ITP 120 Project 3
(100 points)

Due Date: Monday, April 10 at 11:59 PM

Overview

Write a program that implements a file encryption/decryption system. Your program will read in a text file and either encrypt or decrypt the contents at the users request using a simple "Caesar cipher" encryption algorithm. Your program is required to implement at least 2 methods of your choosing. Suggested methods are provided below. Using *classes* for this project is not required.

This assignment is worth 100 points, where 80% is based on functionality (successfully satisfying the requirements), and 20% is based on style (documentation, readability, general program organization).

The name of your (main) java source file should be CaesarCipher.java

Background Information (Caesar Cipher)

The Caesar Cipher is a simple *substitution cipher* in which each letter in the *plaintext* is replaced by a letter some fixed number of positions down the alphabet to create the *ciphertext*. For example, with a right shift of 8 Caesar algorithm, the letter D would be replaced with the letter L because its 8 units downstream in the alphabet. To decrypt the letter L, we shift 8 alphabetic units to the left giving us the letter D.

Care must be taken to handle "rollover conditions" if the beginning/end of the alphabet is reached during the shifting. The shift operation should wrap-around to the other side of the alphabet where the shifting continues. For example, with a right shift of 8, the letter Y would be replaced with the letter G.

Example (using a shift 8 algorithm):

Plain text:	THE QUICK WHITE RABBIT
Cipher text:	BPM YCQKS EPQBM ZIJJQB

Assignment Instructions:

1. Encryption/Decryption will be handled using a **shift 8** operation where
   a. Encryption is a right shift (e.g. A → I)
   b. Decryption is a left shift (e.g. Z →R)

2. To keep things simple, we will only deal with uppercase alphabetic characters (no numbers, or other special symbols), you can assume that your input file will contain only uppercase characters of the alphabet.

3. Blank spaces should be ignored, blank spaces in the plain text should equate to blank spaces in the cipher text.

4. Query the user for a command, valid commands are either **encrypt** or **decrypt.**

5. Query the user for a file to encrypt/decrypt based on the users entered command.

6. Query the user for an output file name that create and fill with the encrypted/decrypted contents.

7. Perform the encryption/decryption on a line by line basis, read in a line of text from the input file, encrypt/decrypt the text and then append the modified contents to the output file.
   a. If you run the program once and encrypt a file, and then run it again and decrypt the generated file from the first run, your new decrypted file should be identical to one you started with

Hint
There are several methods you could use to perform the character shifting needed for the encryption/decryption. Keep in mind that that characters are represented in java by the u*nicode* numbering scheme. This allows you to perform mathematical operations (such as addition, subtraction, etc).

It might also be useful to know that the letter A (uppercase) is equivalent to Unicode value 65, and all uppercase characters are offset from this value (e.g, B=66, C=67, …).

Suggested methods to implement
String encryptText(String plaintext);   // Performs encryption on the text provided
String decryptText(String ciphertext); // Performs decryption on the text provided
char shiftRight(char c); // performs a right shift operation on *c* and returns the result
char shiftLeft(char c); // performs a left shift operation on *c* and returns the result

Zip your source file(s) into a zip file called "proj3_<last name>.zip (e.g. proj3_wolfe.zip). Note, you do not need to include any data files that you have encrypted/decrypted. Submit your source as an attachment to this assignment on blackboard.