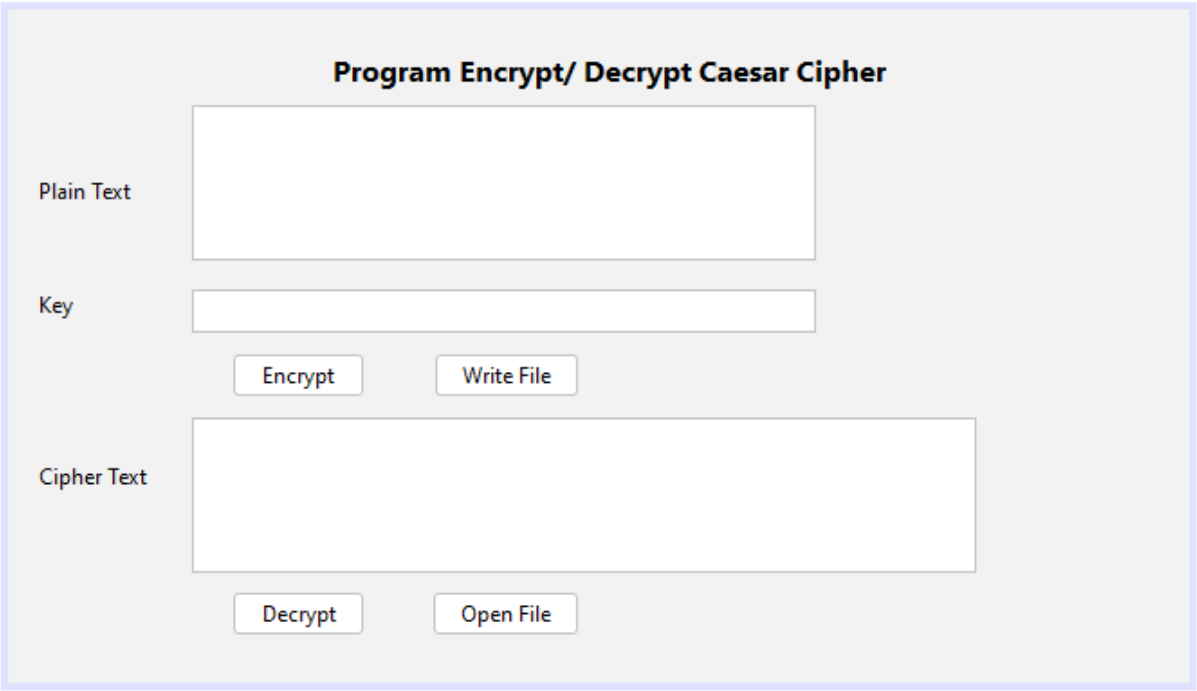
LAB1: Caesar_Cipher



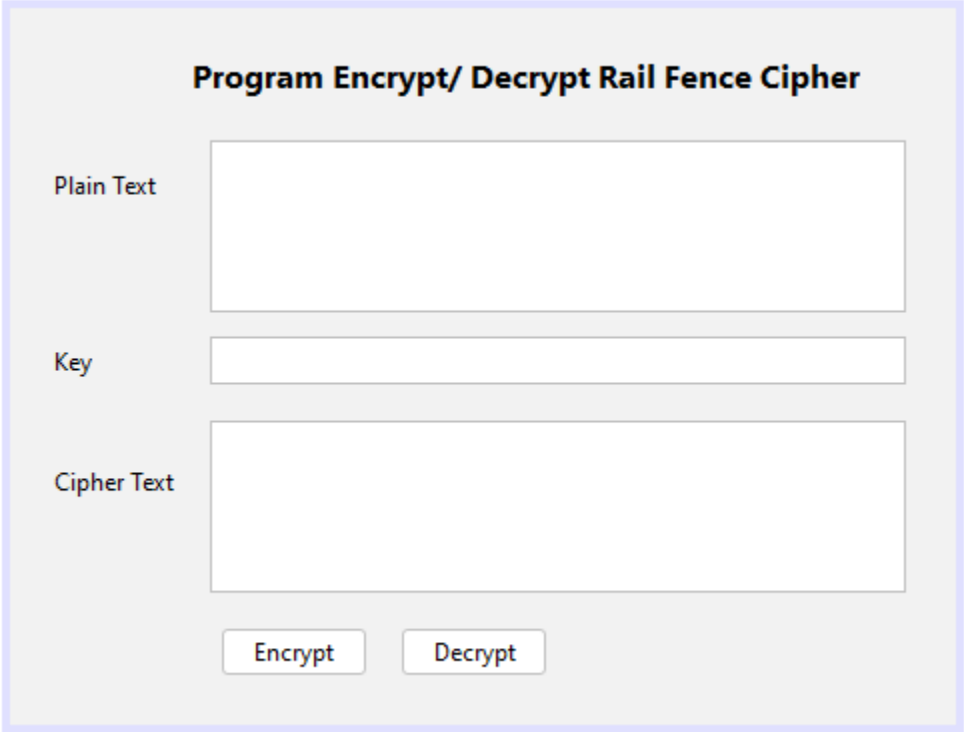

```
package lab1;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;
import java.util.logging.*;
public class Caesar_Cipher extends javax.swing.JFrame {
   public Caesar_Cipher() {
      initComponents();
   }
   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(EncryptCaesarCipher(br, k));
}

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(Caesar_Cipher.class.getName()).log(Level.SEVERE, null,ex);
    }
}

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(EncryptCaesarCipher(br, -k));
}

private void btnOpenFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try
```

```
    {
        BufferedReader br = null;
        String fileName = "D:\\Lab1.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();
        this.txtVanBan.setText(chuoi);
    }
    catch (IOException ex)
    {
        Logger.getLogger(Caesar_Cipher.class.getName()).log(Level.SEVERE, null,ex);
    }
}

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 EncryptCaesarCipher(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;
    }
}
```

LAB2_1: Rail_Fence



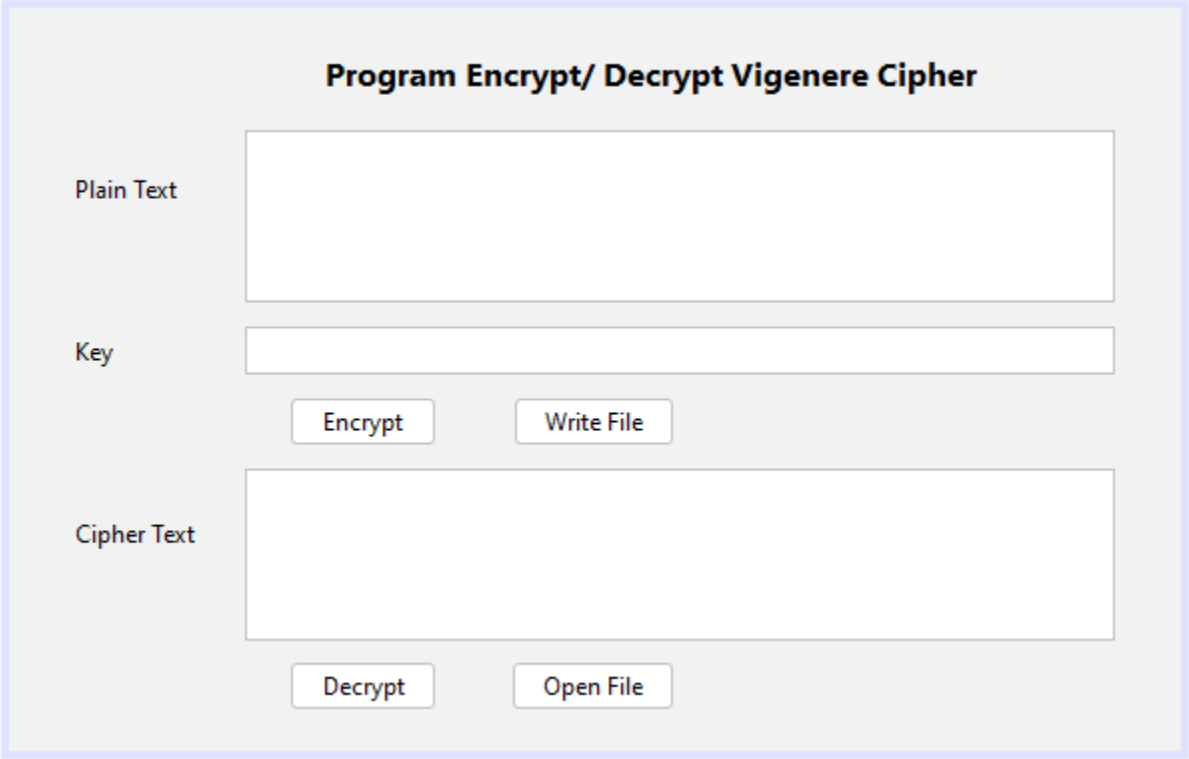

```
package lab2;
public class Rail_Fence extends javax.swing.JFrame {
    public Rail_Fence() {
        initComponents();
    }
    private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
```

```
int k = Integer.valueOf(this.txtKey.getText());
String s = this.txtPlain.getText();
int n = s.length();
int sd, sc;
sd = k;
sc = n / sd + 1;
char hr[][] = new char[sd][sc];
int c,d;
c = 0;
d = 0;
int sodu = n % sd;
for(int i = 0; i < n; i++)
{
   hr[d][c] = s.charAt(i);
   d++;
   if(d == k)
   {
      c++;
      d = 0;
   }
}
String kq = "";
int sokytu = sc;
for(int i = 0; i < sd; i++)
{
   if (i>= sodu)
      sokytu = sc - 1;
   for(int j = 0; j < sokytu; j++)
      kq = kq + hr[i][j];
}
this.txtCipher.setText(kq);
```

```
}

private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int k = Integer.valueOf(this.txtKey.getText());
    String s = this.txtCipher.getText();
    int n = s.length();
    int sd, sc;
    sd = k;
    sc = n / sd + 1;
    int sodu = n % sd;
    int sokytu = sc;
    int t = 0;
    String kq = "";
    char hr[][] = new char[sd][sc];
    for(int i = 0; i < sd; i++)
    {
        if(i >= sodu)
            sokytu = sc - 1;
        for(int j = 0; j < sokytu; j++){
            hr[i][j] = s.charAt(t);
            t++;
        }
    }
    int c,d;
    c = 0;
    d = 0;
    for(int i = 0; i < n; i++)
    {
        kq += hr[d][c];
        d++;
```

```
            if (d == k)
            {
                c++;
                d = 0;
            }
        }
        this.txtPlain.setText(kq);
    }
}
```

LAB2_2: Vigenere_Cipher




```
package lab2;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
```

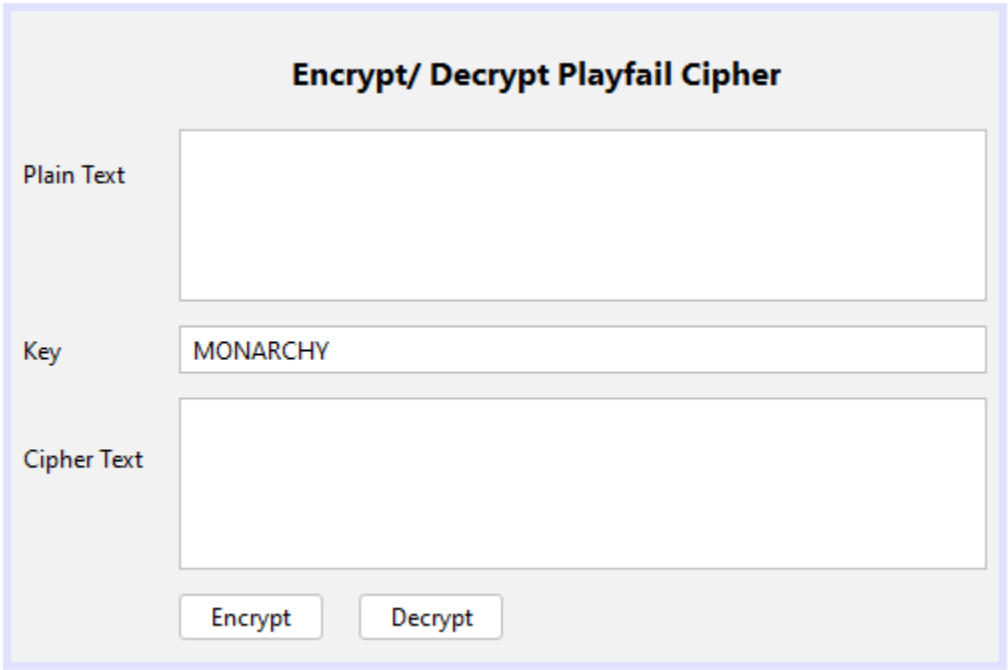
```
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;
import java.util.logging.*;
public class Vigenere_Cipher extends javax.swing.JFrame {
   int Vig[][];
   public Vigenere_Cipher() {
      initComponents();
      Vig = new int[26][26];
      for (int i = 0; i < 26; i++)
         for (int j = 0; j < 26; j++)
            Vig[i][j] =(i + j) % 26;
   }

   private String Encryption(String plainText, String key)
   {
      int n =  plainText.length();
      String CipherText = "";
      int k = 0;
      for(int i = 0; i < n; i++){
         if(Character.isLetter(plainText.charAt(i)))
         {
            CipherText += Encrypt(plainText.charAt(i), key.charAt(k));
            k++;
            k = k%key.length();
         }
         else{
            CipherText += plainText.charAt(i);
         }
      }
      return CipherText;
```

```
}

char Encrypt(char x, char k){
    int xn = Character.toUpperCase(x) - 'A';
    int kn = Character.toUpperCase(k) - 'A';
    int yn = Vig[kn][xn];
    return (char) (yn + 'A');
}

private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String plainText = this.txtPlain.getText();
    String k = this.txtKey.getText();
    String CipherText = Encryption(plainText, k);
    this.txtCipher.setText(CipherText);
}

private void btnWriteFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try
    {
        BufferedWriter bw = null;
        String fileName = "D:\\Lab2.txt";
        String s = txtPlain.getText();
        bw = new BufferedWriter(new FileWriter(fileName));
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, "Wrote File Success!!!");
    }
    catch (IOException ex)
    {
```

```
        Logger.getLogger(Vigenere_Cipher.class.getName()).log(Level.SEVERE, null,ex);
    }
}

private void btnOpenFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try
    {
        BufferedReader br = null;
        String fileName = "D:\\Lab2.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();
        txtPlain.setText(chuoi);
    }
    catch (IOException ex)
    {
        Logger.getLogger(Vigenere_Cipher.class.getName()).log(Level.SEVERE, null,ex);
    }
}

private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        // TODO add your handling code here:
        String CipherText = this.txtPlain.getText();
        String k = this.txtKey.getText();
        String kt1 = "";
        int kn = k.length();
        for(int i = 0; i < kn; i++)
        {
            kt1 += (char)(((26-(Character.toUpperCase(k.charAt(i)) - 'A')) % 26) + 'A');
        }
        this.txtKey.setText(kt1);
        String PlainText = Encryption(CipherText, kt1);
        this.txtPlain.setText(PlainText);
    }
}
```

LAB3_1: PlayFail_Cipher



```
package lab3;
public class PlayFail_Cipher extends javax.swing.JFrame {
    char pf[][] = {{'M','O','N','A','R'},
                   {'C','H','Y','B','D'},
                   {'E','F','G','I','K'},
                   {'L','P','Q','S','T'},
                   {'U','V','W','X','Z'}
                 };
    public PlayFail_Cipher() {
        initComponents();
        this.txtKey.disable();
    }

    private String Encrypt(String banro){
        int n = banro.length();
        int i = 0;
        String banma = "";
        char a,b;
        while (i < n){
            if (i == n - 1){
                a = banro.charAt(i);
                b = 'X';
                i++;
            }
            else{
                a = banro.charAt(i);
                b = banro.charAt(i+1);
                if(a == b){
                    b = 'X';
                    i++;
                }
```

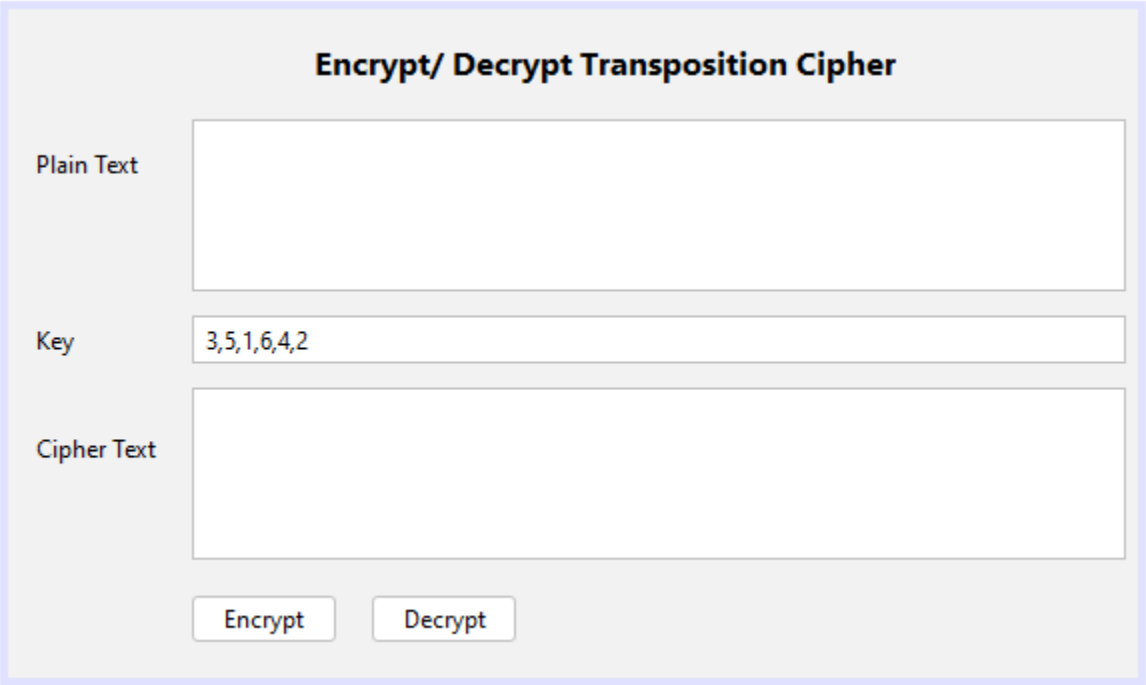
```
            else
                i += 2;
        }
        banma += Replace(a,b);
    }
    return banma;
}

String Replace(char a, char b)
{
    String vta = FindLoacation(a);
    String vtb = FindLoacation(b);
    char x,y;
    if (vta.charAt(0) == vtb.charAt(0)){
        x = pf[vta.charAt(0) - '0'][((vta.charAt(1) - '0') + 1) % 5];
        y = pf[(vtb.charAt(0) - '0')][((vtb.charAt(1) - '0') + 1) % 5];
        return x + "" + y;
    }
    if (vta.charAt(1) == vtb.charAt(1)){
        x = pf[((vta.charAt(0) - '0') + 1) % 5][(vta.charAt(1) - '0')];
        y = pf[((vtb.charAt(0) - '0') + 1) % 5][(vtb.charAt(1) - '0')];
        return x + "" + y;
    }
    x = pf[(vta.charAt(0) - '0')][(vtb.charAt(1) - '0')];
    y = pf[(vtb.charAt(0) - '0')][(vta.charAt(1) - '0')];
    return x + "" + y;
}

private String FindLoacation(char a)
{
    for (int  i = 0; i < 5; i ++){
```

```
        for (int j = 0; j <5; j++){
            if (pf[i][j] == a){
                return i + "" + j;
            }
        }
    }
    return "";
}

private String Decrypt(String banma)
{
    int n = banma.length();
    String banro = "";
    char a,b;
    for(int i = 0; i < n; i += 2){
        a = banma.charAt(i);
        b = banma.charAt(i+1);
        banro += ReverseReplace(a, b);
    }
    return banro;
}

String ReverseReplace(char a, char b){
    String vta = FindLoacation(a);
    String vtb = FindLoacation(b);
    char x,y;
    if (vta.charAt(0) == vtb.charAt(0)){
        x = pf[vta.charAt(0) - '0'][((vta.charAt(1) - '0') - 1 + 5) % 5];
        y = pf[(vtb.charAt(0) - '0')][((vtb.charAt(1) - '0') - 1 + 5) % 5];
        return x + "" + y;
    }
```

```
    if (vta.charAt(1) == vtb.charAt(1)){
        x = pf[((vta.charAt(0) - '0') - 1 + 5) % 5][(vta.charAt(1) - '0')];
        y = pf[((vtb.charAt(0) - '0') - 1 + 5) % 5][(vtb.charAt(1) - '0')];
        return x + "" + y;
    }
    x = pf[(vta.charAt(0) - '0')][(vtb.charAt(1) - '0')];
    y = pf[(vtb.charAt(0) - '0')][(vta.charAt(1) - '0')];
    return x + "" + y;
}
private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String banro = this.txtPlainText.getText();
    banro = banro.toUpperCase();
    banro = banro.replace('J', 'I');
    String banma = Encrypt(banro);
    this.txtCipherText.setText(banma);
}

private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String banma = this.txtCipherText.getText();
    String banro = Decrypt(banma);
    int n = banro.length();
    String br = "";
    for(int i = 0; i < n - 2; i += 2){
        if(banro.charAt(i) == banro.charAt(i+2)){
            br += banro.charAt(i);
        }
        else{
            br += banro.charAt(i) + "" + banro.charAt(i+1);
        }
```

```
        }
        if( banro.charAt(n-1) == 'X'){
            br += banro.charAt(n-2);
        }
        else{
            br += banro.charAt(n-2);
            br += banro.charAt(n-1);
        }
        this.txtPlainText.setText(br);
    }
}
```

LAB3_2: Transposition_Cipher



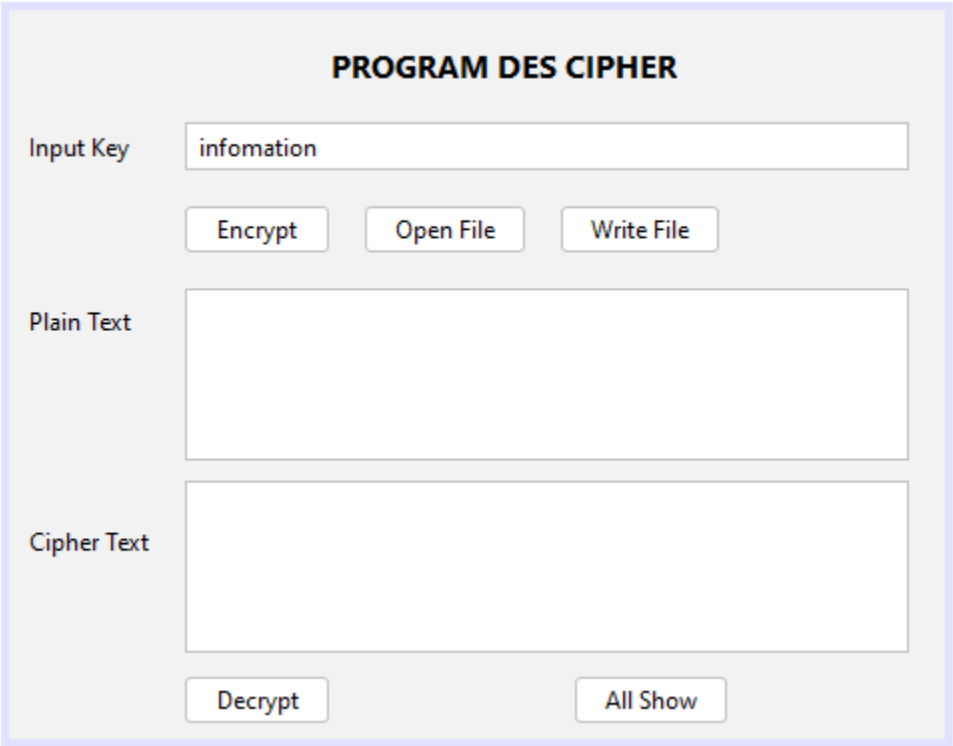

```
package lab3;
public class Transposition_Cipher extends javax.swing.JFrame {
```

```
public Transposition_Cipher() {
    initComponents();
}

private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String k = this.txtKey.getText();
    String ks[] = new String[6];
    ks = k.split(",");
    int key[] = new int[6];
    for(int i = 0; i < 6; i ++){
        key[i] = Integer.valueOf(ks[i]) - 1;
    }
    String sa = this.txtPlainText.getText();
    String kq = "";
    int na = sa.length();
    int d = 0;
    int c;
    String s = "";
    int thieu = 6 - na%6;
    for(int i = 0; i < thieu; i++){
        sa = sa + " ";
    }
    while(d < na){
        c = d + 6;
        s = sa.substring(d,c);
        for(int i = 0; i < 6; i++){
            kq = kq + s.charAt(key[i]);
        }
        d = d + 6;
    }
```

```
    this.txtCipherText.setText(kq);
}

private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String k = this.txtKey.getText();
    String ks[] = new String[6];
    ks = k.split(",");
    int key[] = new int[6];
    for(int i = 0; i < 6; i ++){
        key[i] = Integer.valueOf(ks[i]) - 1;
    }
    int key1[] = new int[6];
    for(int i = 0; i < 6; i ++){
        key1[key[i]] = i;
    }
    String sa = this.txtCipherText.getText();
    String kq = "";
    int na = sa.length();
    int d = 0;
    int c;
    String s = "";
    while(d < na){
        c = d + 6;
        s = sa.substring(d,c);
        for(int i = 0; i < 6; i++){
            kq = kq + s.charAt(key1[i]);
        }
        d = d + 6;
    }
    this.txtPlainText.setText(kq);
```

```
    }
}
```

LAB4_1: DES_Cipher




```
package lab4;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import javax.crypto.Cipher;
```

```
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import javax.swing.JOptionPane;
import java.util.logging.*;
public class DES_Cipher extends javax.swing.JFrame {
    public DES_Cipher() {
        initComponents();
    }

    private int mode;
    private static void doCopy(InputStream is, OutputStream os) throws IOException{
        byte[] bytes = new byte[64];
        int numBytes;
        while((numBytes = is.read(bytes)) != -1){
            os.write(bytes,0,numBytes);
        }
        os.flush();
        os.close();
        is.close();
    }

    public static void encrypt(String key, InputStream is, OutputStream os) throws Throwable{
        encryptOrDecrypt(key, Cipher.ENCRYPT_MODE, is, os);
    }
    public static void decrypt(String key, InputStream is, OutputStream os) throws Throwable{
        encryptOrDecrypt(key, Cipher.DECRYPT_MODE, is, os);
    }
    public static void encryptOrDecrypt(String key,int mode, InputStream is, OutputStream os) throws Throwable{
```

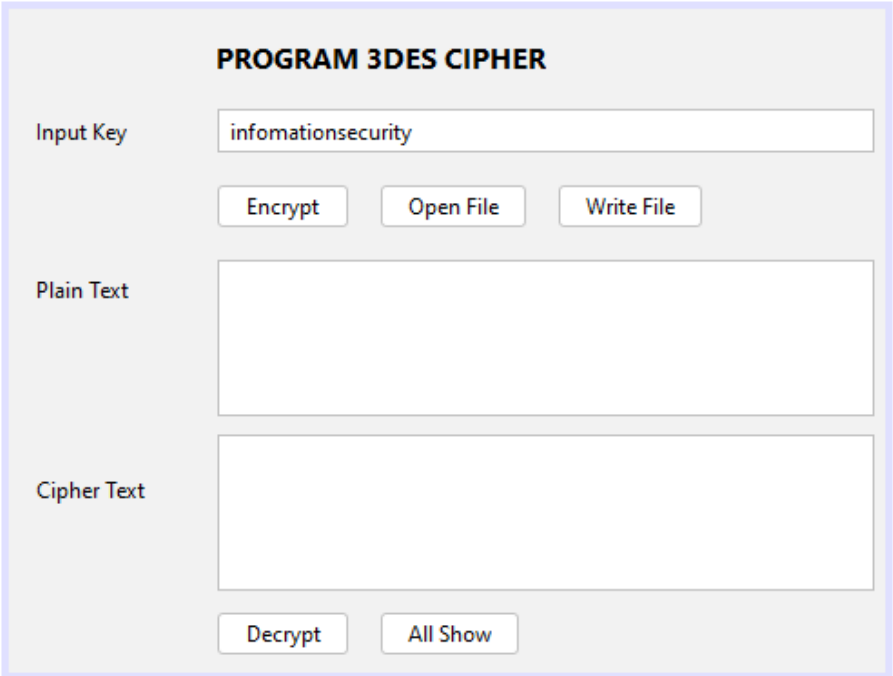
```
    DESKeySpec dks = new DESKeySpec(key.getBytes());
    SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");
    SecretKey desKey = skf.generateSecret(dks);
    Cipher cipher = Cipher.getInstance("DES");

    if(mode == Cipher.ENCRYPT_MODE){
        cipher.init(Cipher.ENCRYPT_MODE, desKey);
        CipherInputStream cis = new CipherInputStream(is,cipher);
        doCopy(cis, os);
    } else if (mode == Cipher.DECRYPT_MODE){
        cipher.init(Cipher.DECRYPT_MODE, desKey);
        CipherOutputStream cos = new CipherOutputStream(os,cipher);
        doCopy(is, cos);
    }
}
private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        String key = this.txtKey.getText();
        FileInputStream fis = new FileInputStream("D:\\Des.txt");
        FileOutputStream fos = new FileOutputStream("D:\\EnDes.txt");
        encrypt(key, fis, fos);
        JOptionPane.showMessageDialog(null, "Encrypted!!!");
    } catch(Throwable e){
        e.printStackTrace();
    }
}

private void btnWriteFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try
```

```
    {
        BufferedWriter bw = null;
        String fileName = "D:\\Des.txt";
        String s = txtPlainText.getText();
        bw = new BufferedWriter(new FileWriter(fileName));
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, "Wrote File Success!!!");
        //txtCipherText.setText(s);
    }
    catch (IOException ex)
    {
        Logger.getLogger(DES_Cipher.class.getName()).log(Level.SEVERE, null,ex);
    }
}

private void btnOpenFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        BufferedReader br = null;
        String fileName = "D:\\EnDes.txt";
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, "Opened File!!!");
        char[] ca = new char[5];
        while(br.ready()){
            int len = br.read(ca);
            sb.append(ca,0,len);
        }
        br.close();
        System.out.println("Data is: " + sb);
```

```
        String chuoi = sb.toString();
        txtCipherText.setText(chuoi);
    } catch(IOException ex){
        Logger.getLogger(DES_Cipher.class.getName()).log(Level.SEVERE,null,ex);
    }
}

private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FileInputStream fis2 = null;
    try{
        String key = this.txtKey.getText();
        fis2 = new FileInputStream("D:\\Des.txt");
        FileOutputStream fos2 = new FileOutputStream("D:\\EnDes.txt");
        decrypt(key, fis2, fos2);
        BufferedReader br = null;

        br = new BufferedReader(new FileReader("D:\\Des.txt"));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, "Decrypted!!!");
        char[] ca = new char[5];
        while(br.ready()){
            int len = br.read(ca);
            sb.append(ca,0,len);
        }
        br.close();
        System.out.println("Data is: " + sb);
        String chuoi = sb.toString();
        txtPlainText.setText(chuoi);
    } catch(Throwable ex){
        Logger.getLogger(DES_Cipher.class.getName()).log(Level.SEVERE,null,ex);
```

```
        }
        finally{
            try{
                fis2.close();
            } catch(IOException ex){
                Logger.getLogger(DES_Cipher.class.getName()).log(Level.SEVERE,null,ex);
            }
        }
    }

    private void btnAllShowActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
}
```

LAB4_2: fm3DES_Cipher



```
package lab4;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.security.spec.KeySpec;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import java.util.Base64;
import java.util.logging.*;
import javax.crypto.spec.DESedeKeySpec;
import javax.swing.JOptionPane;
public class fm3DES_Cipher extends javax.swing.JFrame {
    public fm3DES_Cipher() {
        initComponents();
    }
    private static final String UNICODE_FORMAT = "UTF8";
    public static final String DESEDE_ENCRYPTION_SCHEME = "DESede";
    private KeySpec myKeySpec;
    private SecretKeyFactory mySecretKeyFactory;
    private Cipher cipher;
    byte[] keyAsBytes;
    private String myEncryptionKey;
    private String myEncryptionScheme;
    SecretKey key;

    public String encrypt(String unencryptedString){
        String encryptedString = null;
        try{
```

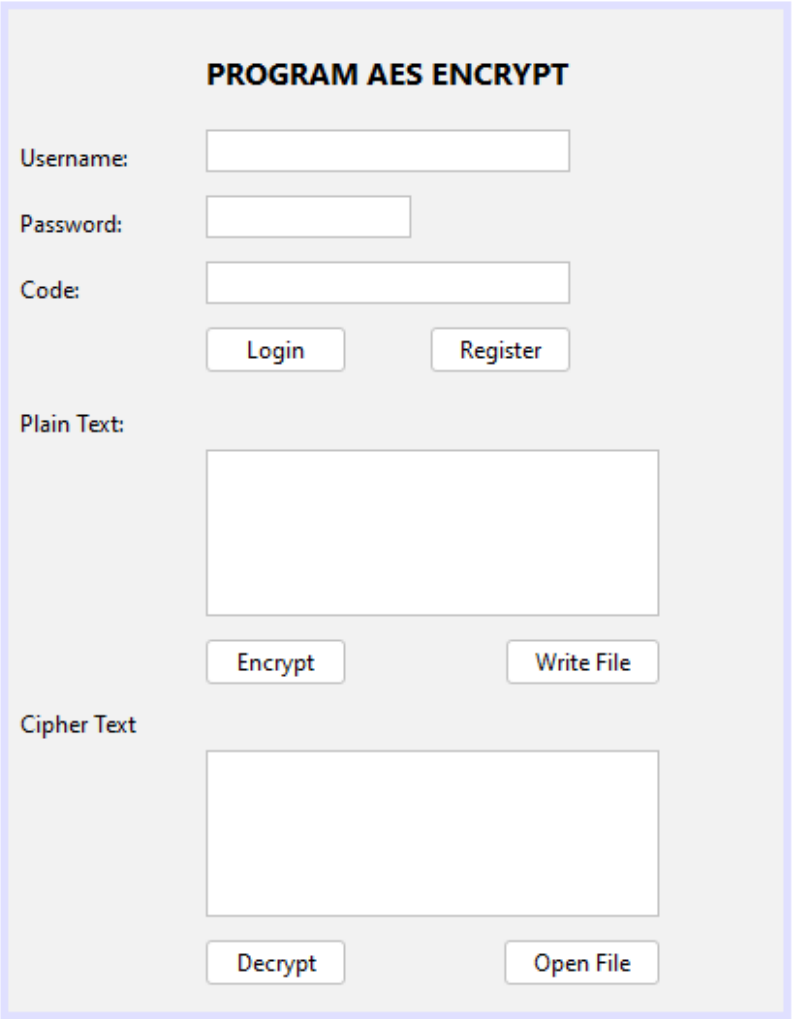
```
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] plainText = unencryptedString.getBytes(UNICODE_FORMAT);
        byte[] encryptedText = cipher.doFinal(plainText);
        encryptedString = Base64.getEncoder().encodeToString(encryptedText);
    }catch (Exception e){
        e.printStackTrace();
    }
    return encryptedString;
}

public String decrypt(String encryptedString){
    String decryptedText = null;
    try{
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] encryptedText = Base64.getDecoder().decode(encryptedString);
        byte[] plainText = cipher.doFinal(encryptedText);
        String a = new String(plainText);
        System.out.println("plainText: " + a);
        decryptedText = a;
    }catch (Exception e){
        e.printStackTrace();
    }
    return decryptedText;
}

private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        myEncryptionKey = this.txtKey.getText();
        myEncryptionScheme = DESEDE_ENCRYPTION_SCHEME;
        keyAsBytes = myEncryptionKey.getBytes(UNICODE_FORMAT);
```

```
        //add them
        if (keyAsBytes.length < 24){
            System.out.println("Input 24 byte of Input Key!");
            return;
        }
        //
        myKeySpec = new DESedeKeySpec(keyAsBytes);
        mySecretKeyFactory = SecretKeyFactory.getInstance(myEncryptionScheme);
        cipher = Cipher.getInstance(myEncryptionScheme);
        key = mySecretKeyFactory.generateSecret(myKeySpec);
        System.out.println("Key k: " + key);
        String plainText = txtPlainText.getText();
        String encrypted = encrypt(plainText);
        System.out.println("Encrypted Value: " + encrypted);
        txtCipherText.setText(encrypted);
    } catch(Exception ex){
        ex.printStackTrace();
    }
}

private void btnOpenFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        BufferedReader br = null;
        String fileName = "D:\\3Des.txt";
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, "Opened File!!!");
        char[] ca = new char[5];
        while(br.ready()){
            int len = br.read(ca);
```

```
            sb.append(ca,0,len);
        }
        br.close();
        System.out.println("Data is: " + sb);
        String chuoi = sb.toString();
        txtPlainText.setText(chuoi);
    } catch(IOException ex){
        Logger.getLogger(fm3DES_Cipher.class.getName()).log(Level.SEVERE,null,ex);
    }
}

private void btnWriteFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try
    {
        BufferedWriter bw = null;
        String fileName = "D:\\3Des.txt";
        String s = txtPlainText.getText();
        bw = new BufferedWriter(new FileWriter(fileName));
        bw.write(s);
        bw.close();
        JOptionPane.showMessageDialog(null, "Wrote File Success!!!");
        //txtCipherText.setText(s);
    }
    catch (IOException ex)
    {
        Logger.getLogger(fm3DES_Cipher.class.getName()).log(Level.SEVERE, null,ex);
    }
}

private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        // TODO add your handling code here:
        try{
            myEncryptionKey = this.txtKey.getText();
            myEncryptionScheme = DESEDE_ENCRYPTION_SCHEME;
            keyAsBytes = myEncryptionKey.getBytes(UNICODE_FORMAT);
            //add them
            if (keyAsBytes.length < 24){
                System.out.println("Input 24 byte of Input Key!");
                return;
            }
            //
            myKeySpec = new DESedeKeySpec(keyAsBytes);
            mySecretKeyFactory = SecretKeyFactory.getInstance(myEncryptionScheme);
            cipher = Cipher.getInstance(myEncryptionScheme);
            key = mySecretKeyFactory.generateSecret(myKeySpec);
            System.out.println("Key k: " + key);
            String cipherText = txtCipherText.getText();
            String decrypted = decrypt(cipherText);
            System.out.println("Decrypted Value: " + decrypted);
            txtPlainText.setText(decrypted);
        } catch(Exception ex){
            ex.printStackTrace();
        }
    }
    private void btnAllShowActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
}
```

LAB5: AES_Encrypt




```
package lab5;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.security.NoSuchAlgorithmException;

import java.util.Base64;

import java.util.logging.Level;

import java.util.logging.Logger;
```

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.swing.JOptionPane;
public class AES_Encrypt extends javax.swing.JFrame {
    public AES_Encrypt() {
        initComponents();
    }

    private String user;
    private String pass;
    private String khoa;
    SecretKey secretKey;
    byte[] byteCipherText;

    private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try{
            user = txtUser.getText();
            pass = txtPass.getText();
            khoa = user + pass;
            BufferedReader br = null;
            String fileName = "D:\\AES.txt";
            br = new BufferedReader(new FileReader(fileName));
            StringBuffer sb = new StringBuffer();
            char[] ca = new char[5];
            while(br.ready()){
                int len = br.read(ca);
                sb.append(ca,0,len);
            }
            br.close();
```

```
        System.out.println("Key is: " + sb);
        String chuoi = sb.toString();
        Boolean k = khoa.equals(chuoi);
        if (k == true){
            JOptionPane.showMessageDialog(null, "Login Successful!!!");
        } else {
            JOptionPane.showMessageDialog(null, "Login Fail!!!");
        }
        txtKey.setText(chuoi.getBytes().toString());
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(128);
        secretKey = keyGen.generateKey();
    } catch (NoSuchAlgorithmException ex){

    } catch (Exception ex){ }
}

private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        user = txtUser.getText();
        pass = txtPass.getText();
        khoa = user + pass;
        BufferedWriter bw = null;
        String fileName = "D:\\AES.txt";
        String s = txtPlainText.getText();
        bw = new BufferedWriter(new FileWriter(fileName));
        bw.write(khoa);
        bw.close();
        JOptionPane.showMessageDialog(null, "Register successfull. Login please!!!");
        txtKey.setText(khoa.getBytes().toString());
```

```
    }catch (IOException ex){
        Logger.getLogger(AES_Encrypt.class.getName()).log(Level.SEVERE, null,ex);
    }
}

private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        System.out.println("Create key: " + secretKey);
        Cipher aesCipher = Cipher.getInstance("AES");
        aesCipher.init(Cipher.ENCRYPT_MODE, secretKey);
        String strData = txtPlainText.getText();
        byte[] byteDataToEncrypt = strData.getBytes();
        byteCipherText = aesCipher.doFinal(byteDataToEncrypt);
        String strCipherText = Base64.getEncoder().encodeToString(byteCipherText);
        System.out.println("Cipher Text generated using AES is: " + strCipherText);
        txtCipherText.setText(strCipherText);
    }catch (Exception ex){
        System.out.println("Encrypt error: " + ex);
    }
}

private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        String cipherText = txtCipherText.getText();
        txtPlainText.setText(cipherText);
        Cipher aesCipher = Cipher.getInstance("AES");
        aesCipher.init(Cipher.DECRYPT_MODE, secretKey, aesCipher.getParameters());
        byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);
        String strDecryptedText = new String(byteDecryptedText);
```

```
        System.out.println("Decrypted Text messaage is: " + strDecryptedText);
        txtCipherText.setText(strDecryptedText);
    }catch (Exception ex){
        System.out.println("Decrypt error: " + ex);
    }
}

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

private void btnOpenFileActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try
    {
        BufferedReader br = null;
        String fileName = "D:\\WriteAES.txt";
```

```
            br = new BufferedReader(new FileReader(fileName));
            StringBuffer sb = new StringBuffer();
            JOptionPane.showMessageDialog(null, "Opened File!!!");
            char[] ca = new char[5];
            while(br.ready())
            {
                int len = br.read(ca);
                sb.append(ca,0,len);
            }
            br.close();
            System.out.println("Data is: " + sb);
            String chuoi = sb.toString();
            this.txtPlainText.setText(chuoi);
            btnDecrypt.enable(true);
        }
        catch (IOException ex)
        {
            Logger.getLogger(AES_Encrypt.class.getName()).log(Level.SEVERE, null,ex);
        }
    }
}
```

LAB6: Thuật toán RSA

```
package lab6;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigInteger;
import java.util.Random;
```

```
public class RSA {
    int primeSize;
    BigInteger p,q;
    BigInteger N;
    BigInteger r;
    BigInteger E,D;

    public RSA(){

    }

    public RSA(int primeSize){
        this.primeSize = primeSize;
        generatePrimeNumbers();
        generatePublicPrivateKeys();
    }

    public void generatePrimeNumbers(){
        p = BigInteger.probablePrime(primeSize / 2, new Random());
        do{
            q = BigInteger.probablePrime(primeSize / 2, new Random());
        }while (q.compareTo(p) == 0);
    }

    public void generatePublicPrivateKeys(){
        N = p.multiply(q);
        r = p.subtract(BigInteger.valueOf(1));
        r = r.multiply(q.subtract(BigInteger.valueOf(1)));
        do{
            E = new BigInteger(2 * primeSize, new Random());
        } while((E.compareTo(r) != -1) || (E.gcd(r).compareTo(BigInteger.valueOf(1)) != 0));
```

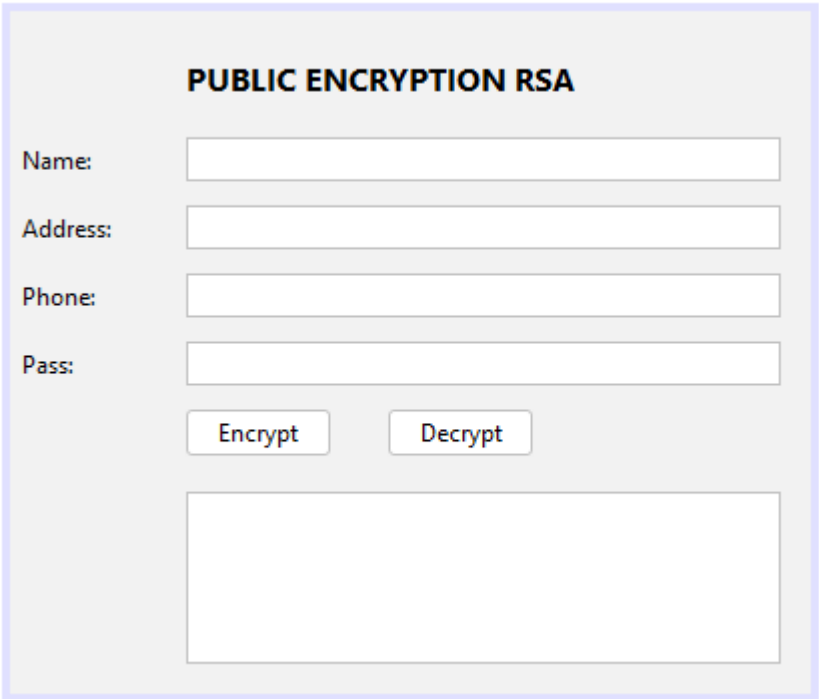
```
    D = E.modInverse(r);
}

public BigInteger[] encrypt(String message){
    int i;
    byte[] temp = new byte[1];
    byte[] digits = message.getBytes();
    BigInteger[] bigdigits = new BigInteger[digits.length];
    for(i = 0; i < bigdigits.length; i++){
        temp[0] = digits[i];
        bigdigits[i] = new BigInteger(temp);
    }
    BigInteger[] encrypted = new BigInteger[bigdigits.length];
    for(i = 0; i < bigdigits.length; i++){
        encrypted[i] = bigdigits[i].modPow(E, N);
    }
    return encrypted;
}

public BigInteger[] encrypt(String message, BigInteger userD, BigInteger userN){
    int i;
    byte[] temp = new byte[1];
    byte[] digits = message.getBytes();
    BigInteger[] bigdigits = new BigInteger[digits.length];
    for(i = 0; i < bigdigits.length; i++){
        temp[0] = digits[i];
        bigdigits[i] = new BigInteger(temp);
    }
    BigInteger[] encrypted = new BigInteger[bigdigits.length];
    for(i = 0; i < bigdigits.length; i++){
        encrypted[i] = bigdigits[i].modPow(userD, userN);
```

```
    }
    return encrypted;
}

public String decrypt(BigInteger[] encrypted, BigInteger D, BigInteger N){
    int i;
    BigInteger[] decrypted = new BigInteger[encrypted.length];
    for(i = 0; i < decrypted.length; i++){
        decrypted[i] = encrypted[i].modPow(D, N);
    }
    char[] charArray = new char[decrypted.length];
    for(i = 0; i < charArray.length; i++){
        charArray[i] = (char)(decrypted[i].intValue());
    }
    return (new String(charArray));
}

public BigInteger getp(){
    return p;
}

public BigInteger getq(){
    return q;
}

public BigInteger getr(){
    return r;
}

public BigInteger getN(){
    return N;
```

```
}

public BigInteger getE(){
    return E;
}

public BigInteger getD(){
    return D;
}

public static void main(String[] args) throws IOException{
    int primeSize = 8;
    RSA rsa = new RSA(primeSize);
    System.out.println("Key size: [" + primeSize + "]");
    System.out.println("");
    System.out.println("Generated prime numbers p and q");
    System.out.println("p: [" + rsa.getp().toString(16).toUpperCase() + "]");
    System.out.println("q: [" + rsa.getq().toString(16).toUpperCase() + "]");
    System.out.println("");
    System.out.println("The public key is the pair (N,E) which will be published.");
    System.out.println("p: [" + rsa.getN().toString(16).toUpperCase() + "]");
    System.out.println("q: [" + rsa.getE().toString(16).toUpperCase() + "]");
    System.out.println("");
    System.out.println("The private key is the pair (N,D) which will be kept private.");
    System.out.println("p: [" + rsa.getN().toString(16).toUpperCase() + "]");
    System.out.println("q: [" + rsa.getD().toString(16).toUpperCase() + "]");
    System.out.println("");
    System.out.println("Please enter message (plaintext):");
    String plainText = (new BufferedReader(new InputStreamReader(System.in))).readLine();
    System.out.println("");
    BigInteger[] cipherText = rsa.encrypt(plainText);
```

```
        System.out.print("Ciphertext: [");
        for(int i = 0; i < cipherText.length; i++){
            System.out.print(cipherText[i].toString(16).toUpperCase());
            if(i != cipherText.length - 1){
                System.out.print(" ");
            }
        }
        System.out.println("]");
        System.out.println("");
        RSA rsa1 = new RSA(8);
        String recoveredPlaintext = rsa1.decrypt(cipherText, rsa.getD(), rsa.getN());
        System.out.print("Recovered plaintext: [" + recoveredPlaintext + "]");
    }
}
```

LAB6: Form RSA



```
package lab6;
import java.math.BigInteger;
import java.util.Scanner;

public class fRSA extends javax.swing.JFrame {
    public fRSA() {
        initComponents();
    }

    private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Scanner in  = new Scanner(System.in);
        String nhash;
        BigInteger[] ciphertext = null;
        BigInteger n = null;
        BigInteger d = null;
        String password = "";

        password = txtPass.getText();

        RSA rsa = new RSA(8);
        n = rsa.getN();
        d = rsa.getD();
        ciphertext = rsa.encrypt(password);
        StringBuffer bf = new StringBuffer();
        for(int i = 0; i < ciphertext.length; i++){
            bf.append(ciphertext[i].toString(16).toUpperCase());
            if(i != ciphertext.length - 1){
                System.out.print("");
            }
        }
```

```
        String message = bf.toString();
        if (txtCipherText.getText().length() > 0){
            txtCipherText.append("\nPass encrypted is: " + message);
        }else
            txtCipherText.append("Pass encrypted is: " + message);
    }

    private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Scanner in  = new Scanner(System.in);
        String nhash;
        BigInteger[] ciphertext = null;
        BigInteger n = null;
        BigInteger d = null;
        String password = "";

        password = txtPass.getText();

        RSA rsa = new RSA(8);
        n = rsa.getN();
        d = rsa.getD();
        ciphertext = rsa.encrypt(password);
        String dhash = rsa.decrypt(ciphertext, d, n);
        txtCipherText.append("\nPass after decrypt is: " + dhash);
    }
}
```

LAB7: Crypto.java

```
package lab7;

public class Crypto {

   public static final String toHexString(byte[] block)

   {

      StringBuffer buf = new StringBuffer();

      int len = block.length;

      for (int i = 0; i < len; i++)

      {

         byte2hex(block[i], buf);

         if (i < len-1)

         {

            buf.append(":");

         }

      }

      return buf.toString();

   }


   public static final void byte2hex(byte b, StringBuffer buf)

   {

      char[] hexChars = { '0', '1', '2', '3',

                          '4', '5', '6', '7',

                          '8', '9', 'A', 'B',

                          'C', 'D', 'E', 'F' };

      int high = ((b & 0xf0) >> 4);

      int low = (b & 0x0f);

      buf.append(hexChars[high]);

      buf.append(hexChars [low]);

   }

}
```

LAB7_2: Alice

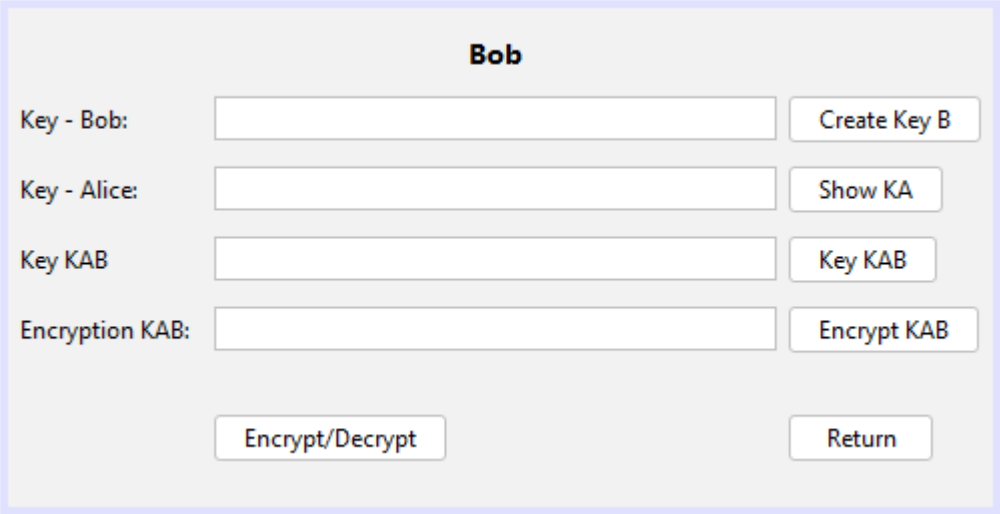

```
package lab7;

import java.io.BufferedWriter;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.security.AlgorithmParameterGenerator;

import java.security.AlgorithmParameters;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PublicKey;

import java.security.spec.X509EncodedKeySpec;

import javax.crypto.Cipher;

import javax.crypto.KeyAgreement;

import javax.crypto.SecretKey;

import javax.crypto.spec.DHParameterSpec;

public class Alice extends javax.swing.JFrame {

    KeyAgreement aliceKeyAgree;

    PublicKey bobPubKey;
```

```
    SecretKey aliceDesKey;
    Cipher aliceCipher;
    public Alice() {
        initComponents();
    }

    private void btnCreateKeyAActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try {
            AlgorithmParameterGenerator paramGen=AlgorithmParameterGenerator.getInstance("DH");
            paramGen.init(512);
            AlgorithmParameters params = paramGen.generateParameters();
            DHParameterSpec dhSkipParamSpec=(DHParameterSpec) params.getParameterSpec
(DHParameterSpec.class);
            System.out.println("Generating a DH Keypair...");
            KeyPairGenerator aliceKpairGen = KeyPairGenerator.getInstance("DH");
            aliceKpairGen.initialize (dhSkipParamSpec);
            KeyPair aliceKpair = aliceKpairGen.generateKeyPair();

            System.out.println("Initializing the KeyAgreement Engine with DH private key");
            aliceKeyAgree= KeyAgreement.getInstance("DH");
            aliceKeyAgree.init(aliceKpair.getPrivate());

            byte[] alicePubKeyEnc= aliceKpair.getPublic().getEncoded();
            FileOutputStream fos =new FileOutputStream("D:/A.pub");
            fos.write(alicePubKeyEnc);
            fos.close();
            txtKeyA.setText(alicePubKeyEnc.toString());
        } catch (Exception e) {
        }
    }
```

```
private void btnShowKhoaBActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        FileInputStream fis = new FileInputStream("D:/B.pub");
        byte[] bkeyP=new byte[fis.available()];
        fis.close();
        txtKeyB.setText (bkeyP.toString());
    } catch (Exception e) {
    }
}

private void btnEncryptKABActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        aliceKeyAgree.doPhase (bobPubKey, true);
        aliceDesKey = aliceKeyAgree.generateSecret("DES");
        txtEncryptionKAB.setText(aliceDesKey.toString());
        BufferedWriter bw = null;
        String fileName="D:\\KeyA.txt";
        bw = new BufferedWriter(new FileWriter(fileName));
        bw.write(aliceDesKey.toString());
        bw.close();
    } catch (Exception e) {
    }
}

private void btnKeyABActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        FileInputStream fis = new FileInputStream("D:/B.pub");
```

```
            byte[] bobPubKeyEnc = new byte[fis.available()];
            fis.read(bobPubKeyEnc);
            fis.close();

            KeyFactory aliceKeyFac=KeyFactory.getInstance("DH");
            X509EncodedKeySpec x509KeySpec=new X509EncodedKeySpec (bobPubKeyEnc);
            bobPubKey= aliceKeyFac.generatePublic(x509KeySpec);
            System.out.println("Executing PHASEl of key agreement...");
            aliceKeyAgree.doPhase (bobPubKey, true);
            byte[] aliceSharedSecret=aliceKeyAgree.generateSecret();
            System.out.println("Key KAB: secret (DEBUG ONLY): " + Crypto.toHexString(aliceSharedSecret));
            txtKeyAB.setText (Crypto.toHexString(aliceSharedSecret));
        } catch (Exception ex) {}
    }

    private void btnEncryptDecryptActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Bob des = new Bob();
        des.setVisible(true);
    }

    private void btnReturnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Alice n = new Alice();
        n.setVisible(true);
    }
}
```

LAB7_3: Bob



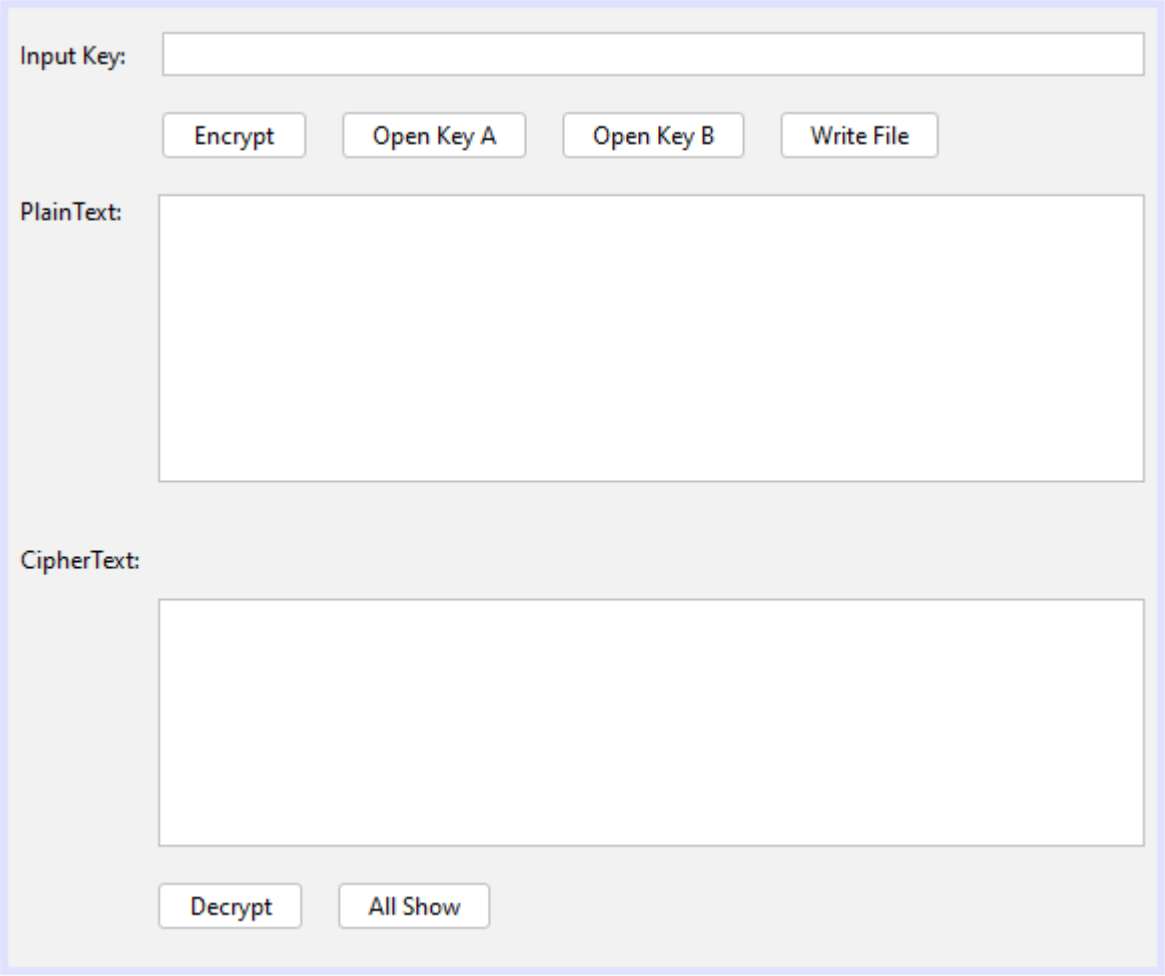

```
package lab7;

import java.io.BufferedWriter;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PublicKey;

import java.security.spec.X509EncodedKeySpec;

import javax.crypto.Cipher;

import javax.crypto.KeyAgreement;

import javax.crypto.SecretKey;

import javax.crypto.interfaces.DHPublicKey;

import javax.crypto.spec.DHParameterSpec;

public class Bob extends javax.swing.JFrame {

   KeyAgreement bobKeyAgree;

   PublicKey alicePubKey;

   SecretKey bobDesKey;
```

```
Cipher bobCipher;
public Bob() {
    initComponents();
}

private void btnCreateKeyBActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        boolean read = false;
        while(!read) {
            try {
                FileInputStream fis = new FileInputStream("D:/A.pub");
                fis.close();
                read=true;
            } catch (Exception ex) {}
        }
        FileInputStream fis = new FileInputStream("D:/A.pub");
        byte[] alicePubKeyEnc = new byte[fis.available()];
        fis.read(alicePubKeyEnc);
        fis.close();
        KeyFactory bobKeyFac = KeyFactory.getInstance("DH");
        X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(alicePubKeyEnc);
        alicePubKey = bobKeyFac.generatePublic (x509KeySpec);
        DHParameterSpec dhParamSpec = ((DHPublicKey) alicePubKey).getParams();
        System.out.println("Generate DH keypair...");
        KeyPairGenerator bobKpairGen = KeyPairGenerator.getInstance("DH");
        bobKpairGen.initialize (dhParamSpec);
        KeyPair bobKpair = bobKpairGen.generateKeyPair();
        System.out.println("initializing KeyAgreement engine...");
        bobKeyAgree = KeyAgreement.getInstance("DH");
        bobKeyAgree.init(bobKpair.getPrivate());
```

```
        byte[] bobPubKeyEnc = bobKpair.getPublic().getEncoded();
        FileOutputStream fos=new FileOutputStream("D:/B.pub");
        fos.write(bobPubKeyEnc);
        fos.close();
        txtkhoab.setText (bobPubKeyEnc.toString());
    } catch (Exception e) {
    }
}

private void btnShowKAActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        FileInputStream fis = new FileInputStream("D:/A.pub");
        byte[] akeyP=new byte[fis.available()];
        fis.read(akeyP);
        fis.close();
        txtkhoaa.setText (akeyP.toString());
    } catch (Exception e) {
    }
}

private void btnKeyABActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        bobKeyAgree.doPhase (alicePubKey, true);
        byte[] bobSharedSecret = bobKeyAgree.generateSecret ();
        System.out.println("Key KAB shared secret (DEBUG ONLY)"+ Crypto.toHexString(bobSharedSecret));
        txtkhoachung.setText(Crypto.toHexString(bobSharedSecret));
    } catch (Exception e) {}
}
```

```
    private void btnEncryptKABActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        try {
            bobKeyAgree.doPhase (alicePubKey, true);
            bobDesKey = bobKeyAgree.generateSecret("DES");
            txtmahoakab.setText(bobDesKey.toString());
            BufferedWriter bw = null;
            String fileName="D:\\KhoaB.txt";
            bw = new BufferedWriter(new FileWriter(fileName));
            bw.write(bobDesKey.toString());
            bw.close();
        } catch (Exception e) {}
    }

    private void btnEncryptDecryptActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Alice des = new Alice();
        des.setVisible(true);
    }

    private void btnReturnActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Bob n = new Bob();
        n.setVisible(true);
    }
}
```

LAB8: DESCS




```
package lab8;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.util.logging.*;
```

```
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import javax.swing.JOptionPane;
public class DESCS extends javax.swing.JFrame {
    public DESCS() {
        initComponents();
    }

    private static void doCopy (InputStream is, OutputStream os) throws IOException{
        byte[] bytes = new byte[64];
        int numBytes;
        while ((numBytes = is.read(bytes))!= -1) {
            os.write(bytes, 0, numBytes);
        }
        os.flush();
        os.close();
        is.close();
    }
    public static void encrypt(String key, InputStream is, OutputStream os) throws Throwable {
        encryptOrDeCrypt(key, Cipher.ENCRYPT_MODE, is, os);
    }

    public static void decrypt(String key, InputStream is, OutputStream os) throws Throwable {
        encryptOrDeCrypt(key, Cipher.DECRYPT_MODE, is, os);
    }

    public static void encryptOrDeCrypt(String key, int mode, InputStream is, OutputStream os) throws Throwable {
```

```
    DESKeySpec dks = new DESKeySpec(key.getBytes());
    SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");
    SecretKey desKey = skf.generateSecret (dks);
    Cipher cipher = Cipher.getInstance("DES");
    if (mode == Cipher. ENCRYPT_MODE) {
        cipher.init(Cipher.ENCRYPT_MODE, desKey);
        CipherInputStream cis = new CipherInputStream(is, cipher);
        doCopy(cis, os);
    }else if (mode == Cipher. DECRYPT_MODE) {
        cipher.init(Cipher.DECRYPT_MODE, desKey);
        CipherOutputStream cos = new CipherOutputStream(os, cipher);
        doCopy(is, cos);
    }
}
private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String key = txtkhoa.getText();
        FileInputStream fis = new FileInputStream("D:\\Des.txt");
        FileOutputStream fos = new FileOutputStream("D:\\EnDes.txt");
        encrypt (key, fis, fos);
        JOptionPane.showMessageDialog(null, "Encrypted");
    } catch (Throwable e) {
        e.printStackTrace();
    }
}

private void btnOpenKeyAActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        BufferedReader br = null;
```

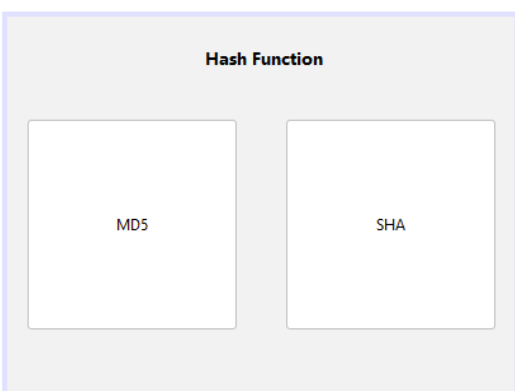
```
        String fileName="D:\\KeyA.txt";
        br = new BufferedReader(new FileReader (fileName));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, "Opened File");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        System.out.println("Data is: " + " " + sb);
        String chuoi = sb.toString();
        txtkhoa.setText(chuoi);
    } catch (IOException ex) {
        Logger.getLogger(DESCS.class.getName()).log (Level. SEVERE, null, ex);
    }
}

private void btnOpenKeyBActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        BufferedReader br = null;
        String fileName = "D:\\KeyB.txt";
        br = new BufferedReader(new FileReader (fileName));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, "Opended File");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
```

```
        br.close();
        System.out.println("Data is: " + " " + sb);
        String chuoi = sb.toString();
        txtkhoa.setText(chuoi);
    } catch (IOException ex) {
        Logger.getLogger(DESCS.class.getName()). log (Level. SEVERE, null, ex);
    }
}

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

private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    FileInputStream fis2 = null;
    try {
        String key = txtkhoa.getText();
        fis2 = new FileInputStream("D:\\EnDes.txt");
```

```
        FileOutputStream fos2 = new FileOutputStream("D:\\DeDes.txt");
        decrypt (key, fis2, fos2);
        BufferedReader br = null;
        String fileName = "D:\\DeDes.txt";
        br = new BufferedReader(new FileReader(fileName));
        StringBuffer sb =new StringBuffer();
        JOptionPane.showMessageDialog(null, "Decrypted!");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        System.out.println("Data is: " + " " + sb);
        String chuoi = sb.toString();
        txtmahoa.setText(chuoi);
    } catch (Throwable ex) {
    }
}

private void btnAllShowActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        BufferedReader br = null;
        String fileName = "D:\\DeDes.txt";
        br = new BufferedReader(new FileReader (fileName));
        StringBuffer sb = new StringBuffer();
        JOptionPane.showMessageDialog(null, "Opened File");
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
```

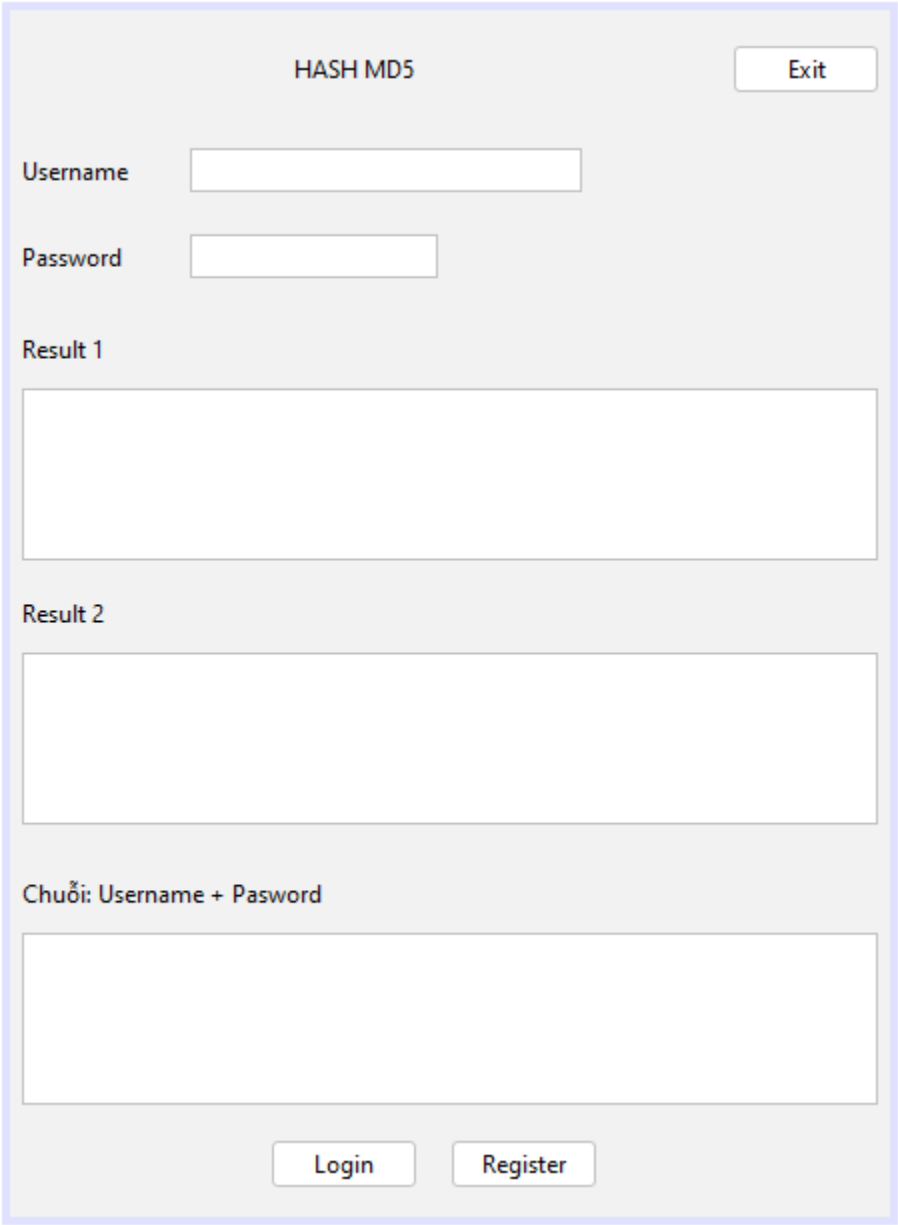
```
            sb.append(ca, 0, len);
        }
        br.close();
        String ff = "D:\\EnDes.txt";
        br = new BufferedReader(new FileReader(ff));
        StringBuffer sb1 = new StringBuffer();
        char[] ca1 = new char[5];
        while (br.ready()) {
            int len = br.read(ca1);
            sb1.append(ca1, 0, len);
        }
        System.out.println("Data is: "+" "+sb);
        String chuoi = sb.toString();
        String chuoi1= sb1.toString();
        txtvanban.setText(chuoi);
        txtmahoa.setText(chuoi1);
    } catch (IOException ex) {}
  }
}
```

LAB9: Main Form



```
package lab9;
public class Main extends javax.swing.JFrame {
    public Main() {
        initComponents();
    }

    private void btnMD5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        MD5 md = new MD5();
        md.show();
        this.dispose();
    }

    private void btnSHAActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        SHA sha = new SHA();
        sha.show();
        this.dispose();
    }
}
```

LAB9_2: MD5



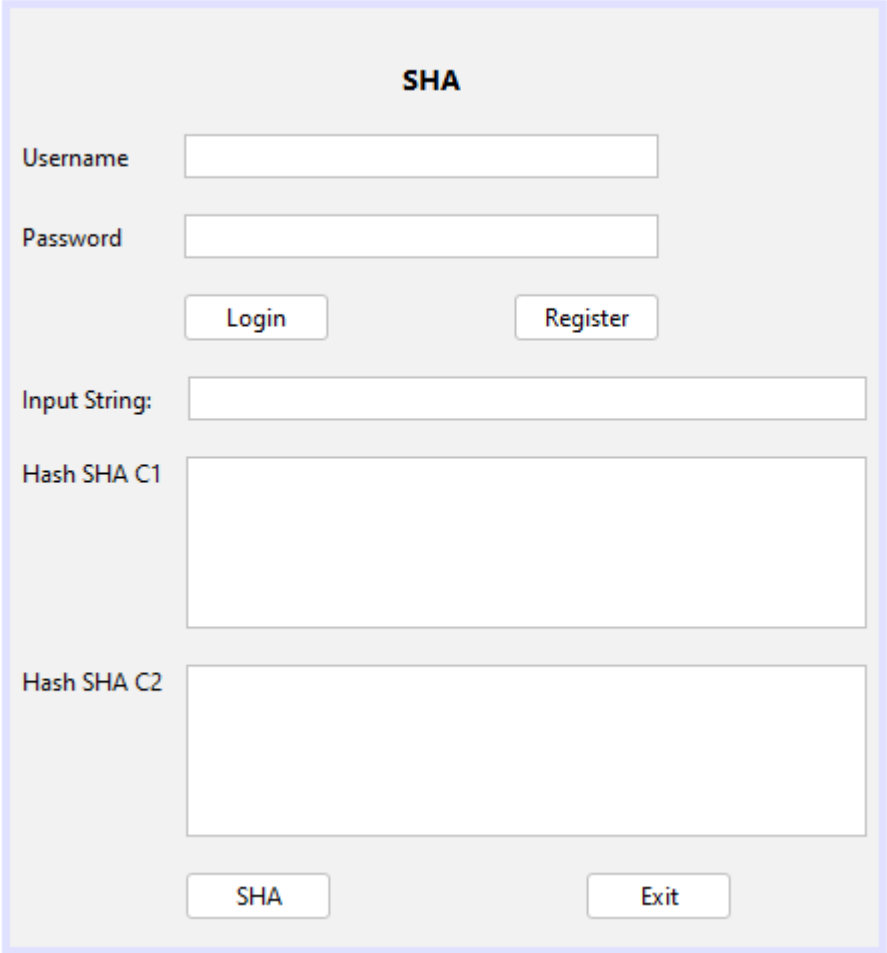

```
package lab9;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.security.MessageDigest;

import javax.swing.JOptionPane;
```

```
public class MD5 extends javax.swing.JFrame {
    public MD5() {
        initComponents();
        txtUser.setText("LeNgocHao");
        txtPass.setText("Abc12345");
    }

    private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String user = txtUser.getText();
        String pass = txtPass.getText();
        String bam = "";
        bam = user + pass;
        BufferedReader br = null;
        String filename = "D:\\HashMD5.txt";
        try {
            br = new BufferedReader(new FileReader (filename));
            StringBuffer sb = new StringBuffer();
            char[] ca = new char[5];
            while (br.ready()) {
                int len = br.read(ca);
                sb.append(ca, 0, len);
            }
            br.close();
            System.out.println("Authentication: " + sb);
            String chuoi = sb.toString();
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update (bam.getBytes());
            byte[] byteData = md.digest();
            StringBuffer hexString = new StringBuffer();
```

```
        for (int i = 0; i < byteData.length; i++) {
            String hex = Integer.toHexString(0xff & byteData[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append (hex);
        }
    } catch (Exception e) {
    }
}

private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String user = txtUser.getText();
        String pass = txtPass.getText();
        String bam = "";
        bam = user + pass;

        MessageDigest md =  MessageDigest.getInstance("MD5");
        md.update (bam.getBytes());
        byte[] byteData = md.digest();

        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < byteData.length; i++) {
            sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
        }
        System.out.println("Digest (in hex format): " + sb.toString());
        txtHash1.setText(sb.toString());

        StringBuffer hexString = new StringBuffer();
```

```
            for (int i = 0; i < byteData.length; i++) {
                String hex = Integer.toHexString(0xff & byteData[i]);
                if (hex.length() == 1) {
                    hexString.append('0');
                }
                hexString.append(hex);
            }
            System.out.println("Digest(in hex format): " + hexString.toString());
            txtHash2.setText(hexString.toString());
            txtString.setText(bam.toString());

            BufferedWriter bw = null;
            String filename = "D:\\HashMD5.txt";
            bw = new BufferedWriter(new FileWriter(filename));
            bw.write(hexString.toString());
            bw.close();
            JOptionPane.showMessageDialog(null, "Registed Success. Please Login");
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, "Error Hash User and Pass: " + ex);
        }
    }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Main main = new Main();
        main.setLocationRelativeTo(null);
        main.show();
        this.dispose();
    }
}
```

LAB9_3: SHA




```
package lab9;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import javax.swing.JOptionPane;

import java.util.logging.*;


public class SHA extends javax.swing.JFrame {

    boolean role;
```

```
public SHA() {
    initComponents();
    role = false;
    txtUser.setText("LeNgocHao");
    txtPass.setText("Abc1234");
    txtString.setText("informationSecurity");
}

private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String user = txtUser.getText();
    String pass = txtPass.getText();
    String bam = "";
    bam = user + pass;

    BufferedReader br = null;
    String filename = "D:\\SHA.txt";
    try {
        br = new BufferedReader(new FileReader (filename));
        StringBuffer sb = new StringBuffer();
        char[] ca = new char[5];
        while (br.ready()) {
            int len = br.read(ca);
            sb.append(ca, 0, len);
        }
        br.close();
        System.out.println("Authentication: " + sb);
        String chuoi = sb.toString();
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update (bam.getBytes());
        byte[] byteData = md.digest();
```

```
        StringBuffer hexString = new StringBuffer();

        for (int i = 0; i < byteData.length; i++) {
            String hex = Integer.toHexString(0xff & byteData[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        System.out.println("Hash username and password: " + hexString.toString());
        Boolean k = hexString.toString().equals(chuoi);
        if (k == true) {
            role = true;
            JOptionPane.showMessageDialog(null, "Login Success!");
            txtbam1.setText (hexString.toString());
            txtbam2.setText(chuoi);
            System.out.println("Username: " + user + "\n" + "Password: "+ pass);
        } else {
            JOptionPane.showMessageDialog(null, "Fail Login!");
            role = false;
        }
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Login Error!");
    }
}

private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        String user = txtUser.getText();
        String pass = txtPass.getText();
```

```
    String bam = "";
    bam = user + pass;

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    md.update (bam.getBytes());
    byte[] byteData = md.digest();

    StringBuffer sb = new StringBuffer();
    for (int i = 0; i < byteData.length; i++) {
        sb.append(Integer.toString((byteData[1] & 0xff) + 0x100, 16).substring(1));
    }
    System.out.println("Digest (in hex format): " + sb.toString());
    txtbam1.setText(sb.toString());

    StringBuffer hexString = new StringBuffer();
    for (int i = 0; i < byteData.length; i++) {
        String hex = Integer.toHexString(0xff & byteData[i]);
        if (hex.length() == 1) {
            hexString.append('0');
        }
        hexString.append (hex);
    }
    BufferedWriter bw = null;
    String filename = "D:\\SHA.txt";
    bw = new BufferedWriter(new FileWriter(filename));
    bw.write(hexString.toString());
    bw.close();
    JOptionPane.showMessageDialog(null, "Registed Success. Login Please!");
} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, "Hash error:user and pass: " + ex);
}
```

```
}

private void btnSHAActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (role == true) {
        try {
            String chuoi = "";
            chuoi = txtString.getText();
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update (chuoi.getBytes());
            byte[] byteData = md.digest();
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < byteData.length; i++) {
                sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
            }
            System.out.println("Hex formatl: " + sb.toString());
            txtbam1.setText(sb.toString());
            StringBuffer hexString = new StringBuffer();
            for (int i = 0; i < byteData.length; i++) {
                String hex = Integer.toHexString(0xff & byteData[i]);
                if (hex.length() == 1) {
                    hexString.append('0');
                }
                hexString.append(hex);
            }
            System.out.println("Hex format2: " + hexString.toString());
            txtbam2.setText (hexString.toString());
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(SHA.class.getName()).log(Level. SEVERE, null, ex);
        }
    }else{
```

```
            JOptionPane.showMessageDialog(null, "Login Please!");
        }
    }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Main main = new Main();
        main.setLocationRelativeTo(null);
        main.show();
        this.dispose();
    }
}
```