Pseudo Code for CryptoManager:

1. stringInBounds(plaintext) method returns Boolean
   1. Create char array with the contents of plaintext, calling the toCharArray() method from string.
   2. Create for loop that loops each item in the char array
      1. Check if the char is less than the lower bound, or higher than the upper bound
         1. If it is, return false because it’s not in bounds
   3. Return true because the for loop should return false if it’s not within bounds.
2. encryptCaesar(plaintext, key) method returns String
   1. Make a new variable that takes the value of the key so the key can be modified if it’s out of range
   2. Check if the key is out of range
      1. If it is, subtract the key by the range until it is in range.
   3. Create variable encryptedString that will hold the encrypted text.
   4. Check if the string is in bounds, by calling the stringInBounds method and passing the plaintext string
      1. If it is, make a for loop that iterated every length of the plaintext, i.
         1. Create a new char and set it to the character at index i
         2. Add the new char by the key
         3. Check if the char is greater than the upper bound
            1. If it is, subtract the char by the upperbound + lower bound – 1
         4. Add the char into the encrypted text.
      2. Return the encrypted text.
   5. If it’s not in bounds, return a blank string.
3. encryptBellaso(plaintext, bellasoStr) method returns String
   1. Make a new String array bellasoKey that’s the same length as the length of the plaintext
   2. Make a char array with the contents of plaintext, calling the toCharArray() method from String.
   3. Make a String called encryptionText that will hold the encrypted text.
   4. Check if the string is in bounds, by calling stringInBounds method passing the plaintext string.
      1. If it is, make an for loop that iterates until int variable i is no longer less than the length of plaintext.
         1. Check if i is greater than the length of the bellaso string - 1.
            1. If it is, make the current item i of the bellasoKey String array equal to the index of the remainder of I divided by the length of the bellasoStr. (So if the String is 4 item long, then 5 will be the first item in the String)
            2. If not, make the current item i of the bellasoKey String array equal to the current index i of bellasoStr.
      2. Make another for loop that iterates until i is no longer less than the length of the char array.
         1. Make a new char encryptedChar equal to the current index i of the char array + the current index i of the char value of the bellaso key.
         2. Check if the encryptedChar is greater than the upper bound
            1. If it is, subtract range from encryptedChar
         3. Check if the encryptedChar is less than the lower bound
            1. If it is, add the range to encryptedChar
         4. Add the encrypted char to the encryptedText
      3. Return the encrypted text
   5. Return an empty string because the encryption had some sort of logic error.
4. decryptCaesar(encryptedText, key) method returns String
   1. Make a new variable that takes the value of the key so the key can be modified if it’s out of range
   2. Check if the key is out of range
      1. If it is, subtract the key by the range until it is in range.
   3. Create variable decryptedString that will hold the decrypted text.
   4. Make a for loop that iterated every length of the encryptedText, i.
      1. Create a new char and set it to the character at index i
      2. Subtract the new char by the key
      3. Check if the char is less than the lower bound
         1. If it is, add the char by the range
      4. Add the char into the decryptedText.
   5. Return the decrypted text.
5. decryptBellaso(encryptedText, bellasoStr) method returns String
   1. Make a new String array bellasoKey that’s the same length as the length of the encryptedText
   2. Make a char array with the contents of encryptedText, calling the toCharArray() method from String.
   3. Make a String called decryptedText that will hold the decrypted text.
   4. If it is, make an for loop that iterates until int variable i is no longer less than the length of encrypted text.
      1. Check if i is greater than the length of the bellaso string - 1.
         1. If it is, make the current item i of the bellasoKey String array equal to the index of the remainder of I divided by the length of the bellasoStr. (So if the String is 4 item long, then 5 will be the first item in the String)
         2. If not, make the current item i of the bellasoKey String array equal to the current index i of bellasoStr.
   5. Make another for loop that iterates until i is no longer less than the length of the char array.
      1. Make a new char decryptedChar equal to the current index i of the char array - the current index i of the char value of the bellaso key.
      2. Check if the decryptedChar is greater than the upper bound
         1. If it is, subtract range from decryptedChar
      3. Check if the decryptedChar is less than the lower bound
         1. If it is, add the range to decryptedChar
      4. Add the decrypted char to the decryptedText
   6. Return the decryptedText

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cases | Input | Expected Output | Actual Output | Did it pass? |
| Case 1 | Encryption  Option: Caesar  Plain text:  HELLO  Key:  12 | Encrypted String:  TQXX[ | Encrypted String:  TQXX[ | Y |
| Case 2 | Decryption  Option:  Caesar  Encrypted text:  TQXX[ key: 12 | Decrypted String:  HELLO | Decrypted String:  HELLO | Y |
| Case 3 | Encryption  Option:  Bellaso  Plain text:  RANDOM STUFF  Key:  CMSC | Encrypted String:  UN!GRZ3VW"YI | Encrypted String:  UN!GRZ3VW"YI | Y |
| Case 4 | Decryption  Option:  Bellaso  Encrypted text:  UN!GRZ3VW"YI  Key: CMSC | Decrypted String:  RANDOM STUFF | Decrypted String:  RANDOM STUFF | Y |
| Case 5 | Encryption  Option:  Caesar  Plain text:  ~~INVALID~~  Key: 2 | In GUI:  encryption failed |  | Y |