Lab 1: Caesar Word Salad

Pre-Lab Notes

Lab submissions will **not be accepted late** – they must be submitted by **midnight on the day they are assigned**. Some latitude might be allowed (I won't disqualify a submission that's mere minutes late!) but that's purely discretionary.

Labs are treated differently from assignments – you're encouraged to work with fellow students, ask questions of the TA during your lab section, and generally collaborate to complete your work. You should still submit your own work, **including citing with whom you collaborated in your submission**, but using answers developed by others will not be treated as plagiarism.

Nevertheless, you are still encouraged to make a good faith effort to complete your labwork, in order to exercise your understanding of the topics it covers. Lab submissions are marked purely 1 or 0 by whether the TA believes you at least made an honest attempt, even if you didn't succeed, so the value is in what the attempt teaches you rather than simply having the right answer.

1. Setting Up Java

One of the goals of this lab is to shake out any problems with our coding setups. As such, it's highly recommended you use the **IntelliJ IDE**, which provides a fairly standardized Java coding environment that cuts down on the risk that your code won't run for the TAs.

Install IntelliJ Community Edition (don't worry, it's free) :
https://www.jetbrains.com/idea/download/

Also be sure you're using Java 14 under File -> Project Structure, you may need to download a new SDK in order to have it available! 15 *should* be compatible as well, though testing if that's true is one of the purposes of a good technical shakedown.

2. Introducing the Caesar Cypher

One of the oldest and most basic cryptography techniques in history, the Caesar Cypher is a way to encrypt a message by substituting every letter of the plain text with a letter some number of positions up or down in the alphabet. So for example, a “left shift” of three – Caesar's preferred cypher – would turn the letter D into a letter A. An A, meanwhile, would have become an X, since the alphabet wraps around again.

These sort of cyphers offer very little protection against a reasonably clever person with some scratch paper and spare time, and effectively none against a computer that can brute force every possibility in the blink of an eye. Nevertheless, setting up a Caesar Cypher Java program can be a fun exercise to test our understanding of Strings, characters, arrays, and objects.

2.1 The Secret Message

Alongside this outline on the course website's assignment page, you should be able to find a plain text file named “question.txt”. This message has been encrypted using a Caesar Cypher, though

the number it's been shifted is a secret.

It'll be up to you to write a simple Java program that can decrypt this message. You must then read it, answer the question it poses, and then encrypt and submit your answer using the same Caesar Cypher as the question. You'll also be submitting your code, which should be set up to decrypt your answer for us.

2.2 Outlining your Objective

For starters, the message is in all capital letters, both encrypted and decrypted, so only concern yourself with those. Your answer should also be written in capital letters. You don't need to encrypt punctuation either.

Your program does, however, need to be able to take in an encoded message. You can't just hard code the contents of question.txt directly into your program, it should either be read from the user typing it into the terminal window or else read out of the plain text file directly (in which case the user may still need to type in the file name).

Your program doesn't need to automatically write an answer.txt, though it should at least print its output to the screen. It should also be able to encrypt a plaintext message – how else are you supposed to encrypt and submit your answer to the question?

As for what the secret shift is for the cypher used to encrypt question.txt, your code doesn't have to solve that for you on its own – brute force is perfectly acceptable – though there are probably ways to make this less painful.

2.3 Tips on Getting Started

This lab will exercise your understanding of Strings, chars, arrays, objects, and primitive data types. In particular, remember that a String can be turned into an array of chars with theString.toCharArray();, while an array of chars can be turned back into a String with new String(theCharArray);.

Remember that the char primitive data type can also be used as a number, and that these numbers are in order – 'A' is equivalent to 65, while 'Z' is 90. You might be able to get somewhere treating letters as numbers, changing those numbers by your cypher's shift, and then swapping them back into letters. You might even want to build some arrays of your own, storing things your modified version of the alphabet.

Don't forget that encrypting and decrypting are essentially two sides of the same coin – if you can figure out one, the other is probably the same thing in reverse!

3. Deliverables

All lab submissions will be done through CourSys at https://courses.cs.sfu.ca/. For this lab, that should include:

1. A .zip archive of your code, organized into a single Java project. To ensure compatibility, it's

recommended to use IntelliJ while developing your code, then using the export function under File -> Export -> Project To .zip File.

2. A plain .txt file containing your encrypted answer to the encrypted question. There is a separate field to submit this on CourSys, which accepts one file named **answer.txt**.

Be sure you tidy up your code before submitting! Check the Java code style guide posted on the course website, and do your best to provide helpful comments.