**Montgomery College**

**CMSC 203**

**Assignment 3 Design**

Class: CMSC203 CRN XXXX

 Program: Assignment 3 Design

Instructor: Farnaz Eivazi

 Summary of Description: Encrypts and Decrypts a phrase using Caesar Cipher and Bellaso Cipher

 Due Date: 7/14/2022

 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: Philip Song

**Part 1: Pseudo Code:**

Turn in pseudo-code for each of the methods specified in CryptoManager.java.   Refer to the [**Pseudocode Guideline**](#PSGdline)on how to write Pseudocode.

1. START
2. User selects Caesar Cipher or Bellaso Cipher
3. User then selects encrypt or decrypt via chosen Cipher
   1. encryptCaesar – encrypts a string according to the Caesar Cipher
4. Prompt user for a string (consisting of characters with ASCII values between 32-95) and an integer value
5. Add the given integer value to ASCII value of each character in the given string
6. Show encrypted word

* 1. decryptCaesar – decrypts a string according to the Caesar Cipher

1. Get an encrypted string and an integer value
2. Subtract integer value from ASCII value of each character in the encrypted string
3. Show decrypted word
   1. encryptBellaso  – encrypts a string according to the Bellaso Cipher
4. Prompt user for a string (consisting of characters with ASCII values between 32-95) and a key word
5. Extend key word to same length of given word
6. Add the ASCII values of corresponding characters in given string and key word
7. Show encrypted word

* 1. decryptBellaso – decrypts a string according to the Bellaso Cipher

1. Get an encrypted string and a key word
2. Extend key word to same length of encrypted string
3. Subtract ASCII values of each character in key word from ASCII values of their corresponding characters in the encrypted string
4. Show decrypted word
5. END

**Part 2: Comprehensive Test Plan**

Turn in a Test Plan table. Test Plan should include:

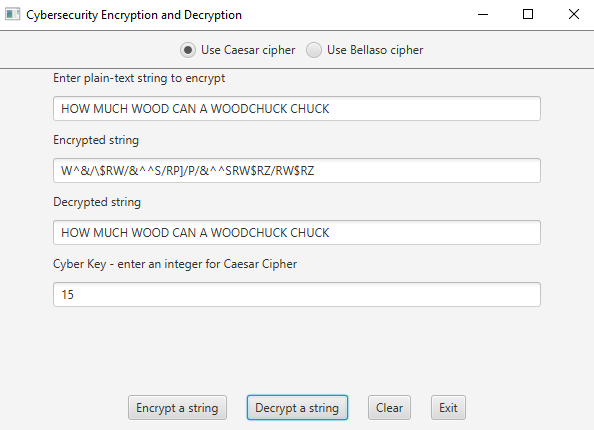
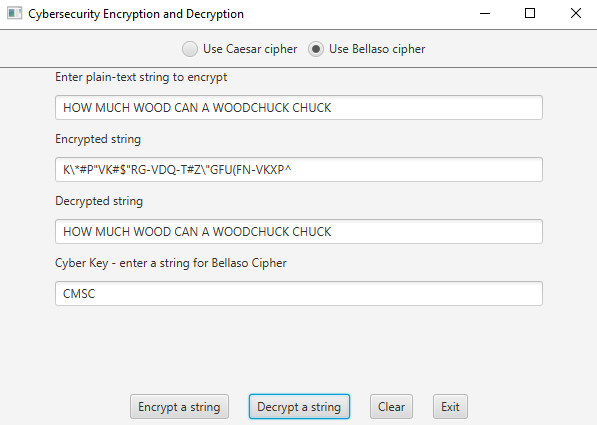
* at least two tests for the Caesar Cipher
* at least two for the Bellaso Cipher.
* at least one string that will fail because it has characters outside the acceptable ones.

**Make sure your tests cover all the possible scenarios.**

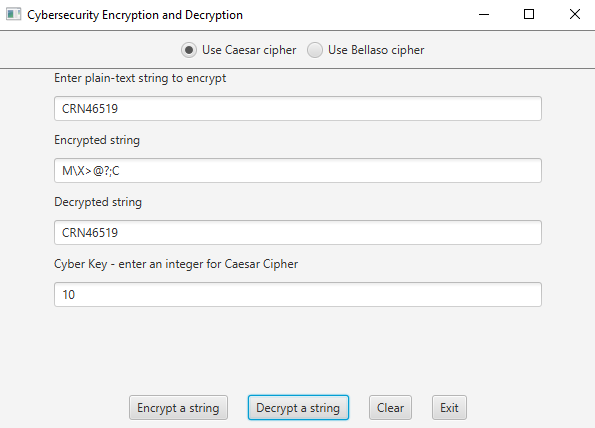
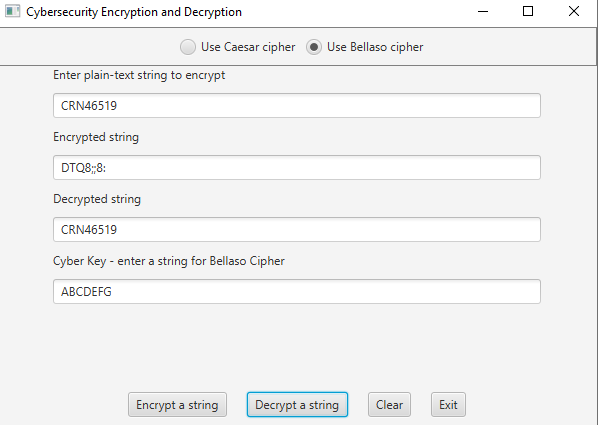
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input text | Input Key | Encrypted (method1) | Encrypted  (method2) | Decrypt (method1) | Decrypt  (method2) |
| HOW MUCH WOOD CAN A WOODCHUCK CHUCK | 15 / CMSC | W^&/\$RW/&^^S/RP]/P/&^^SRW$RZ/RW$RZ | K\\*#P"VK#$"RG-VDQ-T#Z\"GFU(FN-VKXP^ | HOW MUCH WOOD CAN A WOODCHUCK CHUCK | HOW MUCH WOOD CAN A WOODCHUCK CHUCK |
| CRN46519 | 10 / ABCDEFG | M\X>@?;C | DTQ8;;8: | CRN46519 | CRN46519 |
| abc (autocorrects to ABC) | 3 / cmsc (autocorrects to CMSC) | DEF | DOV | ABC | ABC |
| {}|~ | 15 / CMSC | JLKM | >JOA | ;=<> | ;=<> |

**Screenshots for each case listed in the Test Plan**

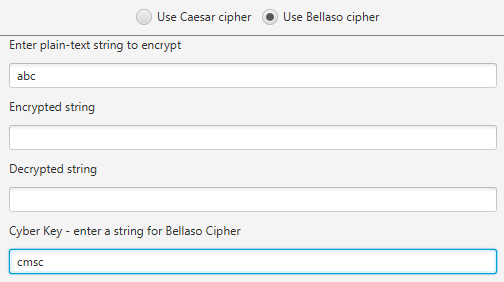
**Case 1:**

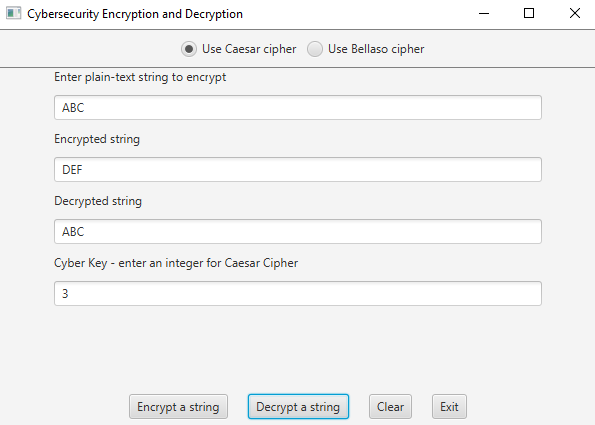
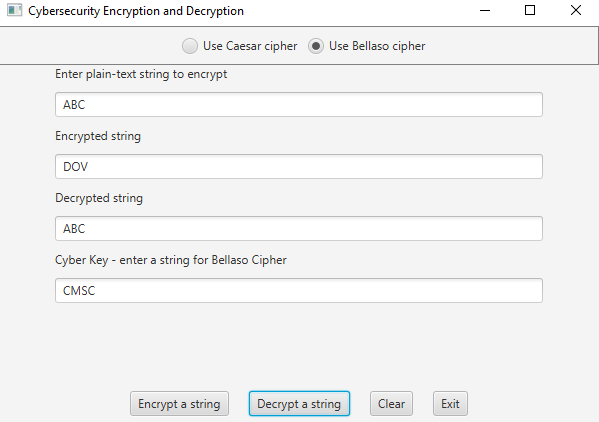
 

**Case 2:**

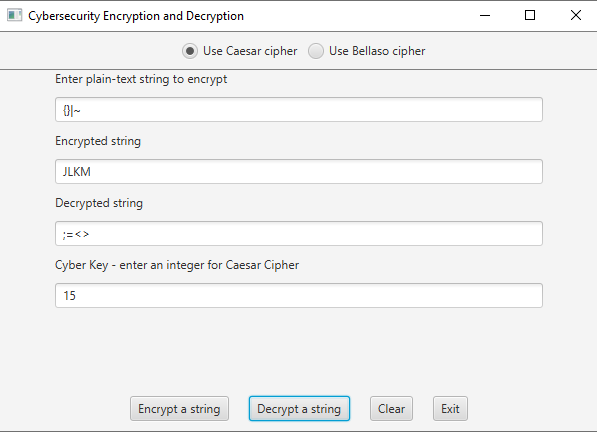
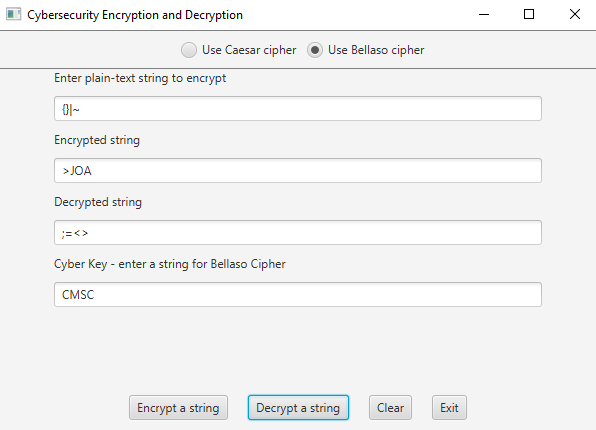
 

**Case 3:**

****

** **

**Case 4:**

**Lessons Learned:**

Write about your Learning Experience, highlighting your lessons learned and learning experience from working on this project.

What have you learned?

* Learned other ways to use for loops, to test for conditions outside of loop
* Learned to use charAt() and toString() methods
* Junit and Test Driver practice

What did you struggle with?

* Writing the pseudocode – I usually jump right into coding
* Design Template not having sections for Screenshots or Lessons Learned
* Code not correctly encrypting/decrypting cipher – I fixed this by applying while loop to repeatedly add/subtract value of RANGE
* Understanding the code that was provided, since we didn’t learn much of it in class

What would you do differently on your next project?

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

**Pseudocode Guideline**

Pseudocode is code written for human understanding­ n­ot a compiler. You can think of pseudocode as “English code,” code that can be understood by anyone (not just a computer scientist). Pseudocode is not language specific, which means that given a block of pseudocode, you could convert it to Java, Python, C++, or whatever language you so desire.

Pseudocode will be important to your future in Computer Science. Typically pseudocode is used to write a high-level outline of an algorithm.

As you may already know, an algorithm is a series of steps that a program takes to complete a specific task. The algorithms can get very complicated without a detailed plan, so writing pseudocode before actually coding will be very beneficial.   
  
**How to Write Pseudocode**

There are no concrete rules that dictate how to write pseudocode, however, there are commonly accepted standards. A reader should be able to follow the pseudocode and hand-simulate (run through the code using paper and pencil) what is going to happen at each step. After writing pseudocode, you should be able to easily convert your pseudocode into any programming language you like.

We use indentation to delineate blocks of code, so it is clear which lines are inside of which method (function), loop, etc. Indentation is crucial to writing pseudocode. Java may not care if you don't indent inside your **if** statements, but a human reader would be completely lost without indentation cues.

**Remember:** Human comprehension is the whole point of pseudocode. So, what does pseudocode look like?

|  |  |
| --- | --- |
| **Pseudocode** | **Real Code in Java** |
| Declare an integer variable called n  Declare an integer variable sum.  Declare an integer variable f1  Declare an integer variable f2  If n is less than 2  sum =n  else  set sum to 0  set f1 and f2 to 1  repeat n times  sum = f1 + f2  f2 = f1  f1 = sum  end loop  print sum | **int** n,k, f1, f2, sum;  **if** ( n < 2 )  sum =n;  **else**  {  sum=0;  f1 = f2 = 1;    **for**(k=2; k<n; k++)  {  sum = f1 + f2;  f2 = f1;  f1 = sum;  }  }  System.***out***.println("Fibonacci of number " + n + " is "+ sum); |

**Finding the Fibonacci numbers till n:**

**Remember that pseudocode is not language specific so we are not looking for “almost Java” code, but instead, we are looking for a strong understanding of the algorithm at hand.**