King Saud university
Software engineering
Department
SWE314

Software Security Engineering (SWE314)

Third semester

-Project Report-

By

	Name	ID
1	Sultan Al-enzi	442106994
2	Khalid Al-harbi	442103477
3	Abdulmajeed Al-romaih	442101425

Supervision

by

Dr. Noureddine Abbadeni

Table of Contents

Section 1: Summary

The educational encryption/decryption app is a comprehensive tool that provides users with an introduction to encryption and decryption by allowing them to practice different algorithms. The report includes a use case model that outlines the app's primary purpose, which is to encrypt and decrypt messages. Additionally, the app uses pseudo-code to explain how each algorithm works and provides a class diagram that helps users understand the architecture of the app. Sample code with explanations is also included to show users how to implement the algorithms in actual code.

Finally, the app has limitations, as it is designed for educational purposes only and may not be suitable for real-world encryption.

Section 2: Introduction/Overview

The educational encryption and decryption application is a powerful tool designed to help users learn about encryption and decryption techniques. The application includes several popular algorithms such as monoalphabetic substitution, Playfair cipher, keyed transportation in both character and bit level, Vigenère cipher, Data Encryption Standard (DES), and a combination of Playfair and monoalphabetic substitution. These algorithms use different techniques to secure information and the application provides a simple interface for users to experiment with them. By using this tool, users can gain a better understanding of how encryption and decryption works and how they can use these techniques to protect sensitive information.

Section 3: Use case model

3.1 Use case diagram

The educational encryption and decryption use case model involves user entering a message, and key selecting an encryption algorithm, and encrypting the message. Same as with decryption.

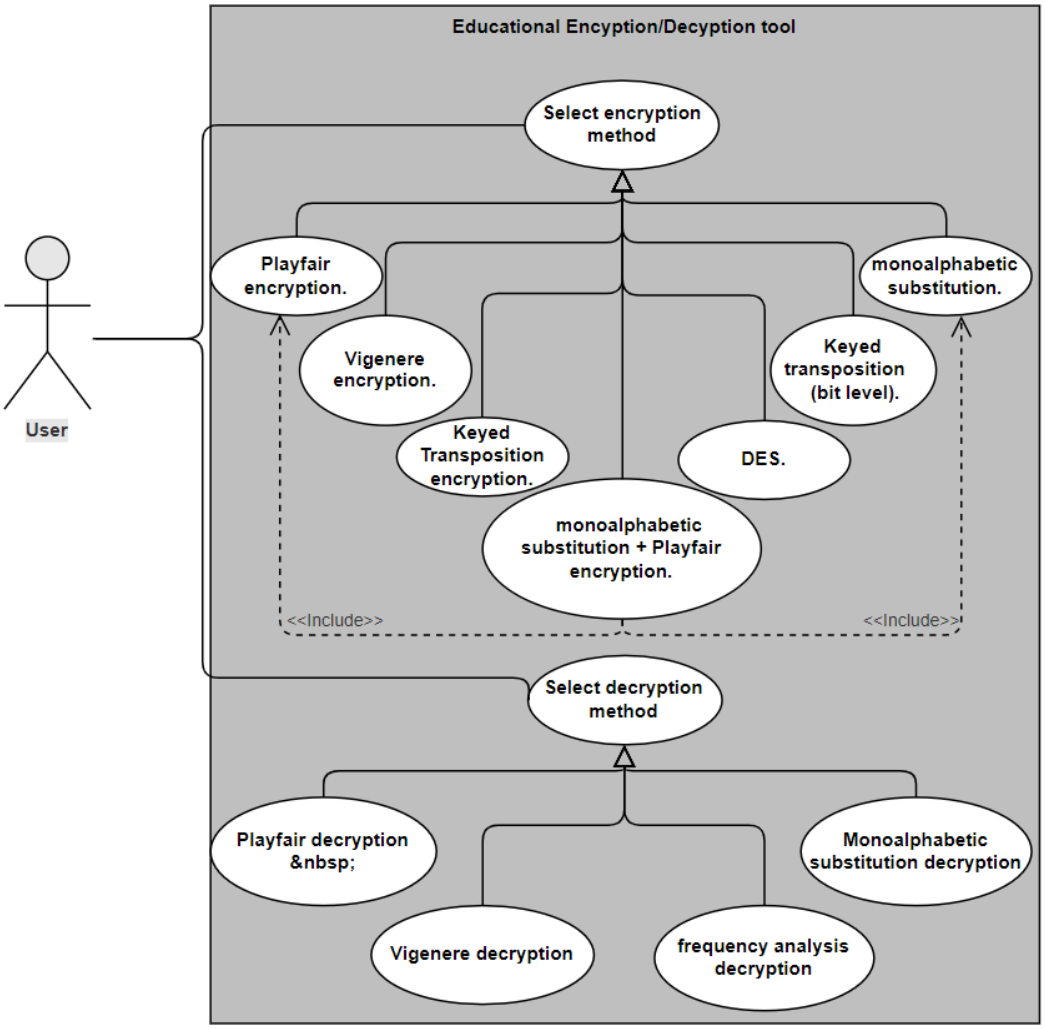

Figure 1: Use case diagram.

3.2 use case description

3.2.1 encryption use case

<table>
<tr><td colspan="2">Use case name: Encrypt</td></tr>
<tr><td colspan="2">Brief Description: This use case describes the scenario where the use can encrypt the text</td></tr>
<tr><td colspan="2">Actor: User</td></tr>
<tr><td colspan="2">Relationships:
Generalization:
Monoalphabetic substitution
Playfair encryption
Vigenère encryption
Keyed Transposition encryption
Monoalphabetic substitution + Playfair encryption
DES
Keyed Transposition (bits level)</td></tr>
<tr><td colspan="2">Preconditions: none</td></tr>
<tr><td colspan="2">Basic Flow:</td></tr>
<tr><td>User</td><td>System</td></tr>
<tr><td>1- The user asks the system to encrypt a plaintext.

3- The user provides the chosen algorithm.

5- The user provides the plaintext and key.</td><td>2- The system requests the user to choose the encryption algorithm:
Playfair encryption
Vigenère encryption
Keyed Transposition encryption
Monoalphabetic substitution + Playfair encryption
DES
Keyed Transposition (bits level)
Monoalphabetic substitution
4- The system requests the user to enter plaintext and key
6- The system encrypts the plaintext based on algorithm and key provided.

7- The system displays the original plaintext

8- The system displays the encrypted plaintext and shows other encryption option</td></tr>
<tr><td colspan="2">Post Conditions: Provide a ciphertext</td></tr>
</table>

3.2.2 Decryption use case description.

<table>
<tr><td colspan="2">Use case name: decrypt</td></tr>
<tr><td colspan="2">Brief Description: This use case describes the scenario where the use can decrypt the text</td></tr>
<tr><td colspan="2">Actor: User</td></tr>
<tr><td colspan="2">Relationships:
Generalization:
Monoalphabetic substitution
Playfair decryption
Vigenère decryption
Keyed Transposition decryption
Monoalphabetic substitution + Playfair encryption
DES
Keyed Transposition (bits level)</td></tr>
<tr><td colspan="2">Preconditions: none</td></tr>
<tr><td colspan="2">Basic Flow:</td></tr>
<tr><td>User</td><td>System</td></tr>
<tr><td>1- The user asks the system to decrypt an encrypted text.

3- The user provides the chosen algorithm.

5- The user provides the encrypted text and key.</td><td>2- The system requests the user to choose the decryption algorithm:
Playfair encryption
Vigenère encryption
Keyed Transposition decryption
Monoalphabetic substitution + Playfair decryption
DES
Keyed Transposition (bits level)
Monoalphabetic substitution
4- The system requests the user to enter encrypted text and key
6- The system decrypt the encrypted text based on algorithm and key provided.

7- The system displays the encrypted text.

8- The system displays the decrypted text</td></tr>
<tr><td colspan="2">Post Conditions: Provide a decrypted text</td></tr>
</table>

Section 4: Pseudo-code of the different algorithms

In this section, we will describe the pseudo-code for different algorithms, focusing on the main functionalities of each algorithm.

4.1 monoalphabetic encryption/decryption:

Monoalphabetic encryption is a simple substitution cipher where each letter in the plaintext is replaced with a corresponding letter from a fixed substitution table. The pseudo-code for monoalphabetic encryption is as follows:

Monoalphabetic substitution encryption pseudo code:

```
ARRAY original chars [26]

ARRAY key chars [26]

FUNCTION encrypt(plaintText )
        VAR encryptedtString = null
        FOR i each index of plaintText
                FOR j = 0 to 26
                        var s = plaintText.charAt(i)
                        IF(S equal to original[j])
                                encryptedtString = encryptedtString + key[j]
                                break
                        END IF
                END LOOP
        END LOOP
        RETURN encryptedtString
END FUNCTION
```

Monoalphabetic substitution Decryption pseudo code:

```
ARRAY original chars[26]
ARRAY key chars[26]
FUNCTION decrypt( encryptedText )
        VAR decryptedtString = null
        FOR i each index of plaintText
                FOR j = 0 to 26
                        var s = encryptedText.charAt(i)
                        IF(s equal to key[j])
                                decryptedtString = decryptedtString + original[j]
                                break
                        END IF
                END LOOP

        END LOOP
        RETURN encryptedtString
END FUNCTION
```

4.2: Playfair encryption/decryption algorithm:

Playfair encryption is a digraph substitution cipher that uses a 5x5 matrix based on a keyword to encrypt pairs of letters in plaintext. The pseudo-code for Playfair encryption is as follows:

4.2.1 Playfair instructions:

1 - split the text into pairs

2 - if there is one alone letter (number of letters are odd) we shall

add Z letter into with the last letter.

3 - the pair shall not be same letter so replace by X letter.

4 - if the pair in same row of matrix we shall replace every letter by

the right of each

5 - if the letters in same column we shall replace every letter by letter in bellow in matrix

6 - if the pair is not in the same row and column we shall replace

every letter the letter in opposite from right direction

4.2.2 Playfair encryption/decryption algorithm:

Generate the key and plaint text:

Step 1: define key, plaint text variables.

Step 2: define 2D char array with size 5 x 5.

Step 3: take the key and plaint text from user then assign them to variables.

Step 4: remove duplicate letters from key variable.

step 5: convert key to char array.

step 6: define engLetters variable contains the English letters.

Step 7: remove the letters of key that contains in engLetters variable.

step 8: combine the key variable (first) then engLetters in 2D char array.

step 9: convert the plaint text into pairs.

step 10: if there is pair containing same letter change last to X letter.

step 11: if the number of letters in plaint text odd, then add last z

-Now we must define function that search in matrix the position of letters, the function shall receive the matrix and the pair (a and b).

step 1: define array y with size of 4 to store the row position and column position for both of pair letters.

step 2: if one of letters are j make the letter equals to i letter.

step 3: now go through the whole matrix and search for element that equals a or b.

step 4: if you find the position of a variable store it in y array y[0] = row position and y[1] = column position , same thing with b variable store in y[2] for row and y[3] for column.

- now we know y[0] and y[2] for row, y[1] and y[3] for column -

step 5: now we must check if the pair is in same row, if they in same row increase the column by 1 (to get the next element from right)

step 6: now we must check if the pair is in same column, if they in same column increase the row by 1 (to get the next element from bottom)

step 7 after generating the y array we have to check if y[4] equals to 5 then reduce it to 1 (cycling).

then return the y.

4.2.3 Encryption steps:

step 1: define x char array then add the plaint text into it

step 2: define int array size of 4 that will store the positions of a and b

step 3: call the function find by sending the x array and pair and receiving positions of pair.

step 4: if two variables in same row take the next elements from right to both

step 5: if two variables in same column take the next elements from bottom to both

step 6: if the pair is not in same column or row take the opposite element from right or left.

4.2.4 Decryption steps:

step 1: define char array then add the encrypted text into it

step 2: define int array size of 4 that will store the positions of a and b

step 3: call the function find by sending the x array and pair and receiving positions of pair.

step 4: if two variables in same row take the next elements from left to both

step 5: if two variables in same column take the next elements from up to both

step 6: if the pair is not in same column or row take the opposite element from left or right.

4.2.5 Playfair algorithm:

INITIAL plaintText input from user

INITIAL key input from user

INITIAL ketMatrix char[][] array

FUNCTION removeDuplicate()

INITIAL s, j, index set to zero, len set to length of plaintText

INITIAL array c set to plaintText chars

LOOP i from 0 to length

    LOOP j from 0 to i

        IF c[i] equals c[j]

            BREAK.

    END LOOP

    IF i equals to j

    c[index++] = c[i]

END LOOP

RETURN c

END FUNCTION

```
FUNCTION removeSpacesAndDuplicates()
INITIAL c[] equals to key chars
INITIAL ch[] equals to key chars


LOOP i from 0 to c length
        LOOP j from 0 to ch length
                IF c[i] equals to ch[j]
                        c[i] = ' '
                END IF
        END LOOP
END LOOP
set key equals to new String of c
remove spaces from key.
return key.
END FUNCTION

FUNCTION splitIntoPair(plaintText)
INITIAL s set to null
LOOP i from 0 to plaintText length
        s = s + plainText.charAt[i]
        IF I < plaintText length - 1 then
                IF plaintText.charAt[i] equals to plaintText.charAt[i+1]
                        s = s + "x"
                END IF
        END IF
END LOOP
IF length of s is odd then
        s = s+ "z"
END IF
RETURN s
END FUNCTION
```

```
FUNCTION find(a , b , x[][])
INITIAL Y[4]
IF a equals to 'j'
        a = 'i'
END IF
IF b equals to 'j'
        b = 'i'
END IF
LOOP i from 0 to 4
        LOOP j from 0 to 4
                IF x[i][j] equals to a
                        y[0] equals i
                        y[1] equals j
                ELSE IF x[i][j] equals to b
                        y[2] equals i
                        y[3] equals j
                END IF
        END LOOP
END LOOP
IF Y[0] equals to y[2]
        INCEASE y[1] by 1
        INCEASE y[3] by 1
END IF
IF Y[1] equals to y[3]
        INCEASE y[0] by 1
        INCEASE y[2] by 1
END IF
IF y[4] equals to 5
        then set y[4] to 0
END IF
RETURN Y
END FUNCTION
```

```
FUNCTION encrypt (plaintText , x[][])
INITIAL ch[] equals to chars of plainText
INITIAL a[4]
LOOP i from 0 to plaintText length
        IF i < pt length - 1 then
                a equals to calling the FUNCTION find(pt[i] , pt[i+1] , x)
        END IF
        IF a[0] equals to a[2] then
                ch[i] equals x[a[0]] [a[1]]
                ch[i + 1 ] equals x[a[0]] [a[3]]
        ELSE IF a[1] equals to a[3]
                ch[i] = x[a[0]][a[1]]
                ch[i + 1] = x[a[2]][a[1]]
        ELSE
                ch[i] = x[a[0]][a[3]];
                ch[i + 1] = x[a[2]][a[1]];
        END IF
END LOOP
plaintText = new String of ch
RETURN PlaintText
END FUNCTION
```

4.3: Encryption/decryption by combining monoalphabetic substitution and Playfair cipher algorithm:

We can combine the monoalphabetic and Playfair techniques. In this process, the plaintext is first encrypted using monoalphabetic substitution. Then, the resulting ciphertext is further encrypted using the Playfair cipher with a keyword and a 5x5 matrix.

```
DECLARE monoCipher AS MonoalphabeticCypher
DECLARE playfairDecryption AS PlayfairDecryption
DECLARE playfairEncryption AS PlayfairEncryption
DECLARE cipherText AS String


PROCEDURE CombiningMonoalphabeticAndPlayfair()
    monoCipher = new MonoalphabeticCypher()
    playfairEncryption = new PlayfairEncryption()
    playfairDecryption = new PlayfairDecryption()
END PROCEDURE

FUNCTION encryptUsingMonoAlphabeticAndPlayfair(PlaintText, key)
    cipherText = monoCipher.stringEncryption(PlaintText)
    cipherText = playfairEncryption.encryptUsingplayfair(cipherText, key)
    RETURN cipherText
END FUNCTION

FUNCTION decryptUsingMonoAlphabeticAndPlayfair(PlaintText, key)
    cipherText = playfairDecryption.decryptUsingplayfair(PlaintText, key)
    cipherText = monoCipher.stringDecryption(cipherText)
    RETURN cipherText
END FUNCTION
```

4.4 Encryption/decryption by Vigenère cipher pseudo-code:

The Vigenère cipher uses a 26×26 table with A to Z as the row heading and column heading. The pseudo-code for Vigenère encryption is as follows:

```
INITIAL Vigenère input from user
INITIAL key input from user
  FUNCTION encrypt(plaintext)
    ciphertext = ""
    keyIndex = 0
    LOOP character c in plaintext
      IF c is a letter
        shift = toUpperCase(key[keyIndex]) - 'A'
        encryptedChar = ((toUpperCase(c) - 'A' + shift) % 26) + 'A'
        append encryptedChar to ciphertext
        keyIndex = (keyIndex + 1) % length(key)
      ELSE
        append c to ciphertext
      END IF
    END LOOP
    RETURN ciphertext
END FUNCTION
```

FUNCTION DECRYPT(ciphertext)

This pseudo code will describe Vigenère decryption. The pseudo-code for Vigenère decryption is as follows:

```
    keyIndex = 0
    LOOP character c in ciphertext
      IF c is a letter
        shift = toUpperCase(key[keyIndex]) - 'A'
        decryptedChar = ((toUpperCase(c) - 'A' - shift + 26) % 26) + 'A'
        append decryptedChar to plaintext
        keyIndex = (keyIndex + 1) % length(key)
      ELSE
        append c to plaintext
      END IF
    END LOOP
    RETURN plaintext
END FUNCTION
```

4.5 Frequency analysis pseudo-code:

Input ciphertext → String ciphertext= input.next

Make ciphertext uppercase → ciphertext.toUpperCase()

An array type string of size 3 to store the 3 possibilities of plaintext → String[] plaintext = new String[3];

String to store the most used letter "ETAOINSHRDLCUMWFGYPBVKJXQZ" → String T= "ETAOINSHRDLCUMWFGYPBVKJXQZ"

An array type int of size 26 to store frequency letters from cipher text → int freq[] = new int[26]

An array type int of size 26 to store the frequency of letter → int freqSort[]= new int [26]

An array type int of size 26 to store used letters → int used[]=new int[26]

Loop "i=0" less than length ciphertext :

If char not space → ciphertext.chart(i)! = ' '

Store it to frequency letter++ → freq[ciphertext.chart(i) – 'A']++

End Loop

Loop "i=0" less than 26:

Copy the value of frequency-to-frequency sort → freqSort[i] = freq[i]

End Loop

Sort freqSort → Arrays.sort(frqSort)

Loop "i=0" less than 3:

Set ch to -1 → int ch = -1

Loop "k=0" less than 26:

If freqSort of i equal freq of k and used of k equal to zero → freqSort[i] == freq[k] && Used[k] == 0

Used[k] =1;

Ch = k;

Break;

End if

End Loop

If ch is still -1 → ch == -1

break out of the main loop. → break;

Set x as T char i – ‘A’ → int x = T.charAt(i) – ‘A’

Calculate x as x minus ch. → x = x-ch

Create an empty string called current. → String current = “”

Loop “j=0” less than length of ciphertext:

If character j in the ciphered text is a space, add a space to current and continue with the next iteration. → ciphertext.charAt(j) == ‘ ’

Current += (char) ‘ ’;

Continue.

End If

Set y to character k in the ciphered text minus 'A'.

Add x to y → y+=x

If y is less than zero → y<0

add 26 to it → y+=26

If y is greater than 25 → y>26

subtract 26 from it → y-=26

Add character 'A' plus y to current → current+=(char) (‘A’ +y)

Store current to plaintext[i].

End Loop (main)

4.6 Keyed transposition encryption (char-level) pseudo code:

Keyed transposition is a method of encryption that rearranges the order of characters in the plaintext based on a secret key. The following pseudo-code will describe the instructions:

1-Input plaintext → String plaintext= input.next()

2-Store plaintext to String str → String str= plaintext

3-While str.length() % 4:

Add z to str → str+= "z"

End While

4-Input key must be 4 digits from 0-3 and not repeating.

function encrypt(String plaintext, int[] key) {

5-Set an array of char to size length of str → char [] MsgBasedOnKey= new char [str. length]

6-Set an array of char to str of chars → char[] Letters = plaintext.toCharArray();

7-Set "i" to zero → int i=0

8-While "i" less than length of str: → i < plaintext.length()

Sort array based on key given.

First 4 letters only

MsgBasedOnKey[key[0] +i]= letter[i+0]

MsgBasedOnKey[key[1] +i]= letter[i+1]

MsgBasedOnKey[key[2] +i]= letter[i+2]

MsgBasedOnKey[key[3] +i]= letter[i+3]

Increment "i" by 4 → i+=4

End While

9-Print out MsgbasedOnKey

}

4.7 Keyed transposition encryption (bit-level) pseudo code:

Keyed transposition is a method of encryption that rearranges the order of characters in the plaintext after converting plaintext to bits, based on a secret key. The following pseudo-code will describe the instructions:

1-Input plaintext → String plaintext= input.nextLine()

2-Store plaintext to String str → String str= plaintext

3-While str.length() % 4:

Add z to str → str+= "z"

End While

4-Input key must be 4 digits from 0-3 and not repeating.

5-Convert plaintext to bits

String result = "";

Store messChar array to plaintext chars → Char[] messChar = plaintext.toCharArray()

Loop "i=0" less than messChar length: → i < meesChar.length

result+= String.*format*("%8s", Integer.*toBinaryString*(messChar[i]).replace(' ','0'));

End loop

function encrypt(String plaintext, int[] key) {

6-Set an array of char to size length of str → char [] MsgBasedOnKey= new char [str. length]

7-Set an array of char to str of chars → str.toCharArray() (method via java)

8-Set "i" to zero

9-While "i" less than length of str

MsgBasedOnKey[key[0] +i]= letter[i+0]

MsgBasedOnKey[key[1] +i]= letter[i+1]

MsgBasedOnKey[key[2] +i]= letter[i+2]

MsgBasedOnKey[key[3] +i]= letter[i+3]

Increment "i" by 4 → i+=4

End While

10- Print out MsgbasedOnKey

Section 5: Class Diagram

The class diagram consists of a 'Main' class serving as the entry point of the application, connected to other classes such as Monoalphabetic, Playfair Cipher, Vigenère, Frequency Analysis, Keyed transposition, and DES, that handle data encryption and decryption respectively.

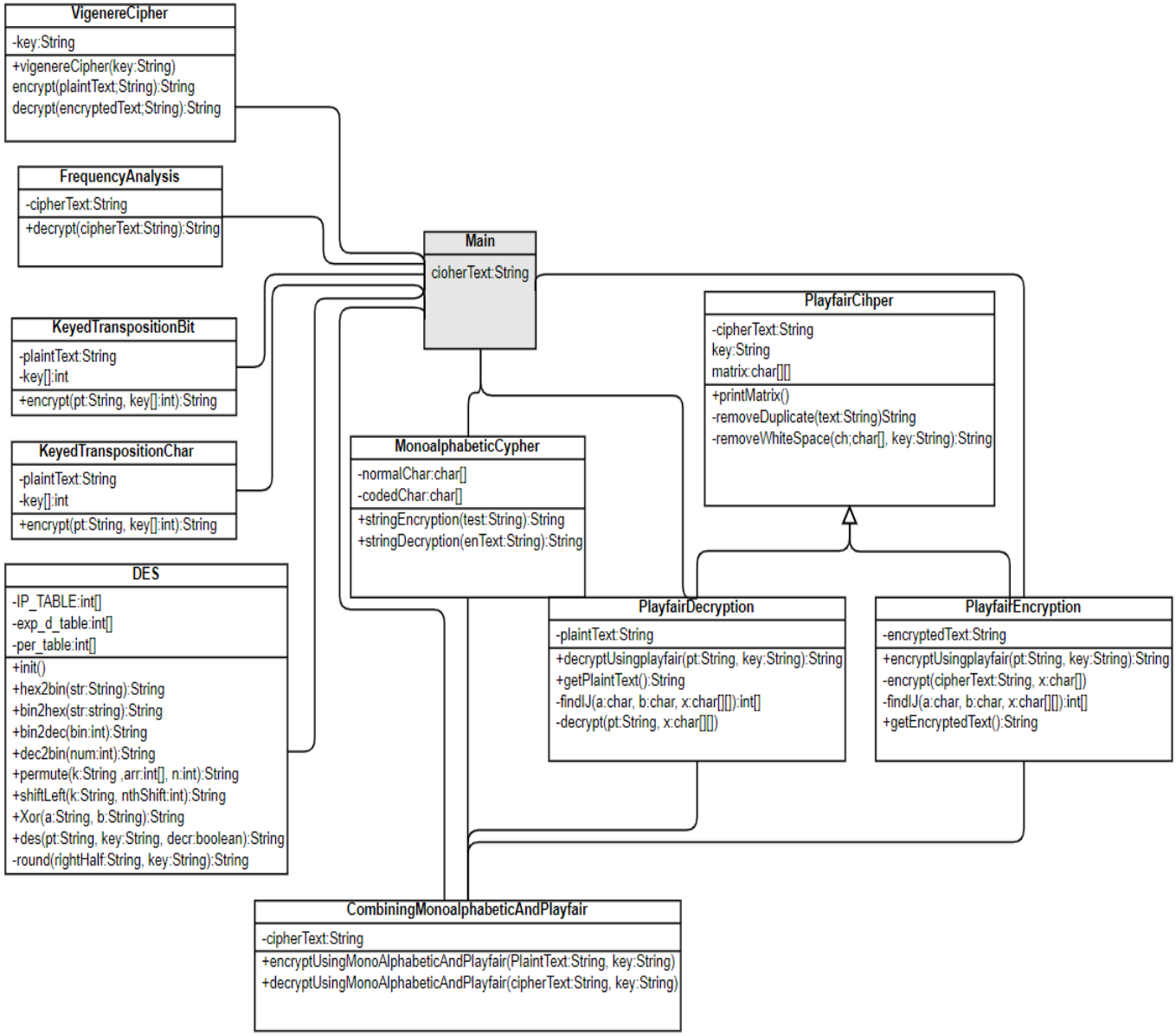

Figure 2 Class diagram

Section 6: Samples of code with appropriate explanations

6.1 Monoalphabetic substitution code:

These are two arrays represent the normal letters and the key that we will used to encrypt/decrypt.

```
public static char normalChar[]
    = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
        'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
        's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };

public static char codedChar[]
    = { 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
        'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
        'U', 'V', 'W', 'X', 'Y', 'Z', 'A', 'B' };
```

Figure 3 the letters that will exchange with original letter.

Here is the encryption method, first, it will receive the plaintext in variable, and it will take each letter in plaintext and replace it by the key letter and store it into encryptedString.

```
// Function which returns encrypted string
public  String stringEncryption(String s)
{
    // initializing an empty String
    String encryptedString = "";



    // comparing each character of the string and
    // encoding each character using the indices
    for (int i = 0; i < s.length(); i++) {
        for (int j = 0; j < 26; j++) {

            // comparing the character and
            // adding the corresponding char
            // to the encryptedString
            if (s.charAt(i) == normalChar[j])
            {
                encryptedString += codedChar[j];
                break;
            }

            // if there are any special characters
            // add them directly to the string
            if (s.charAt(i) < 'a' || s.charAt(i) > 'z')
            {
                encryptedString += s.charAt(i);
                break;
            }
        }
    }

    // return encryptedString
    return encryptedString;
}
```

To check the position of letter to replace it by the key.

Figure 4 monoalphabetic encryption method.

For educational purposes, the system has a promising future where users can choose how to implement the given tool by printing the instructions of the algorithm. In this case, users can print the instructions of the monoalphabetic substitution encryption.

```
1
Enter the plaint text that you want to encrypt
where are you now
The plaint text is: where are you now
The encypted text using monoalphabetic is: YJGTG CTG AQW PQY
*** *** *** *** *** *** *** ***
Press 1 to decrypt the message
Press 2 to print the encryption instructions of monoalphabetic
Press 3 to print the decryption instructions of monoalphabetic
Press 4 to store the encrypted text into the system
Press -1 to exit the function
*** *** *** *** *** *** *** ***
2
1. Choose a secret key: This is a random sequence of letters that will be used to
 encrypt and decrypt messages.

2. Create a table: Create a table with two columns, one for the plaintext alphabet
 and one for the corresponding ciphertext alphabet.

3. Substitute letters: Substitute each letter in the plaintext alphabet with a
 corresponding letter in the ciphertext alphabet using the secret key.

4. Encrypt message: To encrypt a message, replace each letter in the plaintext
 message with its corresponding ciphertext letter from the table.

5. Decrypt message: To decrypt a message, replace each letter in the ciphertext
 message with its corresponding plaintext letter from the table using the secret key.

*** *** *** *** *** *** *** ***
Press 1 to decrypt the message
Press 2 to print the encryption instructions of monoalphabetic
Press 3 to print the decryption instructions of monoalphabetic
Press 4 to store the encrypted text into the system
Press -1 to exit the function
*** *** *** *** *** *** *** ***
```

User can choose to print the monoalphabetic encryption instructions.

6.2 Playfair cipher

Before generating the key, we have to remove any spaces and duplicate letters. So, this method will receive the real key in char array and all English letters, then it will remove the duplicate letters and spaces.

```
// Removing the white spaces from string 'st'
// which was replaced by the key as space.
public String removeWhiteSpace(char[] ch, String key)
{

    char[] c = key.toCharArray();

    // removing character which are input by the user
    // from string st

    for (int i = 0; i < c.length; i++) {

        for (int j = 0; j < ch.length; j++) {

            if (c[i] == ch[j])

                c[i] = ' ';
        }
    }

    key = new String(c);

    key = key.replaceAll(" ", "");


    return key;

}
```

Remove the duplicates letters from key.

Remove the spaces from key.

Figure 5 remove the spaces and duplicates letters from key.

Now, after we remove the spaces and duplicates. We should split the plaint text into pairs. The method will divide the text into pairs. This method will check if the pair is symmetric or not, if the pair is same letter then will change the second letter by x, and if the length of text is odd it will add z letter at last.

```
public String splitIntoPair(String str) {
    String s  = "";
    //to check the pair is not same
            for (int i = 0 ; i < str.length();i++) {
                    if (str.charAt(i) == ' ')
                        //to jump from spaces
                        continue;

                    else {
                        s += str.charAt(i);
                    }
                    if (i < str.length() - 1)
                        if (str.charAt(i) == str.charAt(i + 1))

                            s += "x";
                }

            if (s.length() % 2 != 0)
                s += "z";
            s = s.toLowerCase();

            return s;
    }
```

If we found same letters, we add x to the pair.

If the length of message is odd, add z letter at last.

Figure 6 split the text into pairs.

After generating the key, we have to put the key in matrix 5 x 5 and put the remaining English characters after.

```
int indexOfSt = 0, indexOfKey = 0;
for (int i = 0; i < 5; i++) {
    // this function to fill the matrix with key in first
    //then fill the remain alphabets letters

    for (int j = 0; j < 5; j++) {

        if (indexOfKey < key.length())

            x[i][j] = ch[indexOfKey++];

        else

            x[i][j] = c[indexOfSt++];
    }
}
```

This condition to check if the index of key became bigger than the key, that means the key letters filled in the matrix. then refill the remaining English letters.

Figure 7 Inserting the key and remaining letters in matrix 5 x 5.

After done from the key now we have the last step to encrypt the text.

Note: a [0] is the row of element 1, a [1] is column of element 1, a [2] is the row of element 2, a[3] is column of element 2.

```
public String encrypt(String pt, char x[][])
{
    char ch[] = pt.toCharArray();
    int a[] = new int[4];

    for (int i = 0; i < pt.length(); i += 2) {

        if (i < pt.length() - 1) {

            a = findIJ(pt.charAt(i), pt.charAt(i + 1),x);

            if (a[0] == a[2]) {

                ch[i] = x[a[0]][a[1]];
                ch[i + 1] = x[a[0]][a[3]];
            }

            else if (a[1] == a[3]) {

                ch[i] = x[a[0]][a[1]];
                ch[i + 1] = x[a[2]][a[1]];
            }

            else {

                ch[i] = x[a[0]][a[3]];
                ch[i + 1] = x[a[2]][a[1]];
            }
        }
    }

    pt = new String(ch);


    return pt;
}
```

This method returns the position of the pair in matrix.

If the pair in same row, then take the element from right position.

This condition means the pair letters are in same column, take the above letter.

The pair letters are not in same column or row, take opposite element.

Figure 8 Encryption method using Playfair.

6.3 Keyed Transposition chars

The plaintext is encrypted by rearranging the characters according to the key. Each character's position is switched with a new position determined by the key, resulting in a transformed ciphertext.

```
public static void encrypt(String plaintext, int[] key) {
    String encryptedMsg;
    char[] MsgBasedOnKey = new char[plaintext.length()];
    char[] Letters = plaintext.toCharArray();
    int i = 0;
    while (i < plaintext.length()) {
        MsgBasedOnKey[key[0] + i] = Letters[i + 0];
        MsgBasedOnKey[key[1] + i] = Letters[i + 1];
        MsgBasedOnKey[key[2] + i] = Letters[i + 2];
        MsgBasedOnKey[key[3] + i] = Letters[i + 3];
        i += 4;
    }
    System.out.println(MsgBasedOnKey);

}
```

The function accepts 2 parameters plaintext and key

The process of the code is to sort the cipher text based on key provided.

6.4 Keyed Transposition bits level

As stated, before keyed transposition bits is same as keyed transposition characters but the only difference is we convert the plaintext into bits which make harder to decrypt with the modern block

```
System.out.println("Enter digits: ");
String digit = input.nextLine();
while (digit.length() != 4 || unique(digit) == false) {
    System.out.println("enter the key as 4 digits only(e.g. 3210,1230,0312)");
    digit = input.nextLine();

}
int num = Integer.parseInt(digit);
int key[] = new int[4];

for (int i = 3; i > -1; i--) {
    key[i] = num % 10;
    num = num / 10;
}

String result = "";
char[] messChar = str.toCharArray();

for (int i = 0; i < messChar.length; i++) {
    result += String.format("%8s", Integer.toBinaryString(messChar[i]).replace(' ','0'));
    //result += Integer.toBinaryString(messChar[i]) + " ";
}
```

The only thing we changed from previous code is to convert string to bit by this method.

```
public static void encrpyt(String plaintext, int[] key) {

    char[] MsgBasedOnKey = new char[plaintext.length()];
    char[] Letters = plaintext.toCharArray();
    int i = 0;
    while (i < plaintext.length()) {
        MsgBasedOnKey[key[0] + i] = Letters[i + 0];
        MsgBasedOnKey[key[1] + i] = Letters[i + 1];
        MsgBasedOnKey[key[2] + i] = Letters[i + 2];
        MsgBasedOnKey[key[3] + i] = Letters[i + 3];
        i += 4;
    }

    System.out.println(MsgBasedOnKey);
}
```

Section 7: Test cases

Section 8: A series of executions showing the input and the output of the program:

8.1 Monoalphabetic substitution execution

In this execution it will display both encryption and decryption.

This is the main page where the user will choose the monoalphabetic substitution by pressing number 1.

```
*****************************************************
Press 1 to Encrypt using monoalphabetic subtitution
Press 2 to Encrypt using playfair
Press 3 to Encrypt using playfair and monoalphbetic cipher
Press 4 to Encrypt using Keyed trasposition
Press 5 to to Encrypt using keyed tranposition (bit-level)
Press 6 to Encrypt using vigenere cipher
Press 7 to Encrypt using DES
Press 8 to Decrypt using frequency analysis*
Press 9 to Display the encrypted texts
*******  Press -1 if u want to exit  *******
*****************************************************
```

Figure 9 main page

After selecting monoalphabetic substitution, the user will have the option to enter the text that want to encrypt. The system will then display both the plaintext and the encrypted text. Furthermore, the system will offer several features, including decrypting the message, printing encryption/decryption instructions for educational purposes, and storing the encrypted text for future reference.

```
1
Enter the plaint text that you want to encrypt
where are you now
The plaint text is: where are you now
The encypted text using monoalphabetic is: YJGTG CTG AQW PQY
*** *** *** *** *** *** *** ***
Press 1 to decrypt the message
Press 2 to print the encryption instructions of monoalphabetic
Press 3 to print the decryption instructions of monoalphabetic
Press 4 to store the encrypted text into the system
Press -1 to exit the function
*** *** *** *** *** *** *** ***
1
The encrypted text is: YJGTG CTG AQW PQY
The decrypted text using monoalphabetic is: where are you now
*** *** *** *** *** *** *** ***
Press 1 to decrypt the message
Press 2 to print the encryption instructions of monoalphabetic
Press 3 to print the decryption instructions of monoalphabetic
Press 4 to store the encrypted text into the system
Press -1 to exit the function
*** *** *** *** *** *** *** ***
4
*** *** *** *** *** *** *** ***
Press 1 to decrypt the message
Press 2 to print the encryption instructions of monoalphabetic
Press 3 to print the decryption instructions of monoalphabetic
Press 4 to store the encrypted text into the system
Press -1 to exit the function
*** *** *** *** *** *** *** ***
-1
```

Figure 10 execution the monoalphabetic encryption method

8.2 Playfair cipher execution

During the execution, the user selects encryption using the Playfair cipher by entering the number 2 and providing the text their want to encrypt. Additionally, the user have to enter a key of their choice. If the key is provided, the word will be placed at the beginning of the matrix, and the remaining English letters will be added to the matrix. After encryption that the user chosen, the user enter decrypt the encrypted message.

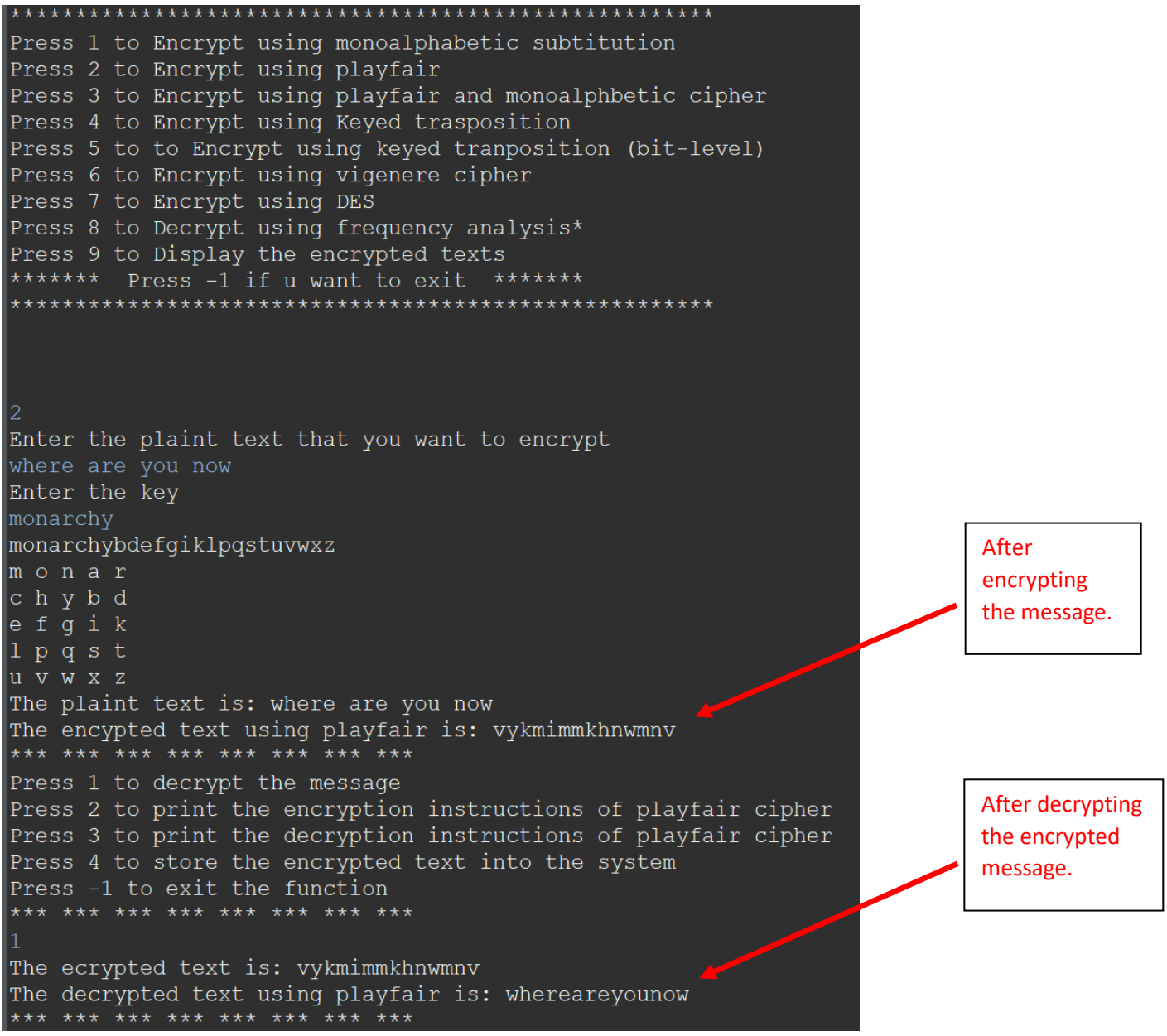

Figure 11 Playfair encryption execution

8.3 Encryption by using both monoalphabetic and Playfair execution.

During the execution, the user selects encryption using both the Playfair cipher and monoalphabetic substitution. The user enters the message and the key for the Playfair cipher. The system then prints the encrypted message and displays it. Additionally, the system provides several features.

```
**************************************************
Press 1 to Encrypt using monoalphabetic subtitution
Press 2 to Encrypt using playfair
Press 3 to Encrypt using playfair and monoalphbetic cipher
Press 4 to Encrypt using Keyed trasposition
Press 5 to to Encrypt using keyed tranposition (bit-level)
Press 6 to Encrypt using vigenere cipher
Press 7 to Encrypt using DES
Press 8 to Decrypt using frequency analysis*
Press 9 to Display the encrypted texts
*******  Press -1 if u want to exit  *******
**************************************************



3
Enter the plaint text that you want to encrypt using combination between mone and playfair
where are you now
Enter the playfair key
monarchy
The plaint text is: where are you now
after using playfair vykmimmkhnwmnv
after using mono XAMOKOOMJPYOPX
The encypted text using playfair and mono is: XAMOKOOMJPYOPX
*** *** *** *** *** *** *** ***
Press 1 to decrypt the message
Press 2 to print the encryption instructions of mono and playfair cipher
Press 3 to print the decryption instructions of mono and playfair cipher
Press 4 to store the encrypted text into the system
Press -1 to exit the function
*** *** *** *** *** *** *** ***
```

Figure 12 execution both mono and Playfair

8.4 Keyed transposition execution

During the execution, the user selects encryption using keyed transposition. The user has the option to enter the message they want to encrypt. The system prints the key used to encrypt the message and displays the encrypted message.

```
**************************************************
Press 1 to Encrypt using monoalphabetic subtitution
Press 2 to Encrypt using playfair
Press 3 to Encrypt using playfair and monoalphbetic cipher
Press 4 to Encrypt using Keyed trasposition
Press 5 to to Encrypt using keyed tranposition (bit-level)
Press 6 to Encrypt using vigenere cipher
Press 7 to Encrypt using DES
Press 8 to Decrypt using frequency analysis*
Press 9 to Display the encrypted texts
*******  Press -1 if u want to exit  *******
**************************************************



4
Enter the plaint text that you want to encrypt
where are you now
The key is: 1023
The plaint text is: where are you now
The encrypted message is: hwer ear eyo unozwzz
*** *** *** *** *** *** *** ***
Press 1 to decrypt the message
Press 2 to print the encryption instructions of keyed transposition cipher
Press 3 to print the decryption instructions of keyed transposition  cipher
Press 4 to store the encrypted text into the system
Press -1 to exit the function
*** *** *** *** *** *** *** ***
```

Figure 13 Keyed transposition encryption exception

8.5 Vigenère encryption execution.

During the execution, the user selects encryption using the Vigenère cipher by entering the number 6. The user provides both the message and the key. The system will print the encrypted text. Additionally, the user chose to decrypt the message that was encrypted during the execution.

```
*******************************************************
Press 1 to Encrypt using monoalphabetic subtitution
Press 2 to Encrypt using playfair
Press 3 to Encrypt using playfair and monoalphbetic cipher
Press 4 to Encrypt using Keyed trasposition
Press 5 to to Encrypt using keyed tranposition (bit-level)
Press 6 to Encrypt using vigenere cipher
Press 7 to Encrypt using DES
Press 8 to Decrypt using frequency analysis*
Press 9 to Display the encrypted texts
*******  Press -1 if u want to exit  *******
*******************************************************



6
Enter the plaint text that you want to encrypt
where are you now
Enter the key
monarchy
The plaint text is: where are you now
The encrypted text is: IVRRV CYC KCH NFY
*** *** *** *** *** *** *** ***
Press 1 to decrypt the message
Press 2 to print the encryption instructions of vigenere cipher
Press 3 to print the decryption instructions of vigenere cipher
Press 4 to store the encrypted text into the system
Press -1 to exit the function
*** *** *** *** *** *** *** ***
1
The encrypted text is: IVRRV CYC KCH NFY
The key used in encyption is: monarchy
The decrypted text using vigenere is: WHERE ARE YOU NOW
*** *** *** *** *** *** *** ***
```

Figure 14 Vigenère encryption execution

8.6 displaying the encrypted text that is stored in the system.

From figure 10 and figure 11 we choose to store the encrypted message to the system.

During the execution, the user chooses to display the encrypted text stored in the system. The system displays the encrypted messages, and the user selects one of them. The system then decrypts the selected message using the corresponding encryption method and prints it with the encryption method used.

```
*********************************************************
Press 1 to Encrypt using monoalphabetic subtitution
Press 2 to Encrypt using playfair
Press 3 to Encrypt using playfair and monoalphbetic cipher
Press 4 to Encrypt using Keyed trasposition
Press 5 to to Encrypt using keyed tranposition (bit-level)
Press 6 to Encrypt using vigenere cipher
Press 7 to Encrypt using DES
Press 8 to Decrypt using frequency analysis*
Press 9 to Display the encrypted texts
*******  Press -1 if u want to exit  *******
*********************************************************



9
1 The encryption message is: YJGTG CTG AQW PQY
2 The encryption message is: vykmimmkhnwmnv
Enter the numbber of message that you want to decrypt
2
1
*this message encrypted by playfair
The decryption text is: whereareyounow
```

Figure 15 The system can store the encrypted message and decrypt them.

Section 9: Limitations

One of the limitations encountered during this project was the of the Data Encryption Standard (DES) algorithm. This limitation starts due to time constraints and the complexity of the algorithm itself.

Section 10: Conclusion

In conclusion, the implementation of our educational system for encryption and decryption using different encryption algorithms has been a valuable learning experience. Throughout the development process, we have gained insights into the functionality, strengths, and limitations of various encryption algorithms. By working with different encryption algorithms, such as the Playfair cipher, monoalphabetic substitution, keyed transposition, Vigenère cipher, and DES.

References:

- **Java Program to Implement the Monoalphabetic Cypher.** Link
- **Java Program to Encode a Message Using Playfair Cipher.** Link
- **Java – Convert String to Binary.** Link
- **Vigenère Cipher.** Link