INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Cryptography

“Shift - Cipher”

Abstract

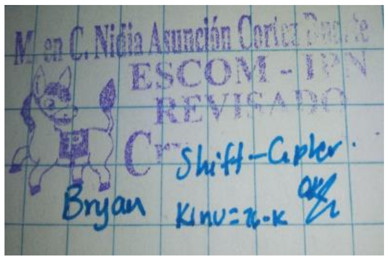
One of the most popular classic ciphers is the Caesar Cipher. In this paper I explain the procedure to apply the shift-cipher algorithm, which is an improvement of the Caesar. The user interface and algorithm were made using Java programing language.

By:

Bryan Dominguez de la Rosa

Professor:
MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

August 2018


M. en C. Nidia Asunción Cortez Duarte
ESCOM - IPN
REVISADO
Cr Shift-Cipher.
Bryan
Kinu=n-k
[illegible]

Index

Contenido

Introduction:

The Caesar Cipher consist in replacing each letter of the alphabet for its corresponding three "jumps" after, for examples, letter A corresponds to letter D, letter B corresponds to letter E and so on. The aim of the Shift-Cipher algorithm is to let the user to decide the number of jumps, i.e. the key, that the alphabet is going to have. If the key given is 5, then the letter A corresponds to the letter F and so on. The main idea of the encryption and decryption algorithm is to use the ASCII code of each letter of the English alphabet (26 symbols) and modular arithmetic to find the correspondence. The user is going to select a plane text file and type the key, then is going to give the name for the encrypted output file. The decryption process is the same.

Literature review:

The cryptology is the science that deal with theoretical problems related with security in encrypted messages exchange from a sender to a receiver through a channel of communication (in informatic terms, this channel is usually a computer network). [1]

The use of cryptographic techniques is almost as old as the cultures of the first villages of our planet. The major advances were achieved in the First and Second World War, especially during and after the last one. The countries in conflict had real companies with a great number of mathematicians whose function was to break the encrypted messages of the tickers exchanged by their enemies.

The Fig. 1 shows the classification of classic cryptosystems [2]:

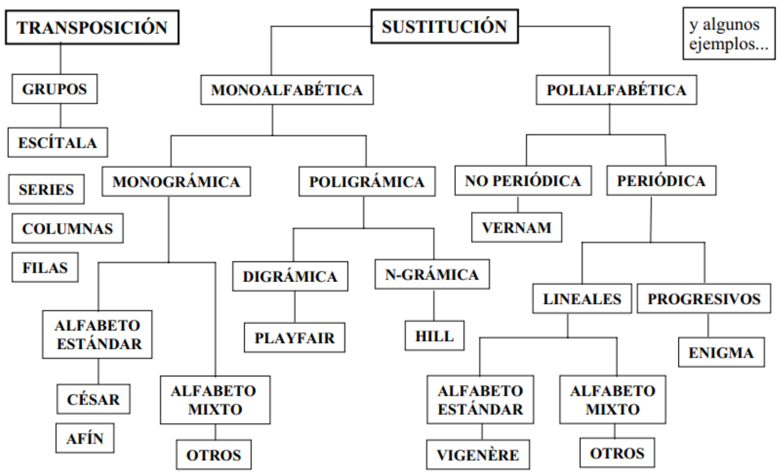


Fig. 1 Classification of classic cryptosystems

As can be seen, the main division is the transposition and substitution algorithms. The Shift-Cipher is in the category of the monoalphabetic substitution, because every single letter of the message that we are encrypting will be mapped to another letter of the alphabet that we are using, usually a 26 letter alphabet.

In the Shift Cipher, the encryption is performed by replacing each letter by the letter a certain number of places on in the alphabet. One technique of breaking this algorithm is using frequency analysis. As can be seen in Fig.2, one graph looks almost like a shift of the other graph. The plain text is given in red and the cipher text is given in blue. [3]

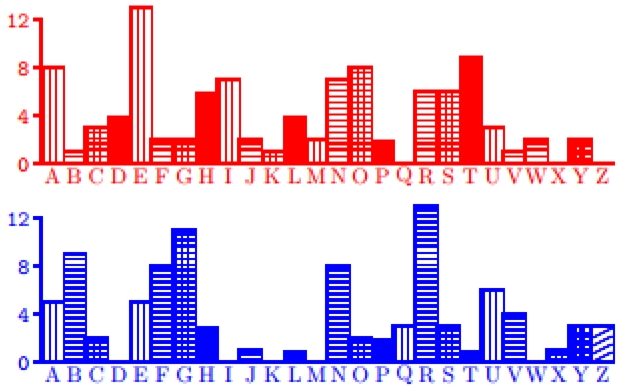


Fig. 2 Comparison of plaintext and ciphertext frequencies for a shift cipher example

Software (libraries, packages, tools):

I use Java programing language to develop this practice, so the tools that I used were:

- NetBeans IDE 8.2
- Java Development Kit (jdk) 8
- All libraries used were of the Java Standard

Procedure:

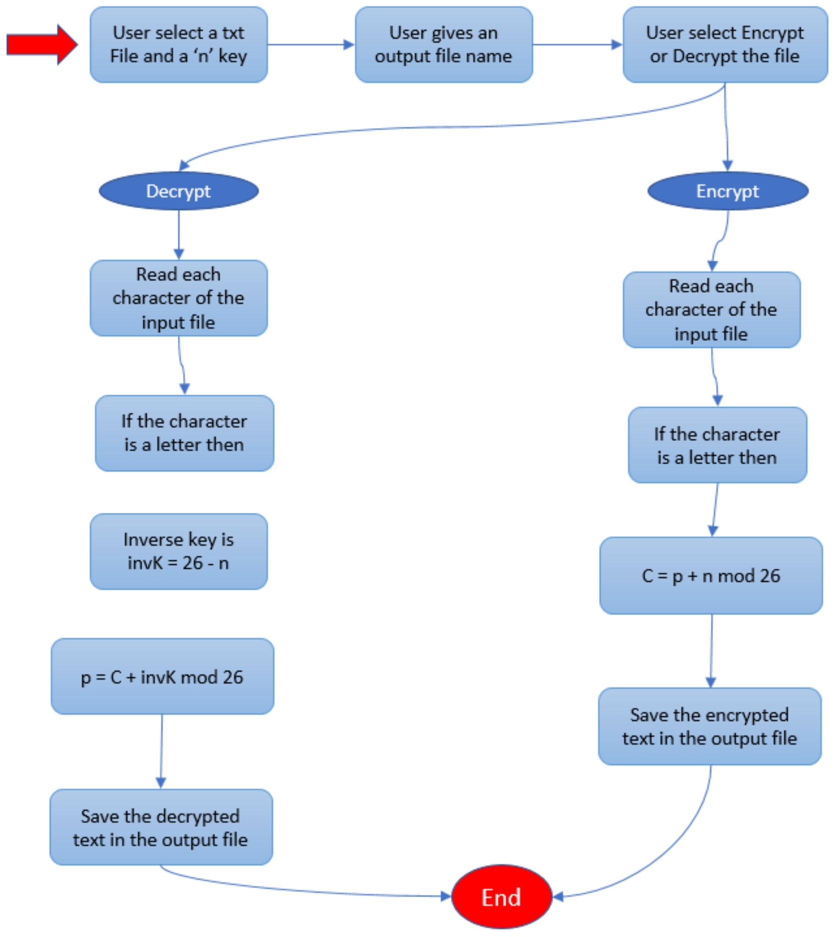

User select a txt File and a 'n' key
User gives an output file name
User select Encrypt or Decrypt the file
Decrypt
Encrypt
Read each character of the input file
Read each character of the input file
If the character is a letter then
If the character is a letter then
Inverse key is invK = 26 - n
C = p + n mod 26
p = C + invK mod 26
Save the encrypted text in the output file
Save the decrypted text in the output file
End

Fig. 3 Block diagram of the procedure of the Shift-Cipher

For the encryption process:

The user must select the input file, which is the one that is going to be encrypted. This file must be in lower case. Then the user has to type the key of the algorithm, i.e. the number of jumps. The user also has to type the name of the output file.

The program is going to read each character of the file, and if the character is a lower-case letter, then it is going to apply the formula:

$$C = p + n \, mod \, 26, \text{ where}$$

C = encrypted character

p = original character

n = key or number of jumps

When the program has finished to encrypt the original message, it is going to save the result in the output txt file.

For the decryption process:

The user must select the input file, which is the one that is going to be encrypted. This file must be in upper case. Then the user has to type the key of the algorithm, i.e. the number of jumps. The user also has to type the name of the output file.

The program is going to read each character of the file, and if the character is an upper-case letter, then it is going to calculate the inverse key, which is necessary to apply the decryption formula. The inverse key is calculated as in $invK = 26 - n$, where n is the original key.

When we have the inverse key then we can apply the next formula:

$p = C + invK \ mod\ 26$, where

C = encrypted character

p = original character

invK = inverse Key

When the program has finished to encrypt the original message, it is going to save the result in the output txt file.

Results:

The user interface is shown in Fig. 4. It is the main interface where the user can select the file to encrypt by clicking the Browse button.

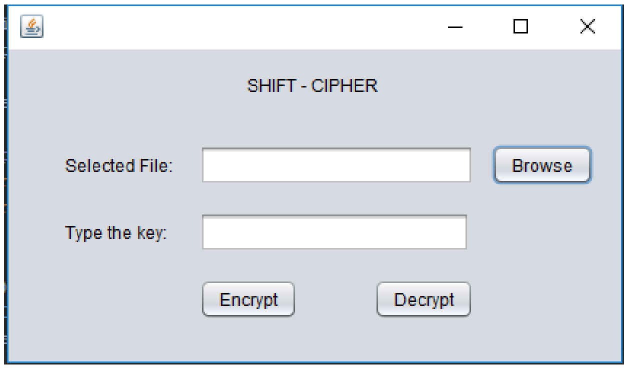


Fig. 4 The main interface of the program.

To make easier the use of the program, I implemented the JFileChooser component of Java (see Fig. 5), it allows the user to navigate through the files in the system and to select the one than he wants.

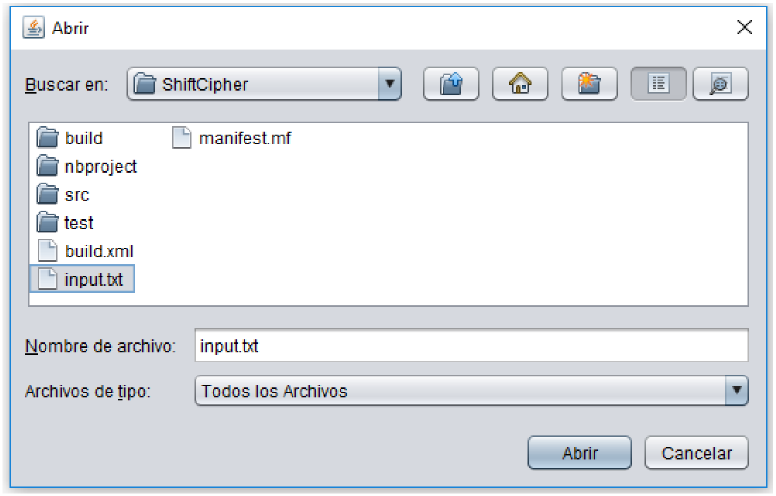


Fig. 5 The JFileChooser implemented in the program

Once the user selects the input file, he has to type the key, or the number of jumps that a letter is going to do. As can be seen in Fig. 6, it's ready to start the encryption or decryption. In this program the input files are going to be successfully encrypted if they are in lower-case. And if the user wants to decrypt a file, that file has to be in upper-case.

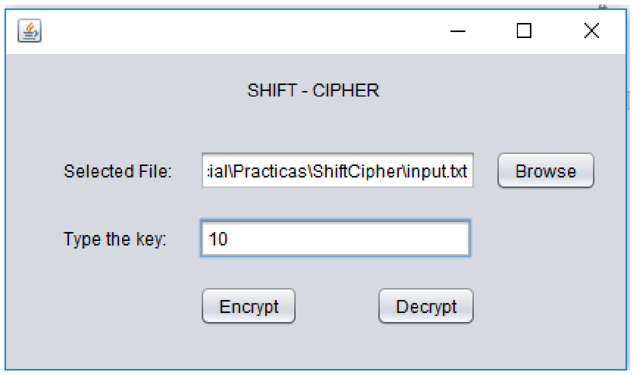


Fig. 6 How the program looks when it's ready

When the user decide to encrypt the file, he has to select the path and the name of the file that is going to be the output. That JFileChooser is shown in Fig. 7.

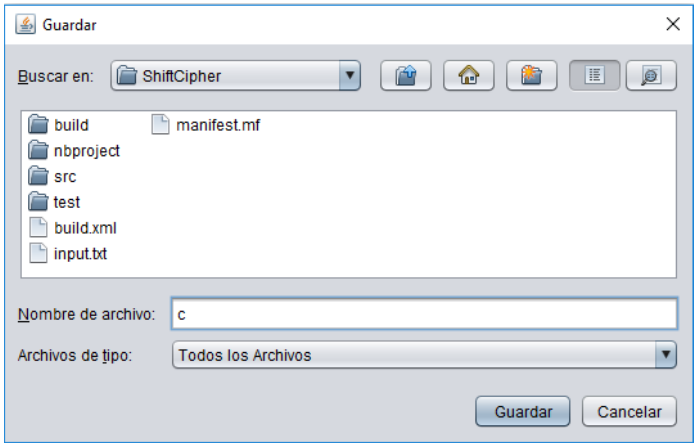
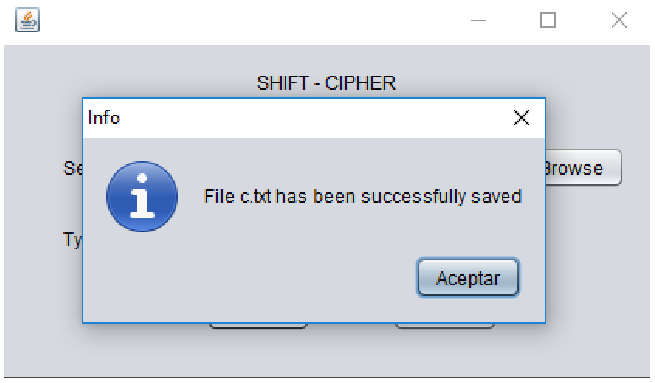


Fig. 7 The JFileChooser that let the user save the output file

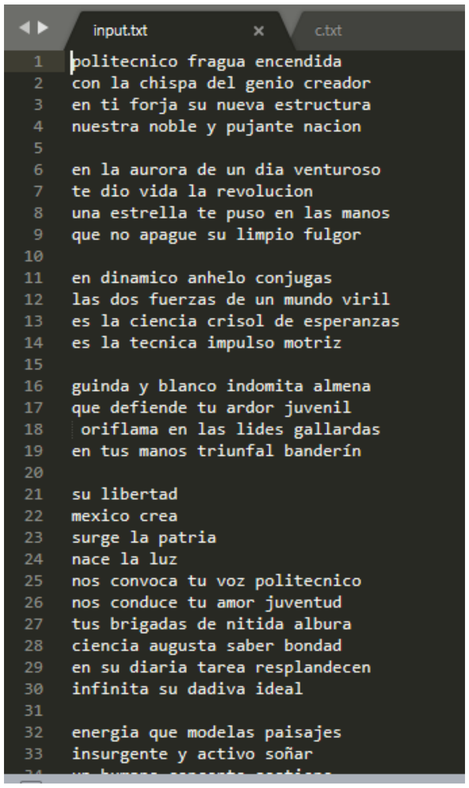
When the program finish, it sends an alert to the user saying that everything has been successfully encrypted and save, as can be seen in Fig. 8.



File c.txt has been successfully saved

Fig. 8 Success message

The Fig. 9 shows the original text file, the message inside is the IPN anthem. We can see that every single character is a lower-case one, because the program ignores any character that is not a lower-case letter, and if we put an upper-case letter or a special symbol, we are going to have loss of information.


```
politecnico fragua encendida
con la chispa del genio creador
en ti forja su nueva estructura
nuestra noble y pujante nacion

en la aurora de un dia venturoso
te dio vida la revolucion
una estrella te puso en las manos
que no apague su limpio fulgor

en dinamico anhelo conjugas
las dos fuerzas de un mundo viril
es la ciencia crisol de esperanzas
es la tecnica impulso motriz

guinda y blanco indomita almena
que defiende tu ardor juvenil
 oriflama en las lides gallardas
en tus manos triunfal banderín

su libertad
mexico crea
surge la patria
nace la luz
nos convoca tu voz politecnico
nos conduce tu amor juventud
tus brigadas de nitida albura
ciencia augusta saber bondad
en su diaria tarea resplandecen
infinita su dadiva ideal

energia que modelas paisajes
insurgente y activo soñar
```

Fig. 9 IPN anthem ready to be encrypted

In the Fig. 10 we can see how the shift-cipher with a 10 key looks. The program omits all the spaces and line breaks and give a unique string. This make the cipher safer, because we cannot identify if we had 2, 3 or 4 letters words.

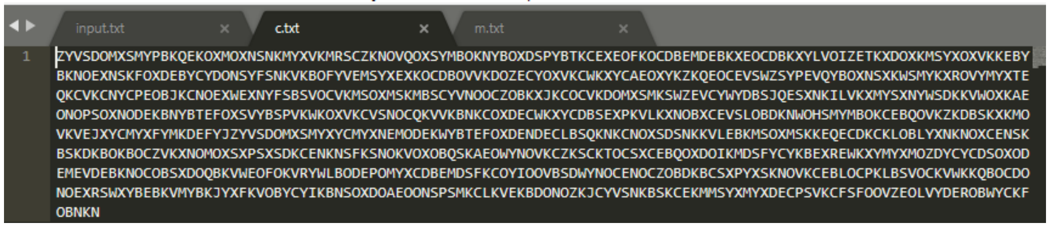
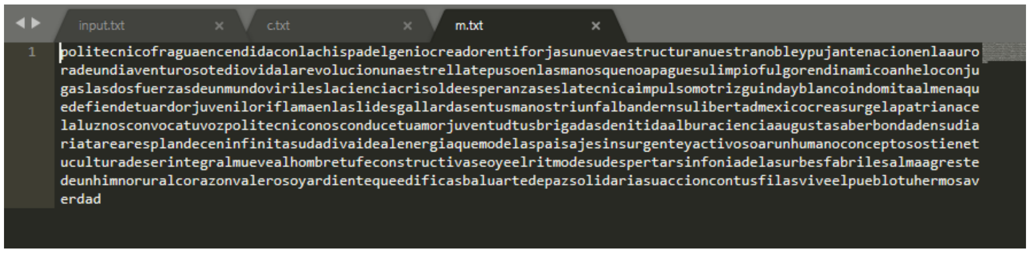

```
ZYVSDOMXSMYPBKQEKOXMOXNSNKMYXVKMRSCZKNOVQOXSYMBOKNYBOXDSPYBTKCEXEOFKOCDBEMDEBKXEOCDBKXYLVOIZETKXDOXKMSYXOXVKKEBY
BKNOEXNSKFOXDEBYCYDONSYFSNKVKBOFYVEMSYXEXKOCDBOVVKDOZECYOXVKCWKXYCAEOXYKZKQEOCEVSWZSYPEVQYBOXNSXKWSMYKXROVYMYXTE
QKCVKCNYCPEOBJKCNOEXWEXNYFSBSVOCVKMSOXMSKMBSCYVNOOCZOBKXJKCOCVKDOMXSMKSWZEVCYWYDBSJQESXNKILVKXMYSXNYWSDKKVWOXKAE
ONOPSOXNODEKBNYBTEFOXSVYBSPVKWKOXVKCVSNOCQKVVKBNKCOXDECWKXYCDBSEXPKVLKXNOBXCEVSLOBDKNWOHSMYMBOKCEBQOVKZKDBSKXKMO
VKVEJXYCMYXFYMKDEFYJZYVSDOMXSMYXYCMYXNEMODEKWYBTEFOXDENDECLBSQKNKCNOXSDSNKKVLEBKMSOXMSKKEQECDKCKLOBLYXNKNOXCENSK
BSKDKBOKBOCZVKXNOMOXSXPSXSDKCENKNSFKSNOKVOXOBQSKAEOWYNOVKCZKSCKTOCSXCEBQOXDOIKMDSFYCYKBEXREWKXYMYXMOZDYCYCDSOXOD
EMEVDEBKNOCOBSXDOQBKVWEOFOKVRYWLBODEPOMYXCDBEMDSFKCOYIOOVBSDWYNOCENOCZOBDKBCSXPYXSKNOVKCEBLOCPKLBSVOCKVWKKQBOCDO
NOEXRSWXYBEBKVMYBKJYXFKVOBYCYIKBNSOXDOAEOONSPSMKCLKVEKBDONOZKJCYVSNKBSKCEKMMSYXMYXDECPSVKCFSFOOVZEOLVYDEROBWYCKF
OBNKN
```

Fig. 10 The result of the encryption with a key of 10

The thing is, that if we want to decrypt the previous file, we cannot achieve a 100% fidelity with the original text, because we have lost the spaces and line breaks, but even if it is a unique string, every person could identify and separate the words to read the original message. This is shown in Fig. 11.



```
politecnicofraguaencendidaconlachispadelgeniocreadorentiforjasunuevaestructuranuestranobleypujantenacionenlaauro
radeundiaventurosotediovidalarevolucionunaestrellatepusoenlasmanosquenoapaguesulimpiofulgorendinamicoanheloconju
gaslasdosfuerzasdeunmundovirileslacienciacrisoldeesperanzaseslatecnicaimpulsomotrizguindayblancoindomitaalmenaqu
edefiendetuardorjuveniloriflamaenlaslidesgallardasentusmanostriunfalbandernsulibertadmexicocreasurgelapatrianace
laluznosconvocatuvozpolitecniconosconducetuamorjuventudtusbrigadasdenitidaalburacienciaaugustasaberbondadensudia
riatarearesplandeceninfinitasudadivaidealenergiaquemodelaspaisajesinsurgenteyactivosoarunhumanoconceptosostienet
uculturadeserintegralmuevealhombretufeconstructivaseoyeelritmodesudespertarsinfoniadelasurbesfabrilesalmaagreste
deunhimnoruralcorazonvalerosoyardientequeedificasbaluartedepazsolidariasuaccioncontusfilasviveelpueblotuhermosav
erdad
```

Fig. 11 The result of the decryption

Discussion:

The algorithm works correctly with any key. Every number bigger than 26 has an equivalent key that is find by calculating $eqKey = n \bmod 26$ for all n > 26. The results shows that we can find the encrypted message as a single string, which is good because if someone intercepts our channel of communication will not be able to understand the word in the first look. The problem I find is that the interceptor could make the inverse process, i.e. he could be able to jump before every single letter of the message and it will take 26 tries to find the original message.

If a person with cryptography knowledge intercepts the message, he could apply the frequency analysis [3], as we have seen before, and he could decrypt the message even faster than the previous interceptor.

To make this algorithm harder to decrypt, The *Substitution Cipher* was invented. [3].

Conclusions:

Thanks to this practice I had to remember how to use modular arithmetic, because it is the key of the sfhift-cipher algorithm. I realize that it's not that hard as I remember. I learned how to cipher a message using a substitution algorithm, which can be used to send secret messages between friends in the middle of the class, for example.

One of the things that we had to consider was to find an equivalent key if the user give one bigger than 26. Another possible improvement could be to transform the message received to lower-case, in that case we could ensure that we are not having lost of information, at least for all the letters.

References:

[1] S. Fernández, "La criptografía clásica", Sigma, no. 24, pp. 119-142, 2004.

[2] J. Ramió, "Seguridad Informática y Criptografía", Sistemas de Cifra Clásicos, Cap. 9, Universidad Politécnica de Madrid, pp. 343-383, 2006.

[3] N. Smart, Cryptography. London: McGraw-Hill, 2003.

Code

Class ShiftCipher.java

```java
package com.shiftcipher;

public class ShiftCipher {

    private String message;
    private int key;
    private final int keyInv;
    private final int factor1 = 97;
    private final int factor2 = 65;
    private final int toUpper = 65;
    private final int toLow = 97;

    public ShiftCipher(String message, int key) {
        this.message = message;
        this.key = key;
        this.keyInv = (26 - key);
    }

    public String encrypt() {

        String decMessage = new String();

        for (char c : message.toCharArray()) {
            if (c >= 97 && c <= 122) {
                decMessage += (char) (((c - factor1 + key) % 26) + toUpper);
            }
        }
        return decMessage;
    }

    public String decrypt() {

        String decMessage = new String();

        for (char c : message.toCharArray()) {
            if (c >= 65 && c <= 90) {
                c -= factor2;
                c = (char) ((c+keyInv)%26);
                decMessage += (char) (c + toLow);
            }
        }
        return decMessage;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public int getKey() {
        return key;
    }

    public void setKey(int key) {
        this.key = key;
    }

}
```

Class FileManager.java

```java
package com.filemanager;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

public class FileManager {

    private String dirPath = System.getProperty("user.dir");
    private File inputFile;
    String sourceData = "";

    public FileManager() {
    }

    public String openFile() {
        String aux = "";

        try {

            JFileChooser fileChooser = new JFileChooser(dirPath);
            fileChooser.showOpenDialog(null);
            inputFile = fileChooser.getSelectedFile();

            if (inputFile != null) {
                FileReader archivos = new FileReader(inputFile);
                BufferedReader br = new BufferedReader(archivos);
                while ((aux = br.readLine()) != null) {
                    sourceData += aux + "\n";
                }
                br.close();
            }
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null, ex + ""
                    + "\nNo se ha encontrado el archivo",
                    "ADVERTENCIA!!!", JOptionPane.WARNING_MESSAGE);
        }
        catch(NullPointerException n){

        }
        return sourceData;
    }

    public void saveToFile(String encData) {
        try {
            JFileChooser fileChooser = new JFileChooser(dirPath);
            fileChooser.showSaveDialog(null);
            File outputFile = fileChooser.getSelectedFile();

            if (outputFile != null) {
                FileWriter fw = new FileWriter(outputFile + ".txt");
                fw.write(encData);
                fw.close();
                JOptionPane.showMessageDialog(null,
                        "File " + outputFile.getName() + ".txt has been successful
ly saved",
                        "Info", JOptionPane.INFORMATION_MESSAGE);
            }
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null,
                    "Su archivo no se ha outputFiledo",
                    "Advertencia", JOptionPane.WARNING_MESSAGE);
        }
    }

    public String getFileName() {
        return inputFile.getName();
    }

    public String getFilePath() {
        try {
            return inputFile.getAbsolutePath();
        } catch (Exception e) {
        }
        return null;
    }
}
```

Class MainFrame.java

```
package com.gui;

import com.shiftcipher.ShiftCipher;
import com.filemanager.FileManager;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

public class MainFrame extends javax.swing.JFrame {

    private String inputText = "";
    String encData = null;
    String decData = null;


    public MainFrame() {
        setLocation(200, 100);
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {
```

```java

        titleLabel = new javax.swing.JLabel();
        inputTextField = new javax.swing.JTextField();
        browseButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        encryptButton = new javax.swing.JButton();
        decryptButton = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        keyTextField = new javax.swing.JTextField();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        titleLabel.setText("SHIFT - CIPHER");

        inputTextField.setEditable(false);
        inputTextField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                inputTextFieldActionPerformed(evt);
            }
        });

        browseButton.setText("Browse");
        browseButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                browseButtonActionPerformed(evt);
            }
        });

        jLabel1.setText("Selected File:");

        encryptButton.setText("Encrypt");
        encryptButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                encryptButtonActionPerformed(evt);
            }
        });

        decryptButton.setText("Decrypt");
        decryptButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                decryptButtonActionPerformed(evt);
            }
        });

        jLabel2.setText("Type the key:");

        keyTextField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                keyTextFieldActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPan
e());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(layout.createSequentialGroup()
                .addGap(39, 39, 39)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1)
                    .addComponent(jLabel2))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(keyTextField)
                        .addGap(104, 104, 104))
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(encryptButton)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 52, Short.MAX_VALUE)
                                        .addComponent(decryptButton))
                                    .addComponent(inputTextField, javax.swing.GroupLayout.Alignment.LEADING))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(browseButton)
                                .addGap(19, 19, 19))))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(titleLabel)
                        .addGap(167, 167, 167))
                );
                layout.setVerticalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(16, 16, 16)
                        .addComponent(titleLabel)
                        .addGap(33, 33, 33)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(inputTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel1)
                            .addComponent(browseButton))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel2)
                            .addComponent(keyTextField, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(encryptButton)
                            .addComponent(decryptButton))
                        .addContainerGap(29, Short.MAX_VALUE))
                );

                pack();
```

```java
    }// </editor-fold>//GEN-END:initComponents

    private void inputTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_inputTextFieldActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_inputTextFieldActionPerformed

    private void browseButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_browseButtonActionPerformed
        FileManager fm = new FileManager();
        inputText = fm.openFile();
        inputTextField.setText(fm.getFilePath());
    }//GEN-LAST:event_browseButtonActionPerformed

    private void encryptButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_encryptButtonActionPerformed
        if (!inputTextField.getText().isEmpty()&& !keyTextField.getText().isEmpty()) {
            int keyValue = Integer.parseInt(keyTextField.getText());
            if (keyValue > 26){
                keyValue = keyValue % 26;
                JOptionPane.showMessageDialog(this, "The key value is invalid. The equivalent key is "+ keyValue, "Info", JOptionPane.INFORMATION_MESSAGE, null);
            }
            try {
                ShiftCipher c = new ShiftCipher(inputText, keyValue);
                encData = c.encrypt();
                FileManager fm = new FileManager();
                fm.saveToFile(encData);
            } catch (Exception ex) {
                Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            clearForm();

        } else {
            JOptionPane.showMessageDialog(this, "Select an input file and give a key value", "Error", JOptionPane.ERROR_MESSAGE, null);
        }
    }//GEN-LAST:event_encryptButtonActionPerformed

    private void decryptButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_decryptButtonActionPerformed
        if (!inputTextField.getText().isEmpty() && !keyTextField.getText().isEmpty()) {
            int keyValue = Integer.parseInt(keyTextField.getText());
            if (keyValue > 26){
                keyValue = keyValue % 26;
                JOptionPane.showMessageDialog(this, "The key value is invalid. The equivalent key is "+ keyValue, "Info", JOptionPane.INFORMATION_MESSAGE, null);
            }
            try {
                ShiftCipher c = new ShiftCipher(inputText, keyValue);
                decData = c.decrypt();
                FileManager fm = new FileManager();
                fm.saveToFile(decData);
            } catch (Exception ex) {
                Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
            clearForm();

        } else {
            JOptionPane.showMessageDialog(this, "Select an input file and give a key value", "Error", JOptionPane.ERROR_MESSAGE, null);
        }
    }//GEN-LAST:event_decryptButtonActionPerformed

    public void clearForm(){
        inputTextField.setText("");
        keyTextField.setText("");
    }

    private void keyTextFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_keyTextFieldActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_keyTextFieldActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MainFrame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton browseButton;
    private javax.swing.JButton decryptButton;
    private javax.swing.JButton encryptButton;
    private javax.swing.JTextField inputTextField;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JTextField keyTextField;
    private javax.swing.JLabel titleLabel;
    // End of variables declaration//GEN-END:variables
}
```