UNIVERSITY OF ECONOMICS & FINANCE

HANDS-ON INFORMATION SECURITY LAB MANUAL

Course summary

The course provides knowledge about information security, encryption and decryption algorithms that are commonly used today. In this course, students will get acquainted and practice writing encryption and decryption programs with such classical algorithms as Caesar, Vigenère, Rail Fence, PlayFair, Transposition; In addition, modern algorithms such as DES, 3DES, AES are also provided, finally hash algorithms such as MD5, SHA.

FACULTY OF INFORMATION TECHNOLOGY

For internal circulation only, 2023

Contents

Practice 1: CLASSICAL SYMMETRIC ENCRYPTION

1.1 OVERVIEW

1.1.1 Introduction

- Lab 1: Caesar Encryption Algorithm
- Practice time: class: 3 study hours, self-study: 3 study hours.
- Requirements: Students using Netbeanss Software

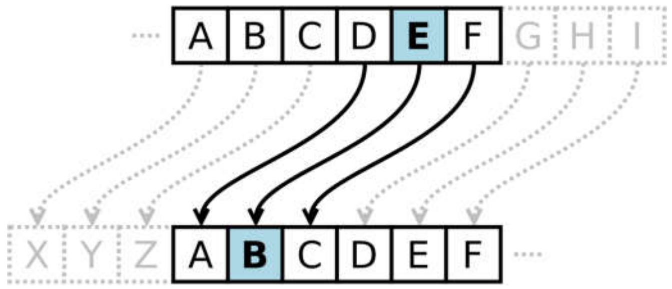
1.1.2 Objective

- This course provides students with knowledge of cryptographic algorithms and how they are used in today's world.
- The content emphasizes the principles, topics, approaches, and problem solving related to the underlying technologies and architectures of the field.

1.2 CONTENTS

1.2.1 Basic knowledge

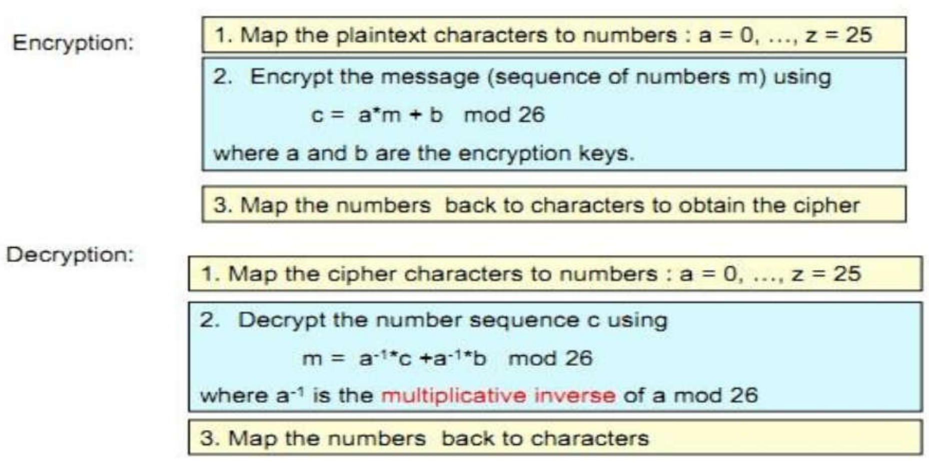
One of the simplest and most well-known encryption methods is the Caesar cipher, often known as Caesar's cipher, the shift cipher, Caesar's code, or Caesar shift. Each letter in the plaintext is replaced by a letter that is located a certain number of places farther down the alphabet in this form of substitution cipher. With a left shift of 3, for instance, D would become A, E would become B, and so on. Julius Caesar, who employed it in his personal communications, gave the approach its name.



– We in turn index the letters starting from 0.
– Let "k" be an integer from 0 -> 25 called key.

– Encryption function: E(p,k)=(p+k)mod26 where p is the index of the character to be encoded.

– Decryption function: D(c,k)=|c-k|mod26 where c is the index of the character to be decoded



Encryption:

1. Map the plaintext characters to numbers : a = 0, …, z = 25

2. Encrypt the message (sequence of numbers m) using

$$c = a*m + b \mod 26$$

where a and b are the encryption keys.

3. Map the numbers back to characters to obtain the cipher

Decryption:

1. Map the cipher characters to numbers : a = 0, …, z = 25

2. Decrypt the number sequence c using

$$m = a^{-1}*c + a^{-1}*b \mod 26$$

where a^{-1} is the multiplicative inverse of a mod 26

3. Map the numbers back to characters

1.2.2 Caesar Cipher

Write a program to encrypt and decrypt text with Caesar encryption algorithm. The program can perform the following functions:

– Allow text input into the system.

– Allows entering text protection keys.

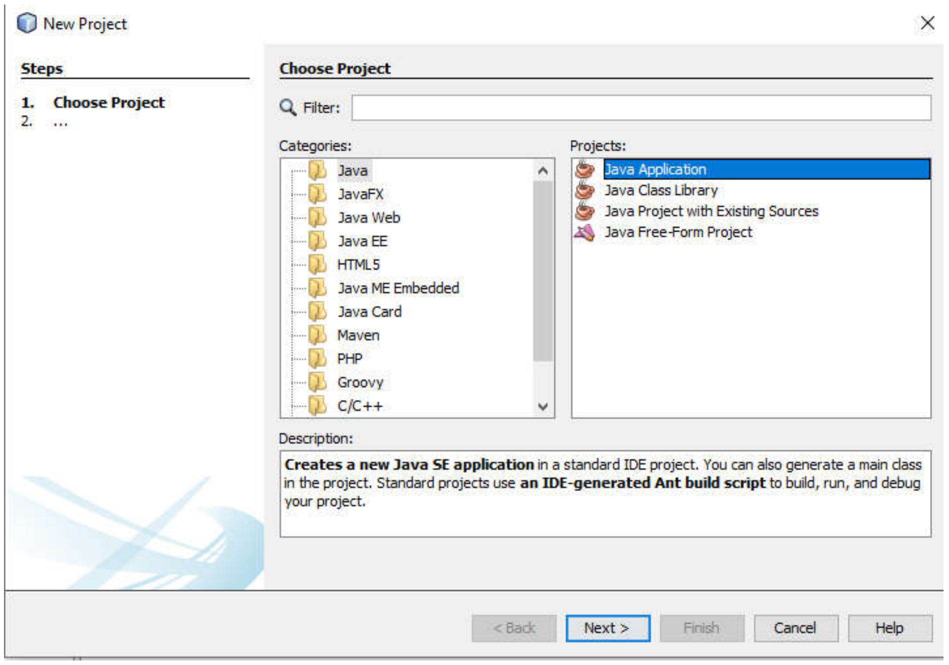
– Allows to write File and open File.

Guide:

❖ **Step 1: Starting Netbeans sofware → File → New Project → Next:**

– Categories: Java

– Project: Java Application

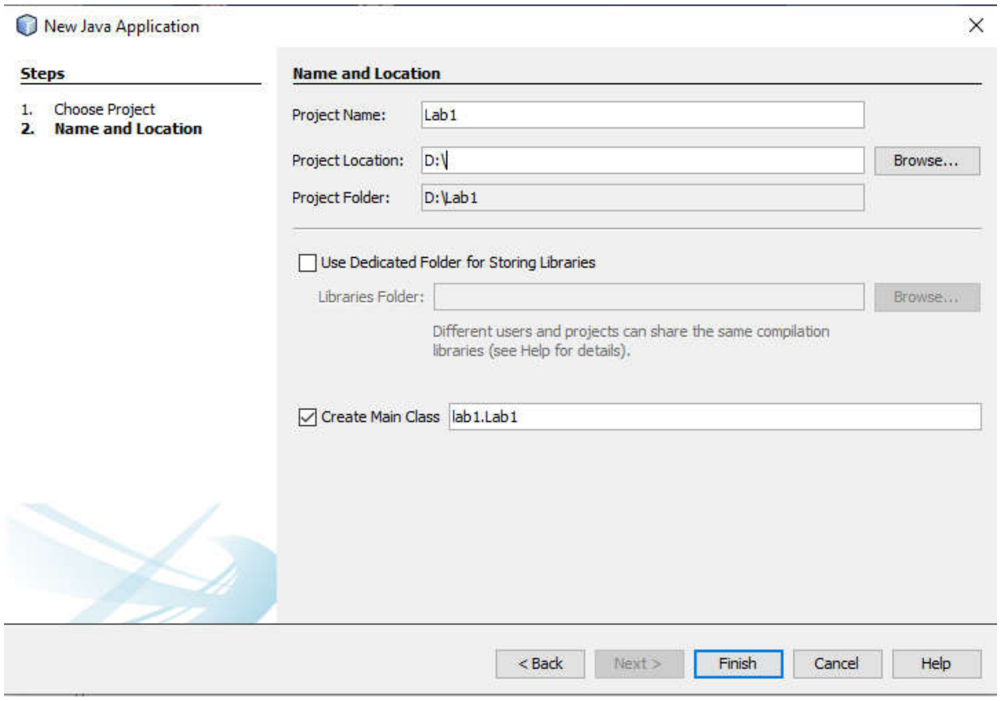


Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

❖ **Step 2: Tying for project → Finish:**

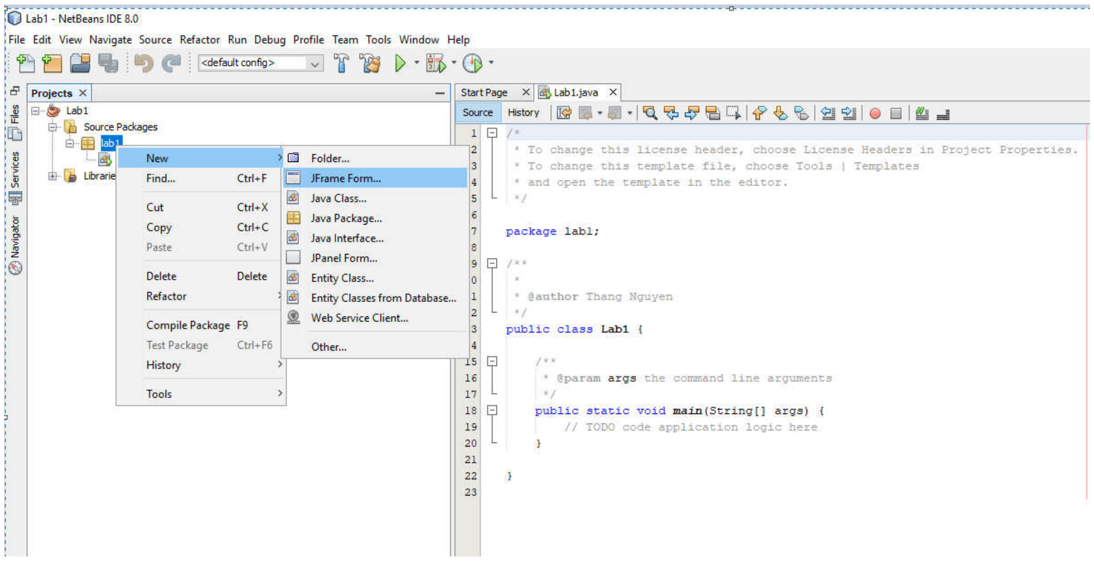
– Project Name: Lab1

– Project Location: Optional



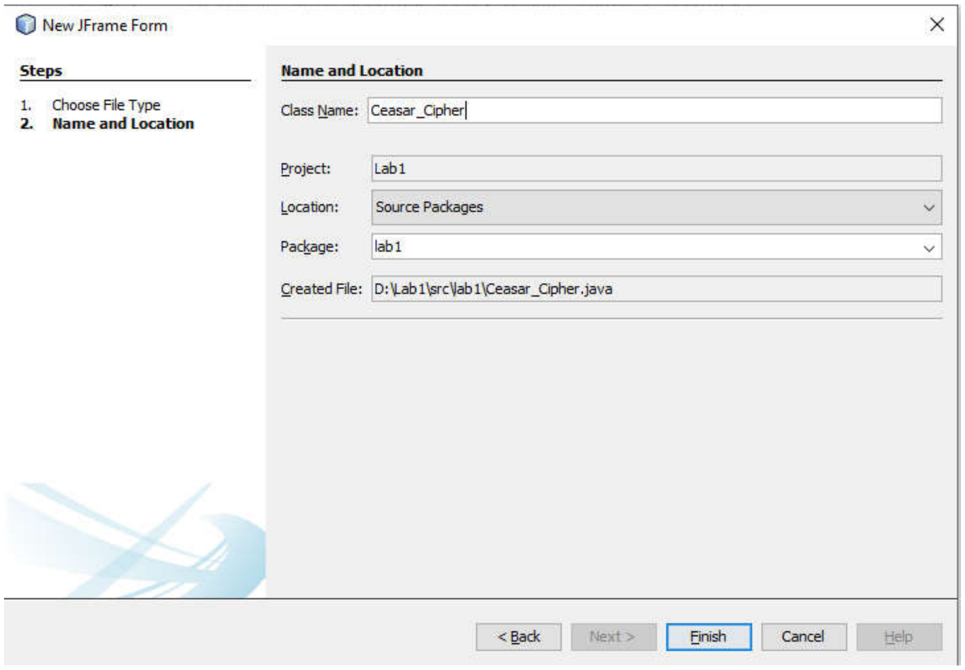
Project Name: Lab1
Project Location: D:\
Project Folder: D:\Lab1
☐ Use Dedicated Folder for Storing Libraries
Different users and projects can share the same compilation libraries (see Help for details).
☑ Create Main Class lab1.Lab1

❖ **Step 3: Create new jFrame Form:**



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package lab1;

/**
 *
 * @author Thang Nguyen
 */
public class Lab1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

❖ **Step 4: Typing Class Name: Caesar_Cipher**



Class Name: Ceasar_Cipher

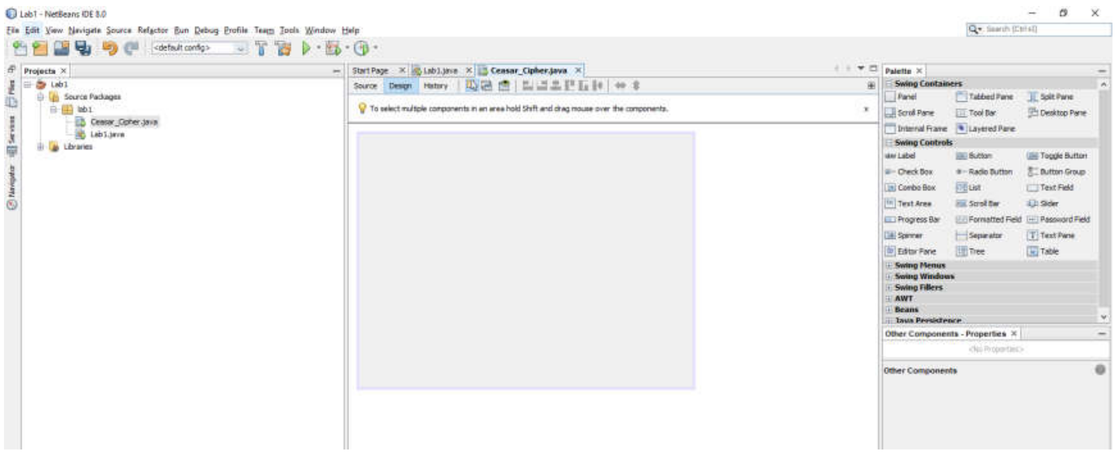
Project: Lab1

Location: Source Packages

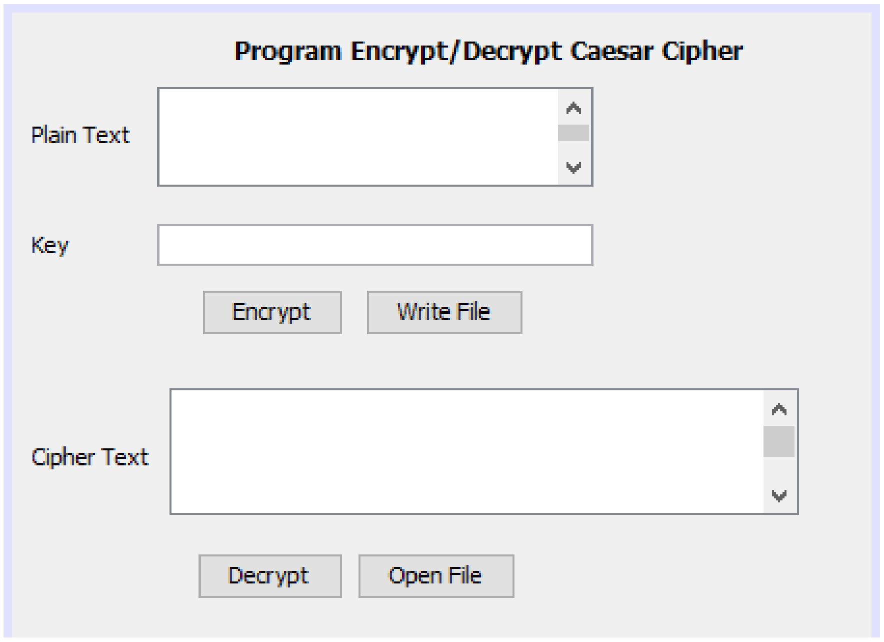
Package: lab1

Created File: D:\Lab1\src\lab1\Ceasar_Cipher.java

❖ **Step 5: Design Form:**



❖ **Step 6: Design follow:**

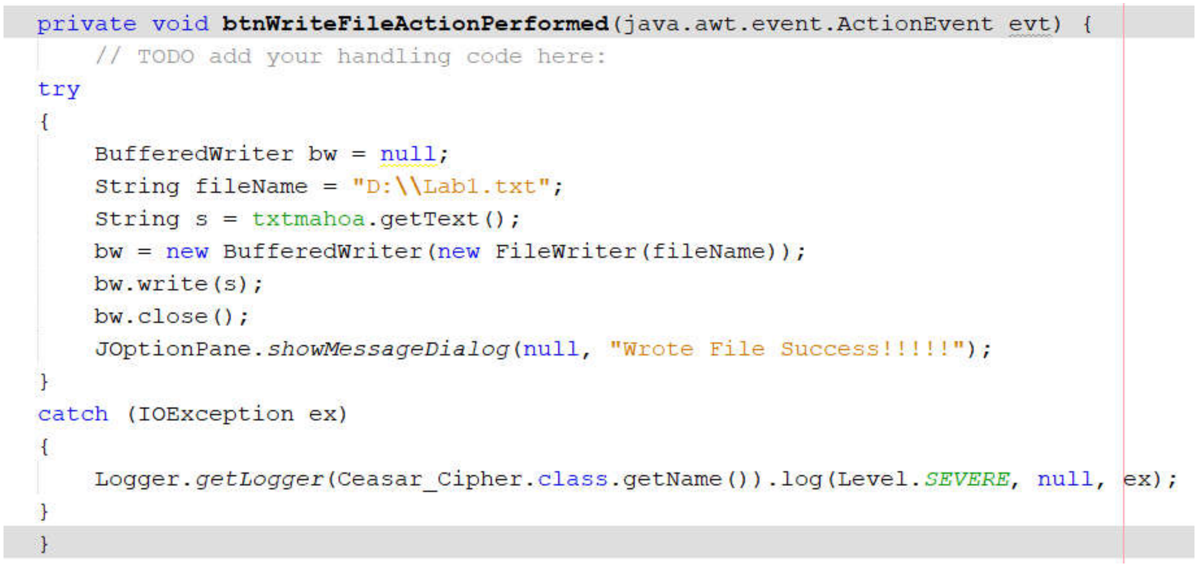


❖ **Step 7: Write an event handler function:**

- **7.1 Encrypt function:**

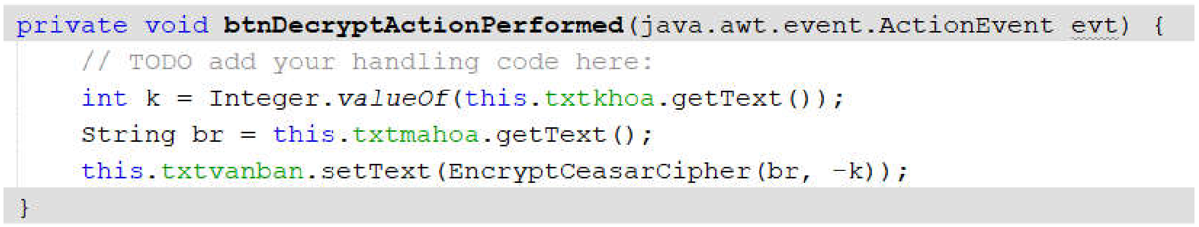
```
private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int k = Integer.valueOf(this.txtkhoa.getText());
    String br = this.txtvanban.getText();
    this.txtmahoa.setText(EncryptCeasarCipher(br,k));
}
```

- **7.2 Write File:**

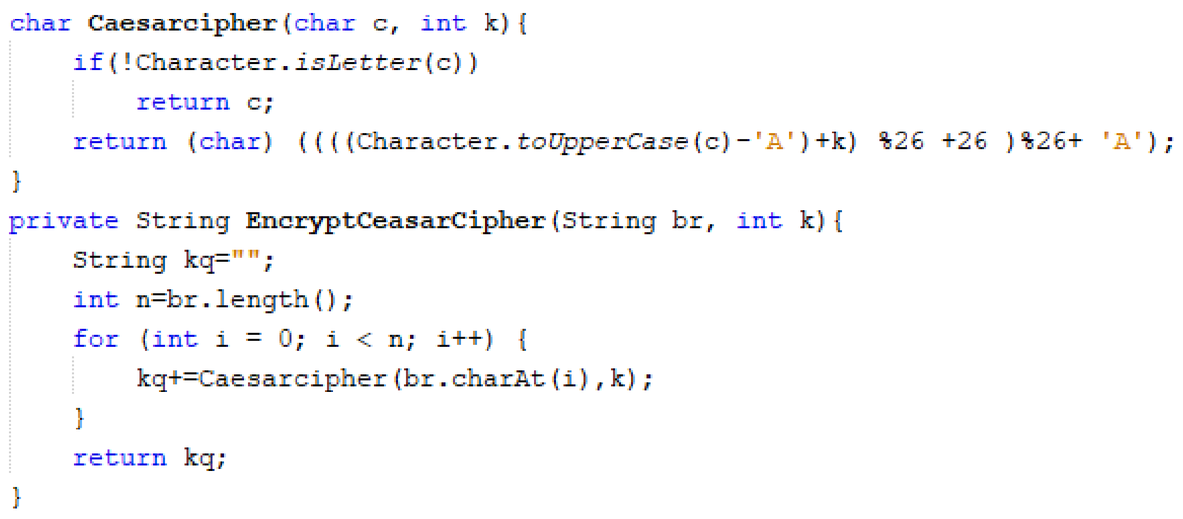


```
private void btnWriteFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
try
{
    BufferedWriter bw = null;
    String fileName = "D:\\Lab1.txt";
    String s = txtmahoa.getText();
    bw = new BufferedWriter(new FileWriter(fileName));
    bw.write(s);
    bw.close();
    JOptionPane.showMessageDialog(null, "Wrote File Success!!!!!");
}
catch (IOException ex)
{
    Logger.getLogger(Ceasar_Cipher.class.getName()).log(Level.SEVERE, null, ex);
}
}
```

- **7.3 Decrypt function**



```
private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int k = Integer.valueOf(this.txtkhoa.getText());
    String br = this.txtmahoa.getText();
    this.txtvanban.setText(EncryptCeasarCipher(br, -k));
}
```

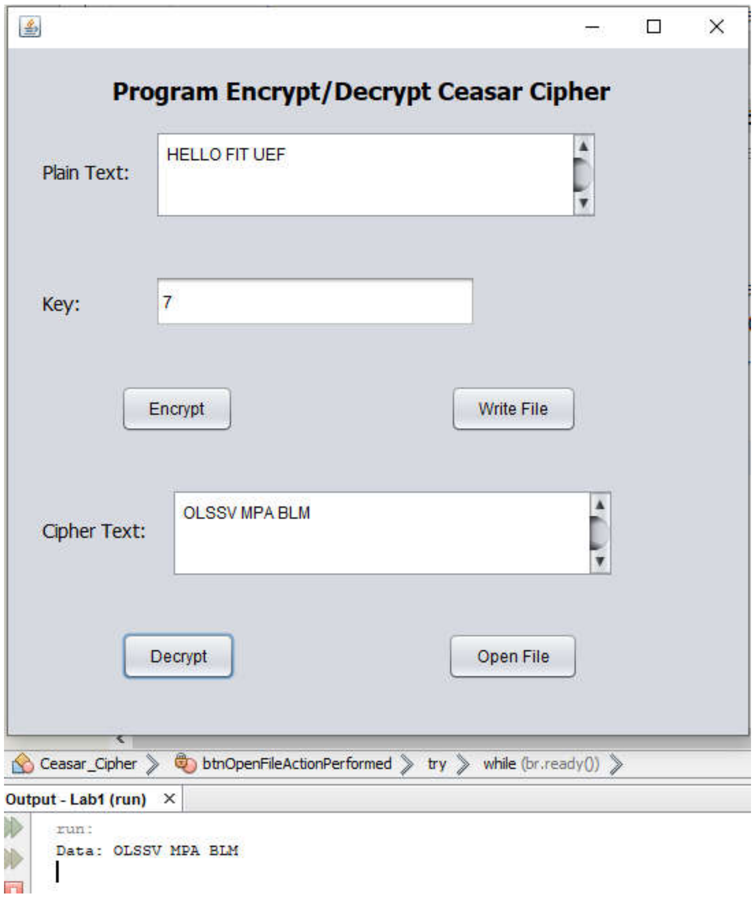


```
char Caesarcipher(char c, int k){
    if(!Character.isLetter(c))
        return c;
    return (char) ((((Character.toUpperCase(c)-'A')+k) %26 +26 )%26+ 'A');
}
private String EncryptCeasarCipher(String br, int k){
    String kq="";
    int n=br.length();
    for (int i = 0; i < n; i++) {
        kq+=Caesarcipher(br.charAt(i),k);
    }
    return kq;
}
```

- **7.4 Open File:**

```
private void btnOpenFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
try{
    BufferedReader br = null;
    String fileName = "D:\\Data.txt";
    br = new BufferedReader(new FileReader(fileName));
    StringBuffer sb = new StringBuffer();
    JOptionPane.showMessageDialog(null, "Opened File Success!!!!!!!!!!");
    char[] ca = new char[5];
    while(br.ready())
    {
        int len = br.read(ca);
        sb.append(ca, 0, len);
    }
    br.close();
    System.out.println("Data:" + " " + sb);
    String chuoi = sb.toString();
    txtvanban.setText(chuoi);
} catch (IOException ex) {
    Logger.getLogger(Ceasar_Cipher.class.getName()).log(Level.SEVERE, null, ex);
}
}
```

Result:


Program Encrypt/Decrypt Ceasar Cipher
Plain Text:
Key:
Encrypt
Write File
Cipher Text:
Decrypt
Open File
Ceasar_Cipher > btnOpenFileActionPerformed > try > while (br.ready())
Output - Lab1 (run)

Plain Text: HELLO FIT UEF

Key: 7

Cipher Text: OLSSV MPA BLM

```
run:
Data: OLSSV MPA BLM
```