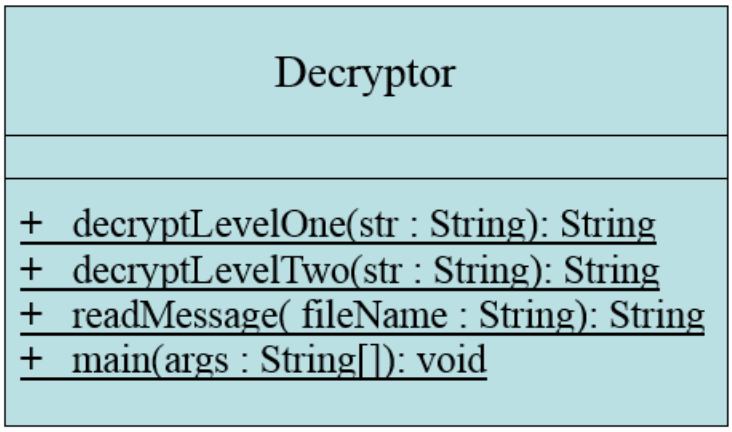
CS 111, Programming Fundamentals II
Lab 4: Encrypted Message

Computer Science

In this lab is you will gain experience with the methods in the wrapper class Character. You will also explore some of the methods of the String and StringBuilder classes.

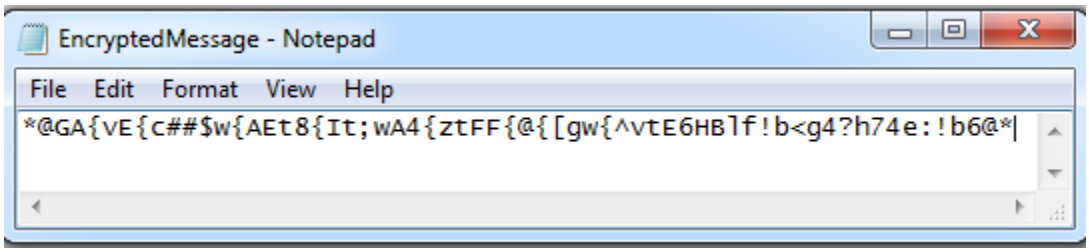
Let us pretend that the company XYZ deals with a lot of sensitive data. Whenever they send clients data over the internet, they encrypt the data. The receiver of the data has a program which decrypts the message, allowing them to read the original information being sent. This way XYZ is able to send sensitive information across the network without any concerns. Due to the restructuring of the engineering staff, the pseudocode for the encryption algorithm has been leaked.

You have been paid very handsomely by XYZ to try and reverse engineer the encryption process from the pseudocode. They want to see if it is possible to decrypt there messages. You are given the pseudocode and the latest message encrypted using their encryption algorithm.



Write the class **Decryptor.java** according to the UML diagram above. You will write 3 custom methods plus the main method. The encryption algorithm has two steps to it. First, the original message gets modified by updating the Unicode decimal values of the individual characters. Then the modified text gets further encrypted by replacing certain characters with other characters, inverting the case of the characters, adding additional characters at the beginning and the end of the message. After all of that the string sequence is reversed and ready to be send.

Since you are trying to reverse engineer the process you will have to start on step two of the encryption algorithm, and then after that do step one. Below is the encrypted message.



I. readMessage()

This method takes in the name of the text file and returns the message as a String. Download the file **EncryptedMessage.txt** from Canvas. There are several ways to read the text file, a simple one would be to create a File reference and pass it to a Scanner. Read the single line and return it.

II. decryptLevelTwo()

It is crucial that these steps are implemented in order. If some of the steps are done out of order than you will not be able to decrypt the message successfully. Here are the steps inverse order, which are done at the second level of the encryption:

- ***Replace every "!"character with an "a"***
- ***Replace every "{" character with a "x"***
- ***Remove the signatures characters "*@" from the end and the beginning of the message***
- ***Reverse the characters in the message***
- ***Change every lowercase character to be uppercase and uppercase to be lowercase***
- ***Return the message***

III. decryptLevelOne()

At the first level of the encryption, the algorithm takes each character and retrieves its code point (Unicode decimal value). Subtracts 13 from it, gets the character at the new code point and replaces the old character with it. You can either use StringBuilder to modify the characters, or simply create a new String and add to it the new characters as you get them. You will have to the do the following in this method for each of the characters:

- ***Get the code point of a character and add 13 to it***
- ***Get the character at the new code point, replace it with the old one***

IV. main()

```
str = readText("EncryptedMessage.txt")

str = decryptLevelTwo(str)

str = decryptLevelOne(str)

print(str)
```

Figure 1: Pseudocode for main method

V. Useful Methods

Below are a few useful methods. For more methods and how to use them please refer to the Java API.

Method	Description
boolean isLowerCase(**char** ***ch***)	**(Character)** Returns **true** if the argument passed into ***ch*** is a lowercase letter. Otherwise returns **false**.
char toUpperCase(**char** ***ch***)	**(Character)** Converts the character argument to uppercase using case mapping information from the UnicodeData file.
int codePointAt(**int** ***index***)	**(String)** Returns the character (Unicode code point) at the specified index.
replace(int *start*, int *end*, String *str*)	**(StringBuilder)** Replaces the characters in a substring of this sequence with characters in the specified String.
reverse()	**(StringBuilder)** Causes this character sequence to be replaced by the reverse of the sequence.
String replace(char old, char new)	**(String)** Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.

VI. Upload your work to Canvas

For this lab, make sure that you upload the following files to the Lab 4 assignment in your Canvas account:

Decryptor.java
Screenshot image containing the decrypted message

Rubric

File / Lab	Points
Method *readMessage* is implemented correctly	15
Method *decryptLevelTwo* is implemented correctly	35
Method *decryptLevelOne* is implemented correctly	25
main method is calling the other methods correctly	10
The message is decrypted correctly	10
Screenshot of the output, code is commented and formatted	5
Total	100