Lab # 6

*Sothea Song*

*100402876*

CPSC 1150 - 003

Instructor: H. Darbandi

Lab Title: Coder

Date submitted: June 27 2023

Department: CPSC

Program to encrypt message

File Name: Coder.java

Purpose: Used to encrypt the user’s message by first encrypting it using the key provided and convert the cipher text into a binary text

Input: user’s message to encrypt and the key used to encrypt it

Output: encrypted message

Technical Information:

(You should fill the following information based on compiler and computer you are using).

Compiler: IntelliJ IDEA Community Edition 2023.1.1

Computer: (R) Core(TM) i7-10870H CPU @ 2.20GHz 2.21 GHz, 16 GB of RAM

Operating System: Windows 10 Home Single Language

Language: Java

Program Logic (Pseudocode)

**encrypt(String msg, int key)**

Algorithm: Encrypts each character in the message using the key. The key determines how many places to the right the character is moved. It also checks if the encrypted character is larger than the largest value in the character group (lowercase = ‘z’, uppercase = ‘Z’, digit = ‘0’).

(definition)

* 1. msg = message to be encrypted
  2. key = how many places to the right the character is moved

START

1. STRING ecrMsg 🡨 “”
2. FOR i to (length of msg)
   1. temp 🡨 (character in msg at i)
   2. IF temp IS UPPERCASE

ecrMsg 🡨 ecrMsg + char((temp + key – int(‘A’)) % (int(‘Z’ – ‘A’ + 1)) + ‘A’)

* 1. ELIF temp IS LOWERCASE

ecrMsg 🡨 ecrMsg + char((temp + key – int(‘a’)) % (int(‘z’ – ‘a’ + 1)) + ‘a’)

* 1. ELIF temp IS DIGIT

ecrMsg 🡨 ecrMsg + char((temp + key – int(‘0’)) % (int(‘9’ – ‘0’ + 1)) + ‘0’)

* 1. ELSE

EcrMsg 🡨 ecrMsg + char(temp)

1. RETURN ecrMsg

END

**ascii2binary(int n)**

Algorithm: Converts the character by passing it as an integer (ascii value) then converting it binary. It also checks if the length of the binary converted is 8 bits long, if its not then 0 is added to the beginning of the string.

(definition)

0.1 n = The passed integer value to be converted into binary

START

1. convBinary 🡨 Integer.toBinaryString(n)
2. IF (length of convBinary) < 8
   1. FOR i to (8 – length of convBinary)

convBinary 🡨 “0” + convBinary

1. RETURN convBinary

END

**checksDigit(int n)**

Algorithm: Checks how many digits there are in the number and returns how many digits there are in the number

(definition)

0.1 n = value of i that is passed through after each iteration

START

1. count 🡨 0
2. WHILE (n>0)

n 🡨 n/10

count 🡨 count + 1

1. RETURN count

END

**getMsg(Scanner input)**

Algorithm: Asks for the user to input the message to be encrypted

(definition)

START

1. myMessage 🡨 “”
2. OUTPUT “Input Message: “
3. MyMessage 🡨 INPUT
4. RETURN myMessage[]

END

**getKey(Scanner input)**

Algorithm: Asks for the user to input the key

(definition)

START

1. OUTPUT “Input Key: “
2. RETURN INPUT

END

**main()**

Algorithm: Asks for the user to input the key

START

1. msg 🡨 [ ]
2. msg 🡨 getMsg(INPUT)
3. key 🡨 getKey(INPUT)
4. OUTPUT “Key: “ + key
5. stringMsg 🡨 “”
6. FOR k to (length of msg)

stringMsg 🡨 stringMsg + msg[k]

1. encryptedMsg 🡨 encrypt(stringMsg, key)
2. binaryMsg 🡨 “”
3. FOR i to (length of encryptedMsg)

binaryMsg 🡨 binaryMsg + ascii2binary(char at i of encryptedMsg)

1. OUTPUT “Output: “ + binaryMsg

Generate your test cases based on the specifications in your lab assignment. Follow following format for each test case: (Refer to external document of your first lab)

*purpose*

*input*

*output*

*expected value*

*passed or failed*

Test Cases:

Test Case 1: AB C

Input Message: AB C

Input Key: 4

Key : 4

Output:

(Encrypted message = EF G)

01000101010001100010000001000111

Expected:

(Encrypted message = EF G)

01000101010001100010000001000111

Passed

Test Case 2: This is a test .

Input Message: This is a test .

Input Key: 5

Key : 5

Output:

(Encrypted message = Ymnx nx f yjxy.)

01011001011011010110111001111000001000000110111001111000001000000110011000100000011110010110101001111000011110010010000000101110

Expected:

(Encrypted message = Ymnx nx f yjxy.)

01011001011011010110111001111000001000000110111001111000001000000110011000100000011110010110101001111000011110010010000000100001

Passed

Test Case 3: cpsc1150 ? Is my course!

Input Message: cpsc1150 ? Is my course!

Input Key: 8

Key : 8

Output:

(Encrypted message = kxak9938 ? Qa ug kwczam!)

011010110111100001100001011010110011100100111001001100110011100000100000001111110010000001010001011000010010000001110101011001110010000001101011011101110110001101111010011000010110110100100001

Expected:

(Encrypted message = kxak9938 ? Qa ug kwczam!)

011010110111100001100001011010110011100100111001001100110011100000100000001111110010000001010001011000010010000001110101011001110010000001101011011101110110001101111010011000010110110100100001

Passed