack at dawn."  After this test case I'm confident that my program can consistently decrypt the string.
6	Testing to ensure that numbers input as the string remain integers and do not get affected by the shift.	I shall be inputting "5555" to ensure that numbers are not affected by the shift.	The expected output should be "5555" as the numbers should not be affected by the shift.	"5555"  The output was correct, the numbers did not get affected by the shift
7	Testing to ensure punctuation is not affected by the shift	I shall be inputting the string "Vlcha siol mqilx. Qy unnuwe un xuqh."  I want to see if the full stops are affecting in this string.	The expected output should be "Bring your sword. We attack at dawn."  And the full stops should not be affected by the shift.	"Bring your sword. We attack at dawn."  The full stops were not affected by the shift meaning that the output was correct.
8	Testing to ensure the program stops after the error message if the user does not enter a string into the args.	I shall not be inputting anything, I want to ensure that the program stops if the user does not enter a string.	The expected output should be "Oops, you have not given enough parameters! " However this test is mainly focussing on ensuring the program stops correctly.	  The program stopped the program correctly. It output to the user correctly then stopped the program. No changes needed.
9	Testing one of the encrypted strings from the assignment brief to ensure the program	I shall be inputting the string "Hvs eiwqy pfckb tcl xiadsr cjsf hvs zonm rcug."	The expected output should be "The quick brown fox jumped over the lazy dogs."	"The quick brown fox jumped over the lazy dogs."  The output was correct. The string was decrypted correctly.

	decrypts it correctly	I want to ensure that my program can decrypt this string correctly		
--	-----------------------	--	--	--

### Pseudocode for methods

MAIN:

```
    INT counter = 0
    FOR CHAR c = a; c <= z; c++
        Loweralphabet[counter] = c
        Upperalphabet[counter] = Character.UPPER(c)
        Counter++
    IF args.length == 0
        OUTPUT ("Not enough parameters")
        EXIT(1)
```

Message = args[0]

DOUBLE lowestval = closeness(message)

STRING lowestmsg = message

FOR INT SHIFTVAL = 0; SHIFTVAL < 26; SHIFTVAL++

```
    FOR INT X = 0; X < message.length; X++
```

```
        Ch = MESSAGE.CHARAT(X)
```

```
        IF Ch >= a && Xh <= z
```

```
            Ch = char ch - SHIFTVAL
```

```
            IF ch < a
```

```
                Ch = CHAR CH + z - a + 1
```

```
            Decryptedmsg += ch
```

```
        ELSE IF Ch >= A && Xh <= Z
```

```
            Ch = char ch - SHIFTVAL
```

```
            IF ch < A
```

```
                Ch = CHAR CH + Z - A + 1
```

```
            Decryptedmsg += ch
```

```
        ELSE
```

```
            Decryptedmsg += ch
```

```
    IF lowestval > closeness(Decryptedmsg)
```

```
        Lowestval = closeness(Decryptedmsg)
```

```
        Lowestmsg = Decryptedmsg
```

```
    Decrpytedmsg = ""
```

OUTPUT (message)

OUTPUT (Lowestmsg)

```
PUBLIC STATIC DOUBLE frequency(String message, CHAR ch)

    DOUBLE counter = 0
    FOR INT I = 0; I < message.length; I++
        IF message.CHARAT(I) == Ch
            Counter++
    RETURN counter/message.length

PUBLIC STATIC DOUBLE closeness(String message)

DOUBLE top = 0
DOUBLE bottom = 0
CHAR character;
FOR INT I = 0; I < message.length; I++
    FOR INT Y = 0; Y < 26; Y++
        Character = message.CHARAT(I)
        IF character == upperalphabet[y] OR character == loweralphabet[y]
            Top = MATH.POW((frequency(message,character) - knownfreq[y], 2)
            Bottom = bottom +(top/(knownfreq[y]))
    RETURN bottom
```

### Part 1 – Question to consider

***“What would we do differently if we know the language we’re examining isn’t English but some other language (e.g. suppose we know the people communicating via this Caesar cipher usually writes/speaks in Polish)?***

***Suppose we (somehow) know that the person doing the encryption uses one shift value for lowercase letters, and a different shift value for uppercase 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?”***

Well firstly we would need to ensure that our known frequency array has the known frequencies for whatever language we would need to decrypt, for example if the language was Polish we would need to find the frequencies for each letter in Polish and store it in an array like we did for the English known frequencies. Also considering that the person doing the encryption uses a shift value of 1 for lowercase letters and uses an unknown shift value for uppercase letters we could simply apply a shift value of 1 for all lowercase letters by creating a variable that is used to shift the lowercase values by one. However since the shift is unknown for the uppercase letters we would

need to use a for loop that iterates through all 26 possible shifts and then use the frequency and closeness methods to calculate which shift has the lowest frequency. The shift with the lowest frequency when compared to the known frequencies for that language should be the decrypted form of that uppercase character.