**CMSC203 Assignment 3 Documentation**

Class: CMSC203 CRN 31338

 Program: Assignment #3

Instructor: Ashique Tanveer

 Summary of Description: GUI and system for encrypting and decrypting text using Caesar and Bellaso ciphers.

 Due Date: 03/21/2025

 Integrity Pledge: I pledge that I have completed the programming assignment independently.

 I have not copied the code from a student or any source.

**Student: William Keller**

**Part1: Pseudo Code:** Here is a pseudo code for Assignment 3 program:

**FXDriver.java**

Define a **class** called FXDriver that extends Application

Define a **public static method** main(args: String[]) -> **void**

**Call** launch(args)

**Override** the start method with the signature (stage: Stage) throws Exception -> **void**

Declare **root**: FXMainPane = new instance of FXMainPane

Declare **scene**: Scene = new instance of Scene(**root**, width=600px,height=350px)

Set the stage’s scene to the created scene

Set the stage’s title to “Cybersecurity Encryption and Decryption”

Call stage’s **show** method

End of class

**CryptoManager.java**

Declare a class called CryptoManager

Define **private static** constants:

* LOWER\_RANGE: char = ‘ ’
* UPPER\_RANGE: char = ‘\_’(ASCII value of underscore)
* RANGE: int = UPPER\_RANGE - LOWER\_RANGE + 1

Define a **public static method** isStringInBounds(plainText: String) -> **boolean**

For each index in plainText:

Declare **character**: char = plainText.charAt(index)

If character < LOWER\_RANGE or character > UPPER\_RANGE:

Return false

Return true

Define a **public static method** caesarEncryption(plainText: String, key: int) -> **String**

If not isStringInBounds(plainText):

Return “The selected string is not in bounds, Try again.”

Declare **encryptedText**: String = “”

While key < 0:

key += RANGE

While key >= RANGE:

key -= RANGE

For each index in plainText:

Declare **encryptedChar**: char = plainText.charAt(index) + key

If encryptedChar > UPPER\_RANGE:

encryptedChar -= RANGE

encryptedText += encryptedChar

Return encryptedText

End of method

Define a **public static method** bellasoEncryption(plainText: String, bellasoStr: String) -> **String**

If not isStringInBounds(plainText):

Return “The selected string is not in bounds, Try again.”

Declare **encryptedText**: String = “”

Declare **bellasoLength**: int = bellasoStr.length()

Declare **keyIndex**: int = 0

For each letter in plainText:

Declare **key**: int = bellasoStr.charAt(keyIndex)

Declare **encryptedChar**: char = plainText.charAt(letter) + key

While encryptedChar > UPPER\_RANGE:

encryptedChar -= RANGE

While encryptedChar < LOWER\_RANGE:

encryptedChar += RANGE

encryptedText += encryptedChar

**Increment** keyIndex

If keyIndex >= bellasoStr.length():

Reset keyIndex to 0

Return encryptedText

Define a **public static method** caesarDecryption(encryptedText: String, key: int) -> **String**

Declare **decryptedText**: String = “”

While key < 0:

key += RANGE

While key >= RANGE:

key -= RANGE

For each index in encryptedText:

Declare **character**: char = encryptedText.charAt(index) - key

If character < LOWER\_RANGE:

character += RANGE

decryptedText += character

Return decryptedText

Define a **public static method** bellasoDecryption(encryptedText: String, bellasoStr: String) -> **String**

Declare **decryptedText**: String = “”

Declare **keyIndex**: int = 0

Declare **bellasoLength**: int = bellasoStr.length()

For each index in encryptedText:

Declare **key**: int = bellasoStr.charAt(keyIndex)

Declare **character**: char = encryptedText.charAt(index) - key

While character < LOWER\_RANGE:

character += RANGE

While character > UPPER\_RANGE:

character -= RANGE

decryptedText += character

**Increment** keyIndex

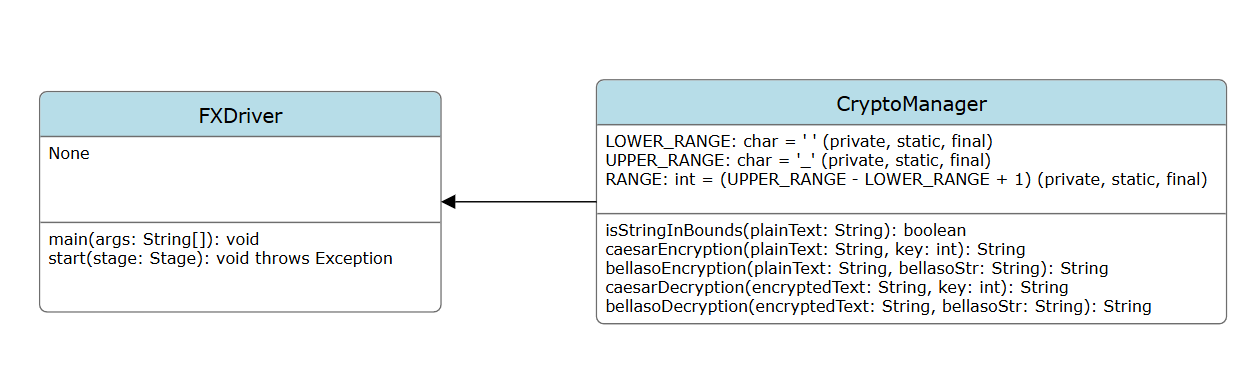
If keyIndex >= bellasoStr.length():

Reset keyIndex to 0

Return decryptedText

End of class

**Part2: UML Class Diagram**

****

**Part3: Comprehensive Test Plan**

A good test plan should be comprehensive. This means you should have a few test cases that test when the input is in and out of range, division by 0, incorrect Data type, etc.(Provide valid and invalid input).

**Notes:**

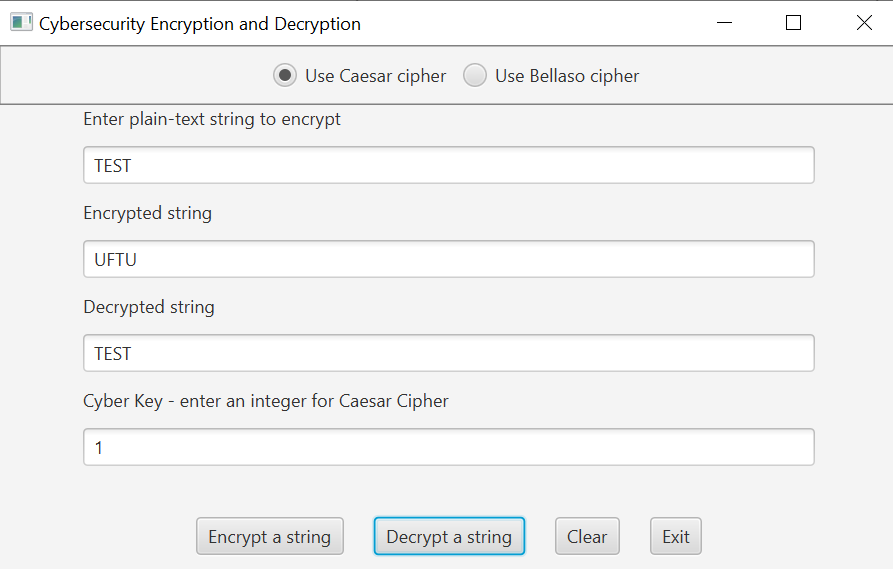
1. **For Design Document: Fill out only the first three columns.**
2. **For Implementation: Complete Test Plan and fill out all columns.**

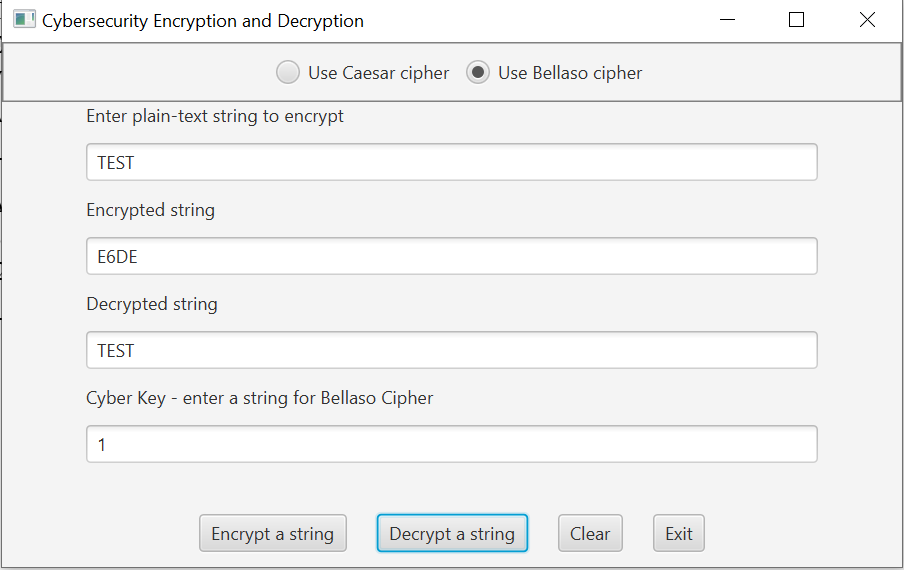
| Cases | Input  Format: (Text, Key) | Expected Output  (Caesar, Bellaso) | Actual Output  (Caesar, Bellaso) | Did Test Pass? |
| --- | --- | --- | --- | --- |
| Case 1 | (“test”, 1) | (“UFTU”, “E6DE”) | (“UFTU”, “E6DE”) | Yes |
| Case 2 | (“HELLO”,105) | (“1.558”, “95A=?”) | (“1.558”, “95A=?”) | Yes |
| Case 3 | (“TESTING”,CIPHER\_IS\_LONGER\_THAN\_THE\_PLAIN\_TEXT) | (“CIPHER\_IS\_LONGER\_THAN\_THE\_PLAIN\_TEXT should be an integer”, “WN#\\N &") | (“CIPHER\_IS\_LONGER\_THAN\_THE\_PLAIN\_TEXT should be an integer”, “WN#\\N &") | Yes |
| Case 4 | (“AいうめじゃASDASDASD”,12121212) | (“The selected string is not in bounds, Try again.”, “The selected string is not in bounds, Try again.”) | (“The selected string is not in bounds, Try again.”, "The selected string is not in bounds, Try again.”) | Yes |

**Part4: Screenshots:**

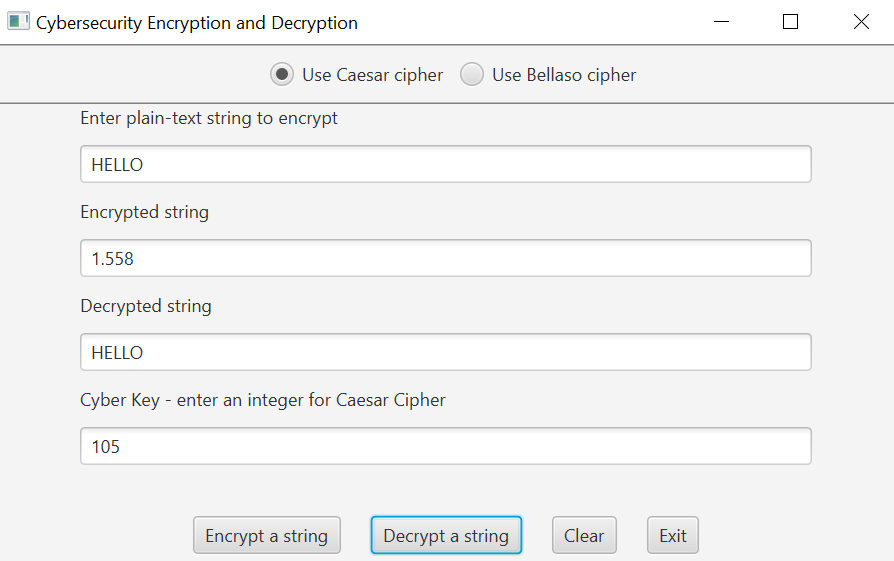
* + - * 1. Screen snapshots of outputs from Eclipse based on your Test Plan

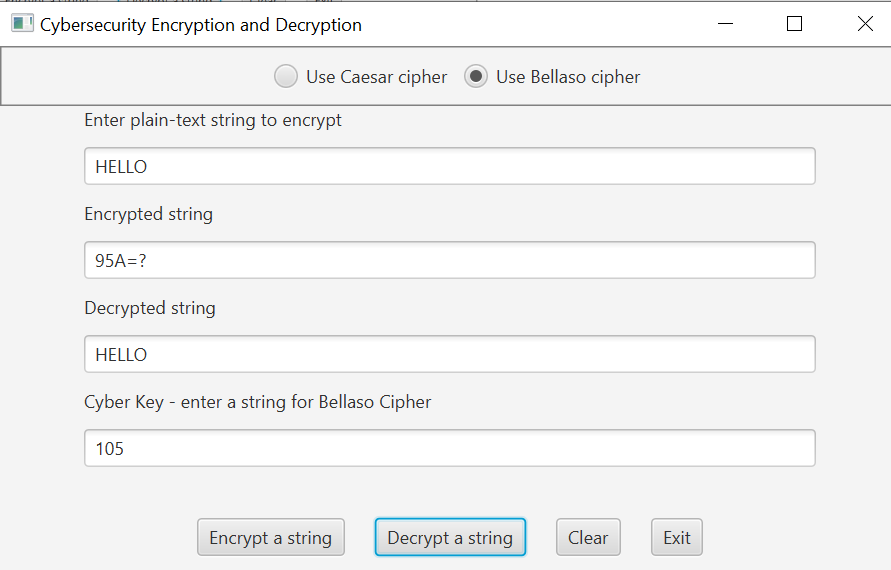
**Case 1**



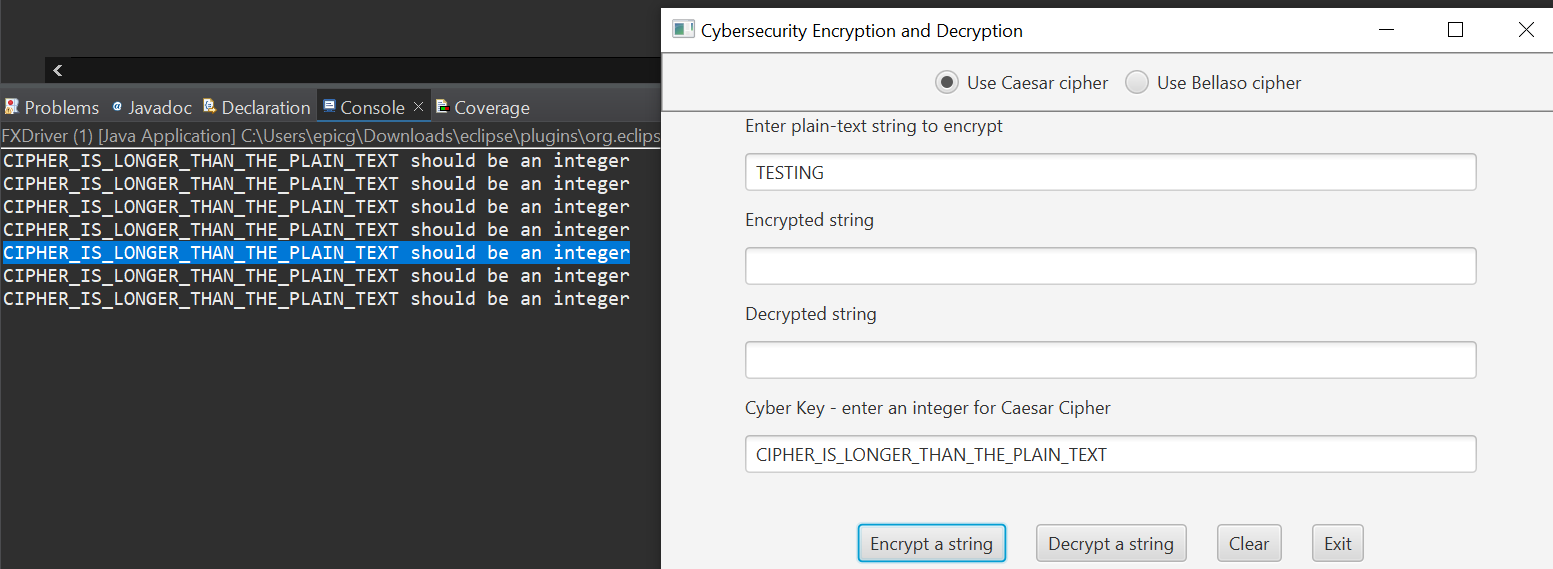


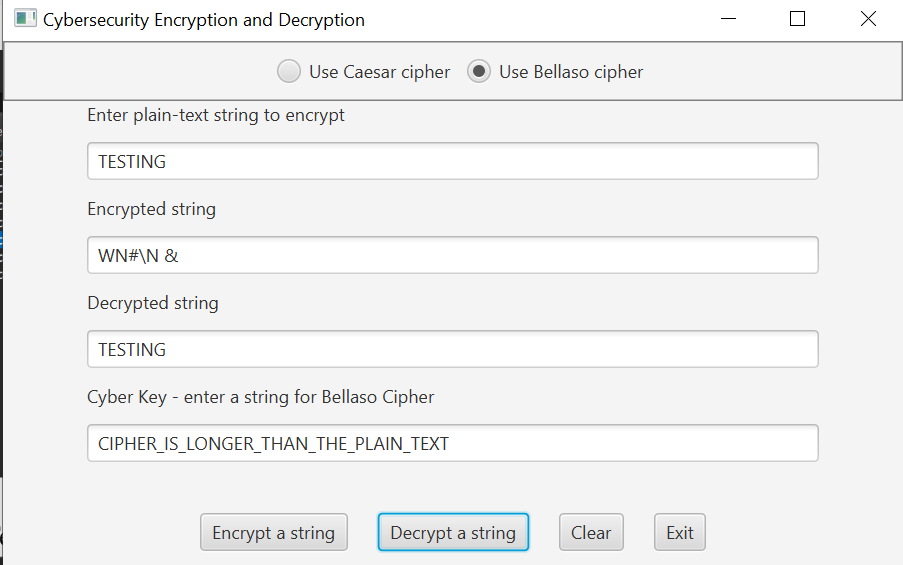
**Case 2**



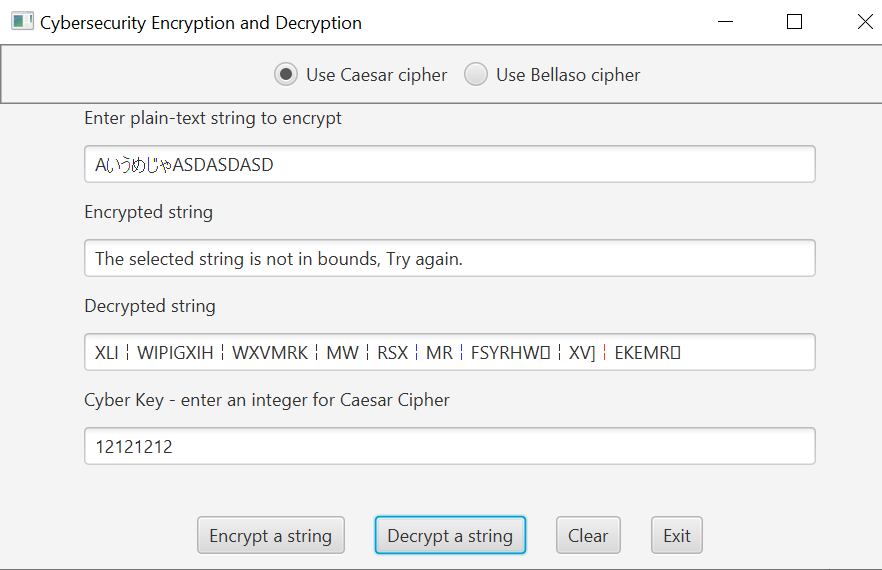


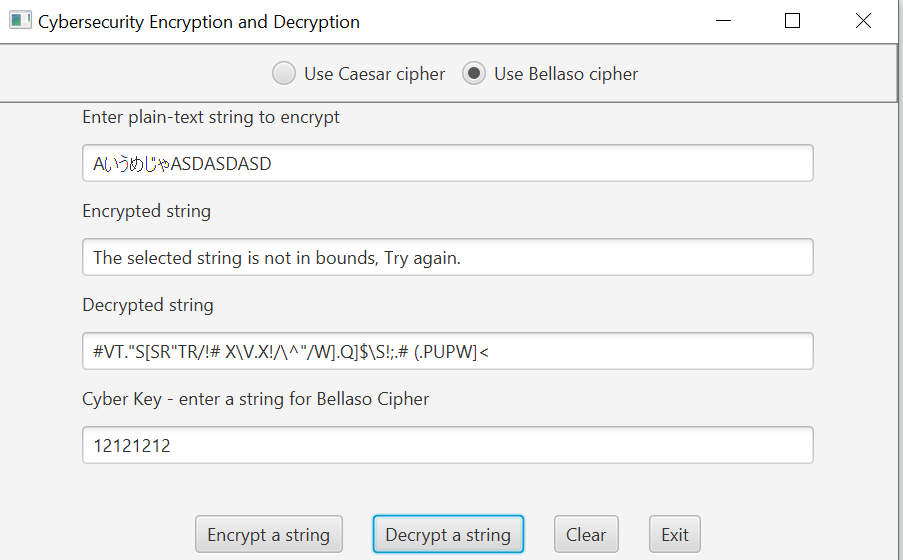
**Case 3**



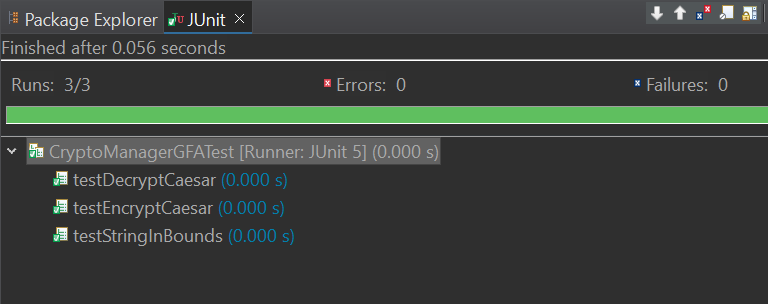


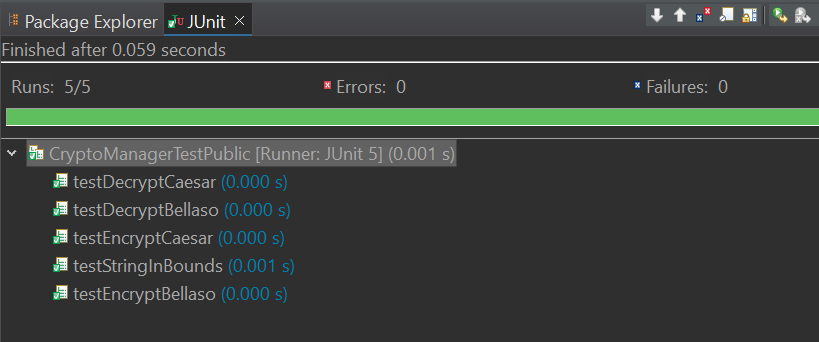
**Case 4**

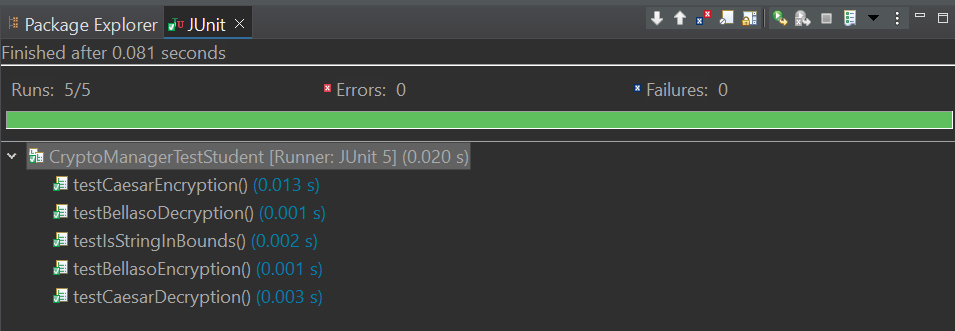




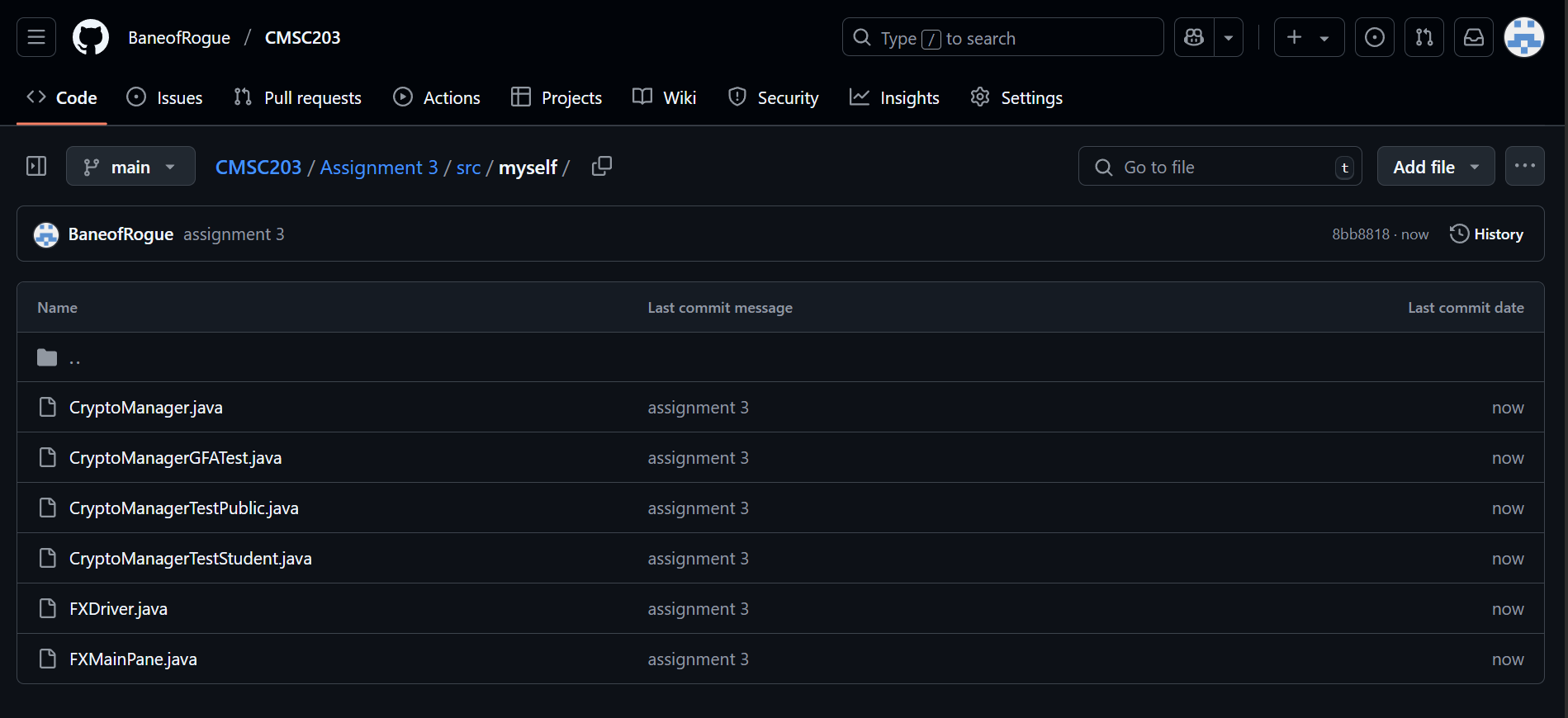
* + - * 1. Screen snapshots of Junit Tests with extended methods’ tests







* + - * 1. Screen shot of src folder files in your GitHub repository.

****

**Lessons Learned:**

**What have you learned?**

How to encrypt and decrypt using caesar and bellaso methods, offsetting and redoing the chars.

**What did you struggle with?**

Getting the logic and my head to wrap around the mixing of the chars \*\_\*

**What would you do differently on your next project?**

I can see how pseudo can be helpful, I guess I could work on my thinking and design aspect.

**What parts of this assignment were you successful with, and what parts (if any) were you not successful with?**

I think this was the trickiest assignment yet. I think the main issue was that encrypting and decrypting isn’t a thing which common people do on a normal basis, which makes it a completely new language for me. I had to sit and think a lot for this one.

**Provide any additional resources/links/videos you used to while working on this assignment/project.**

**The presentation and documents given.**

**Check List::**

| **#** |  | **Y/N** |
| --- | --- | --- |
|  | **Assignment files:** |  |
|  | * FirstInitialLastName\_ Assignment3\_Complete.zip | **Y** |
|  | * FirstInitialLastName\_Assignment3.docx/.pdf | **Y** |
|  | * Source java files | **Y** |
|  | * FirstInitialLastName\_ Assignment3\_JavaFiles.zip | **Y** |
|  | **Program compiles** | **Y** |
|  | **Program runs with desired outputs related to a Test Plan** | **Y** |
|  | **Documentation file:** | **Y** |
|  | * Comprehensive Test Plan | **Y** |
|  | * Screenshots from IDE | **Y** |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **Y** |
|  | * Pseudocode | **Y** |
|  | * Lessons Learned | **Y** |
|  | * Checklist is completed and included in the Documentation | Y |