**CMSC 203**

**Fernando Gonzales-Vigil Richter**

**Assignment 3**

**Pseudocode**

String in Bounds:

FOR EACH character of plaintext (charAt(i))

IF character is below the lower bound or above the upper bound

RETURN false

Otherwise RETURN true

EncryptCaesar:

SET encrypted as an empty string

SET key as the remainder of key/RANGE

FOR EACH character in plaintext charAt(i))

SUM int c as character + key

WHILE c is bigger than UPPER\_BOUND

Withdraw RANGE of c

Cast c to character and CONCATENATE it in **encrypted**

RETURN encrypted

EncryptBellaso:

SET encrypted as an empty string

FOR EACH character in plaintext charAt(i))

SET j as the remainder of the index of I (index of plaintext) / the length of bellasoStr

SET c as the sum of the characters of plaintext at i and bellasoStr at index j

WHILE c is bigger than UPPER\_BOUND

Withdraw RANGE of c

Cast c to character and CONCATENATE it in **encrypted**

RETURN encrypted

DecryptCaesar:

SET decrypted as an empty string

SET key as the remainder of key/RANGE

FOR EACH character in encryptedText - > charAt(i))

SUM int c as character + key

WHILE c is smaller than LOWER\_BOUND

ADD RANGE to c

Cast c to character and CONCATENATE it in **decrypted**

RETURN encrypted

DecryptBellaso:

SET decrypted as an empty string

FOR EACH character in encryptedText -> charAt(i))

SET j as the remainder of the index of I (index of plaintext) / the length of bellasoStr

SET c as the sum of the characters of plaintext at i and bellasoStr at index j

WHILE c is smaller than LOWER\_BOUND

ADD RANGE to c

Cast c to character and CONCATENATE it in **decrypted**

RETURN decrypted

**UML**

A screenshot of a cell phone

Description automatically generated

Test Cases (for FX App)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nº | Encryption/method | String | Key | Decrypt | Fails? |
| 1 | Caesar | Sunday morning | 667 | .0)\_\4;(\*-)$)" |  |
| 2 | Caesar | Sunday morning | 5 | XZSIF^%RTWSNSL |  |
| 3 | Caesar | Hello everyone | 17 | YV]] 1V'V#\* \_V |  |
| 4 | Bellaso | Spring Boot | YO | ,\_+X'V9Q(^- |  |
| 5 | Bellaso | Spring Boot | INJECTION | \^\NQ[)Q]X" |  |
| 6 | Bellaso | Once upon a time | 77777 | FE:<WLGFEW8WK@D< |  |
| 7 | stringInBound() | address@yahoo.com |  |  | Yes |

Screenshots:

Test # 7

A screenshot of a cell phone

Description automatically generated

Test # 1

A screenshot of a cell phone

Description automatically generated

Test # 2

A screenshot of a cell phone

Description automatically generated

Test # 4

A screenshot of a cell phone

Description automatically generated

Test # 5

A screenshot of a cell phone

Description automatically generated

**Lesson Learned**

While trying to figure out how to solve the algorithm to run In anycase, since sometime it threw strange characters just in some specific cases, I solved by simply pen and paper, with a table to take note of what character was expected, what character was the output, what was the ascii code for both characters and the difference with the correct characters. It happened to be the the size of the lower bound, so I used an extra verification when the returning result was going to be out of range while it was a character.

Also, I thought about using Stringbuilder class instead of String, since Strings are immutable so every time you concatenate, you are creating a new string, which is inneficient. Stringbuilder has the concat() method and all stays in the same object. However, since the subject was beyond the scope of the Assignment and the assignment wasn’t really about efficiency but simply passing the Junit tests, I decided to not use it to avoid additional complexity.