**DOP: / /2023** **DOS: / /2023**

## Experiment No: 03

**Aim**: Cryptanalysis or decoding Playfair, Vigenère cipher.

**Theory:**

### Vigenere Cipher:

**Vigenere Cipher** is an encryption and decryption algorithm. It is a type of polyalphabetic substitution cipher, which means that the cipher alphabet is changed regularly during the encryption process. Due to this, the cipher becomes less vulnerable to cryptanalysis.

The Vigenere Cipher was developed in 1585 by Blaise de Vigenere. He used a Vigenere table or square to encode messages

**Encryption:**
The plaintext(P) and key(K) are added modulo 26.
$E_i = (P_i + K_i) \bmod 26$

**Decryption:**
$D_i = (E_i - K_i + 26) \bmod 26$

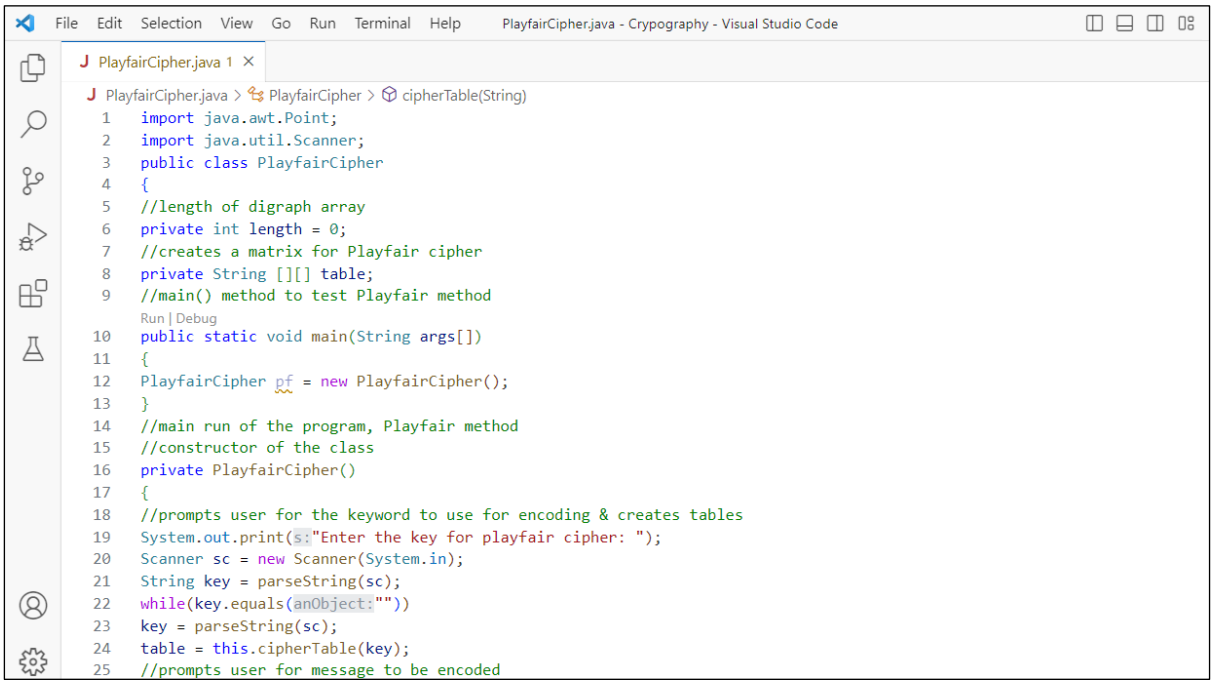
### Playfair Cipher:

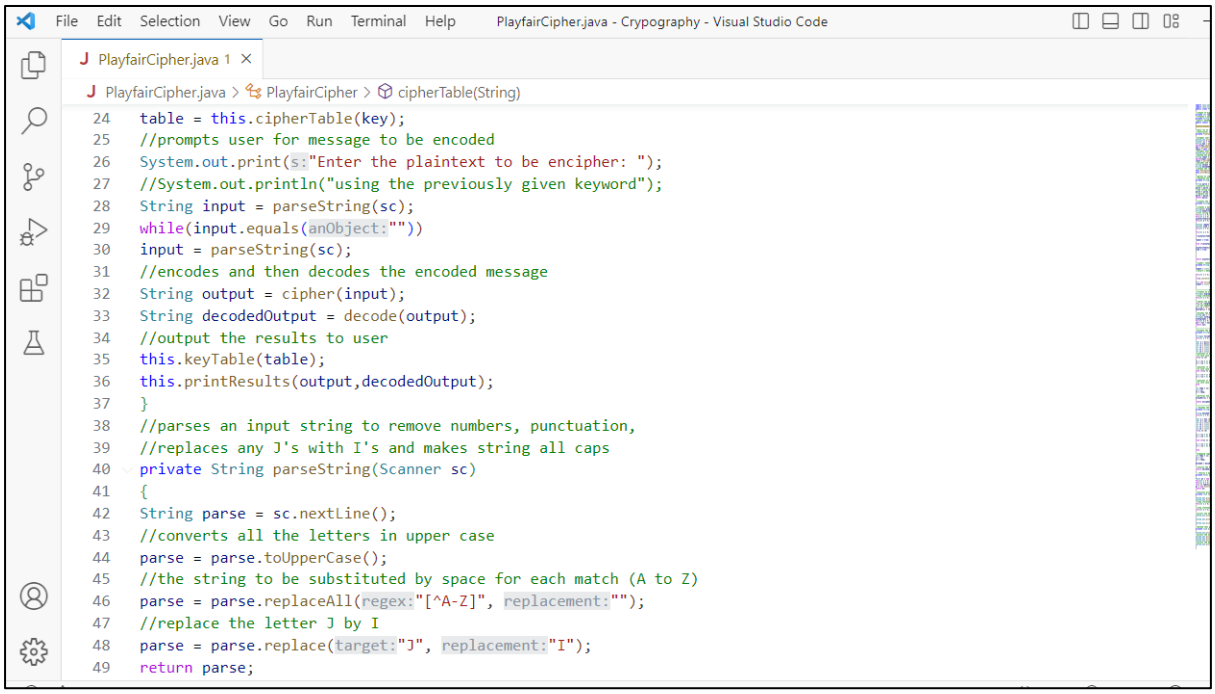
Playfair cipher is an encryption algorithm to encrypt or encode a message. It is the same as a traditional cipher. The only difference is that it encrypts a digraph (a pair of two letters) instead of a single letter.

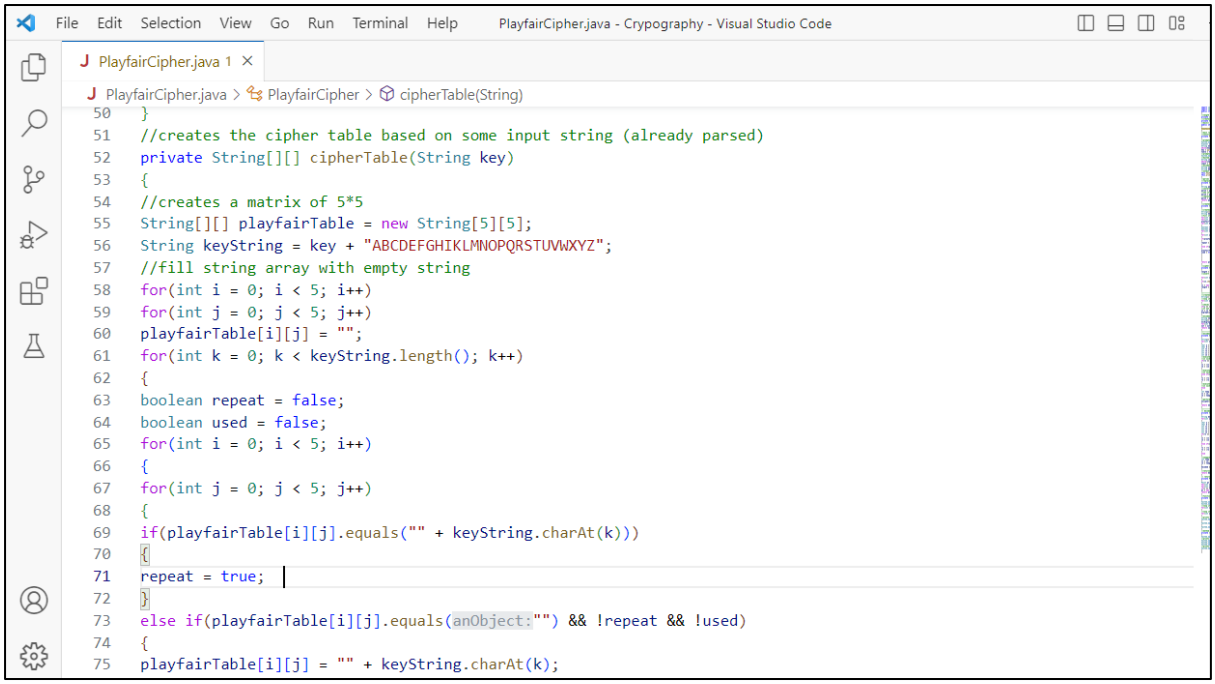
It initially creates a key-table of 5*5 matrix. The matrix contains alphabets that act as the key for encryption of the plaintext. Note that any alphabet should not be repeated. Another point to note that there are 26 alphabets and we have only 25 blocks to put a letter inside it. Therefore, one letter is excess so, a letter will be omitted (usually J) from the matrix. Nevertheless, the plaintext contains J, then J is replaced by I. It means treat I and J as the same letter, accordingly.

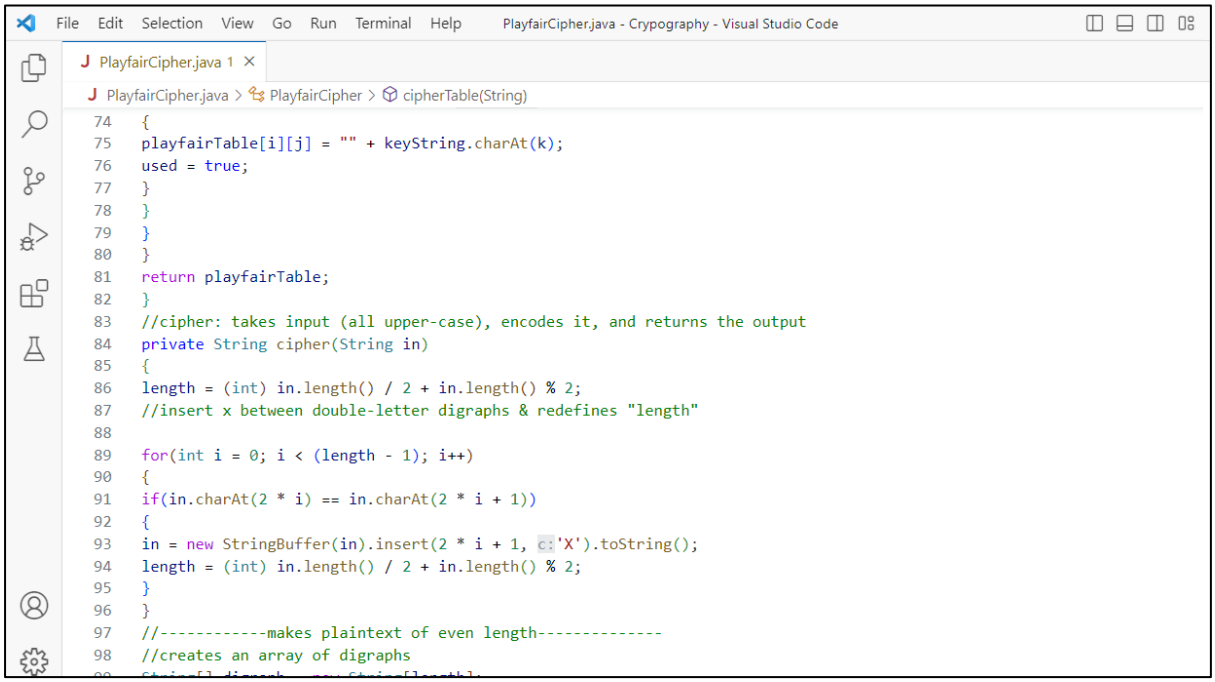
Since Playfair cipher encrypts the message digraph by digraph. Therefore, the Playfair cipher is an example of a digraph substitution cipher

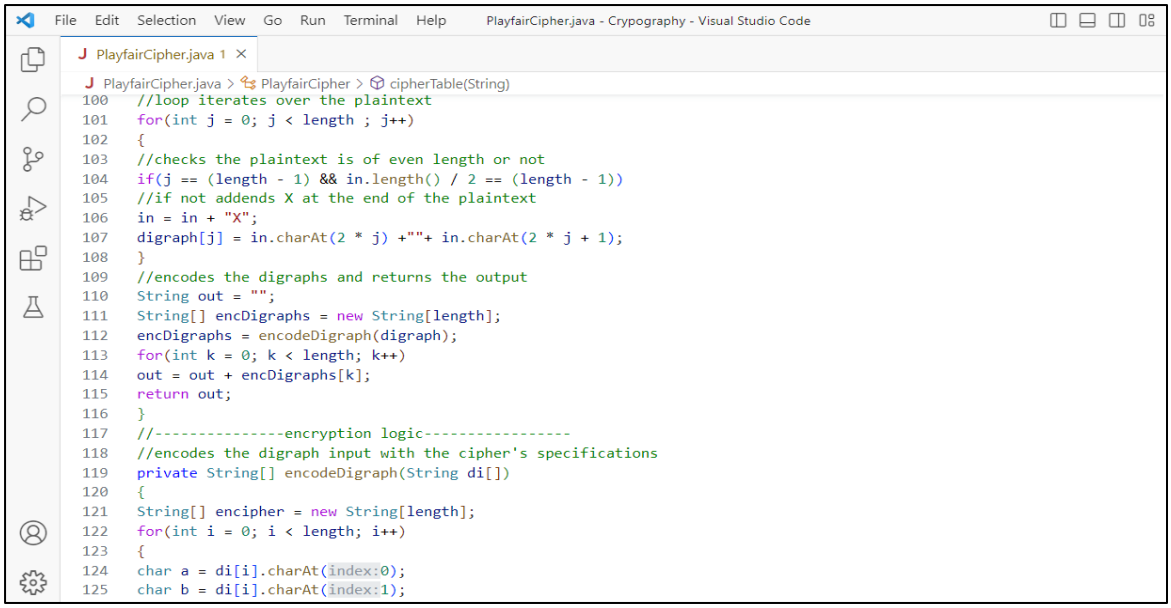
**Input:**

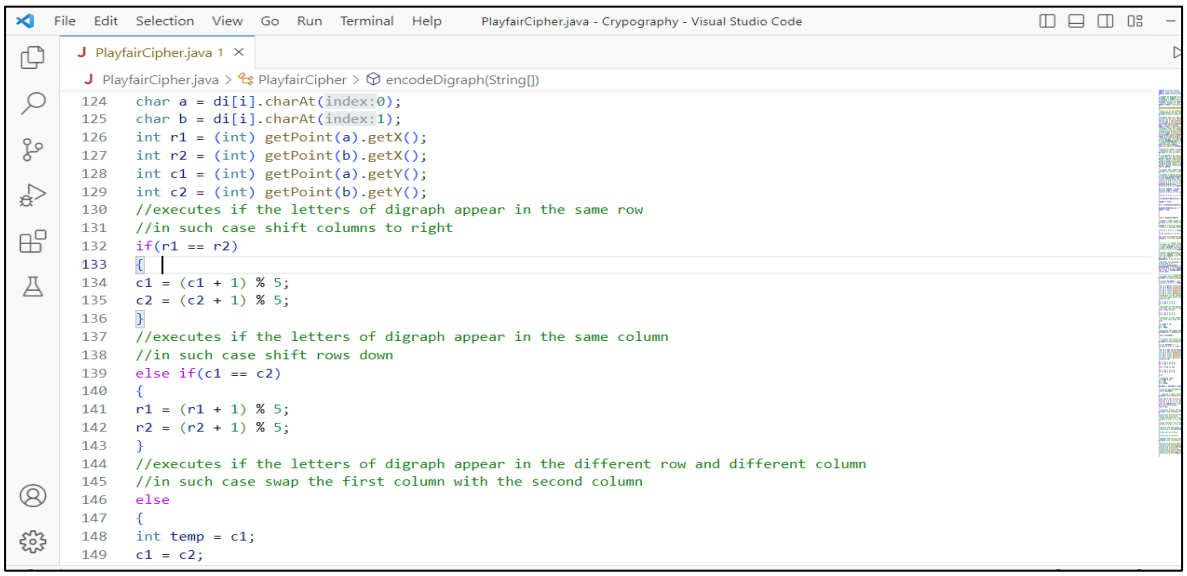

File Edit Selection View Go Run Terminal Help PlayfairCipher.java - Crypography - Visual Studio Code
PlayfairCipher.java 1
PlayfairCipher.java > PlayfairCipher > cipherTable(String)
1 import java.awt.Point;
2 import java.util.Scanner;
3 public class PlayfairCipher
4 {
5 //length of digraph array
6 private int length = 0;
7 //creates a matrix for Playfair cipher
8 private String [][] table;
9 //main() method to test Playfair method
Run | Debug
10 public static void main(String args[])
11 {
12 PlayfairCipher pf = new PlayfairCipher();
13 }
14 //main run of the program, Playfair method
15 //constructor of the class
16 private PlayfairCipher()
17 {
18 //prompts user for the keyword to use for encoding & creates tables
19 System.out.print(s:"Enter the key for playfair cipher: ");
20 Scanner sc = new Scanner(System.in);
21 String key = parseString(sc);
22 while(key.equals(anObject:""))
23 key = parseString(sc);
24 table = this.cipherTable(key);
25 //prompts user for message to be encoded

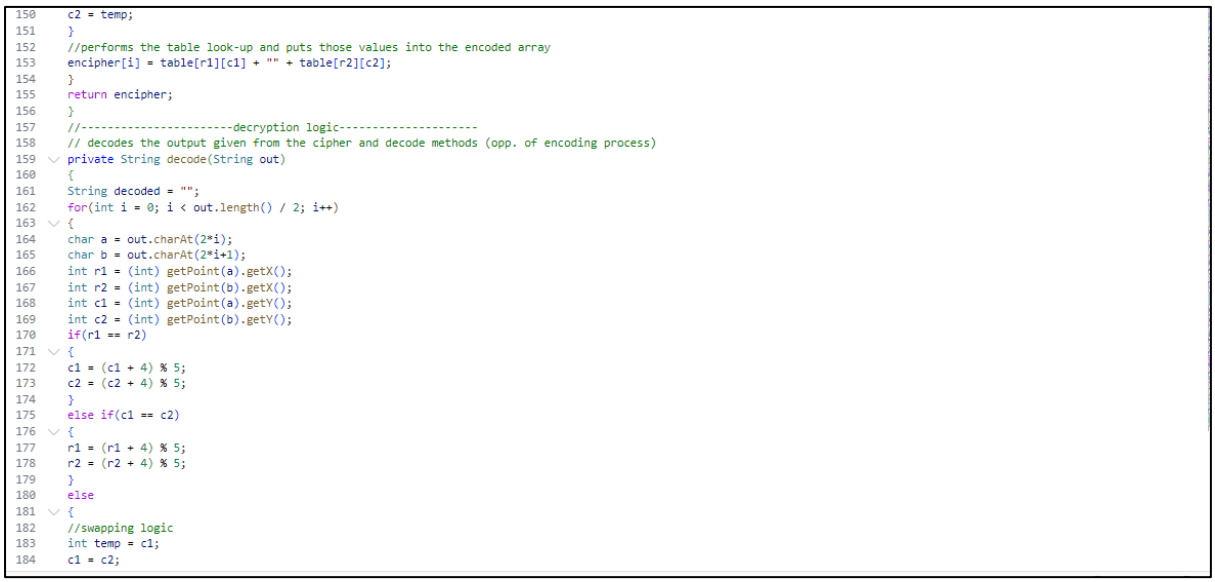

File Edit Selection View Go Run Terminal Help PlayfairCipher.java - Crypography - Visual Studio Code
PlayfairCipher.java 1
PlayfairCipher.java > PlayfairCipher > cipherTable(String)
24 table = this.cipherTable(key);
25 //prompts user for message to be encoded
26 System.out.print(s:"Enter the plaintext to be encipher: ");
27 //System.out.println("using the previously given keyword");
28 String input = parseString(sc);
29 while(input.equals(anObject:""))
30 input = parseString(sc);
31 //encodes and then decodes the encoded message
32 String output = cipher(input);
33 String decodedOutput = decode(output);
34 //output the results to user
35 this.keyTable(table);
36 this.printResults(output,decodedOutput);
37 }
38 //parses an input string to remove numbers, punctuation,
39 //replaces any J's with I's and makes string all caps
40 private String parseString(Scanner sc)
41 {
42 String parse = sc.nextLine();
43 //converts all the letters in upper case
44 parse = parse.toUpperCase();
45 //the string to be substituted by space for each match (A to Z)
46 parse = parse.replaceAll(regex:"[^A-Z]", replacement:"");
47 //replace the letter J by I
48 parse = parse.replace(target:"J", replacement:"I");
49 return parse;

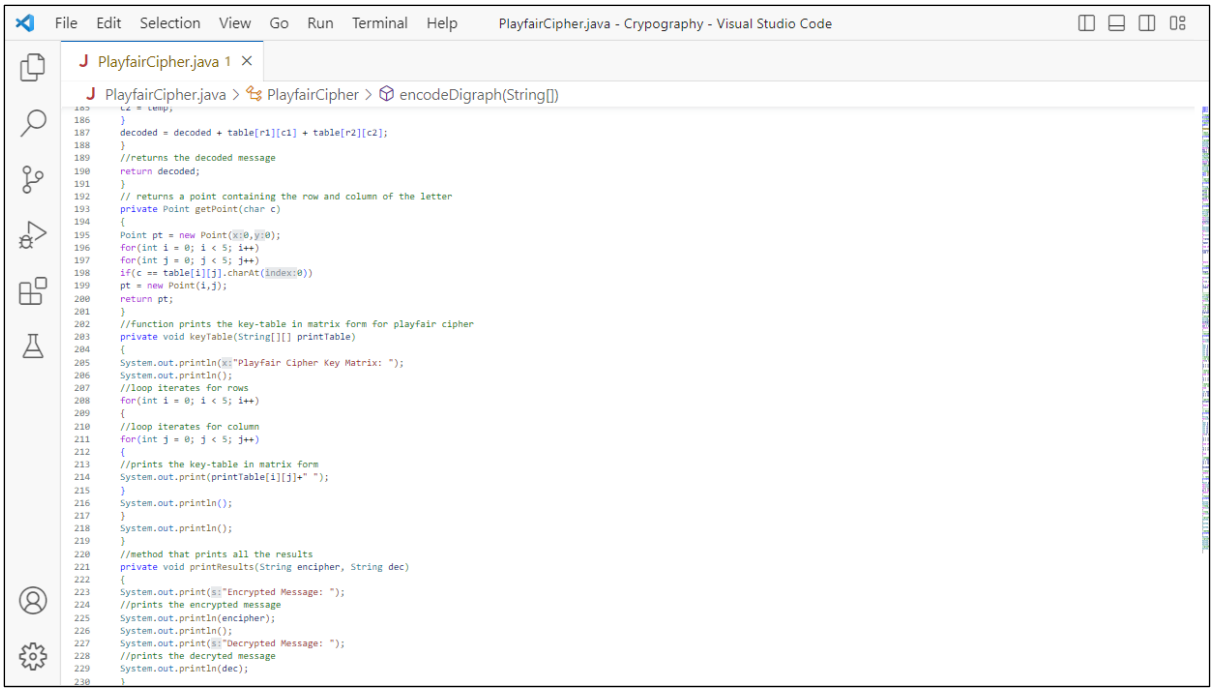

File Edit Selection View Go Run Terminal Help
PlayfairCipher.java - Crypography - Visual Studio Code
PlayfairCipher.java 1
PlayfairCipher.java > PlayfairCipher > cipherTable(String)
50 }
51 //creates the cipher table based on some input string (already parsed)
52 private String[][] cipherTable(String key)
53 {
54 //creates a matrix of 5*5
55 String[][] playfairTable = new String[5][5];
56 String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";
57 //fill string array with empty string
58 for(int i = 0; i < 5; i++)
59 for(int j = 0; j < 5; j++)
60 playfairTable[i][j] = "";
61 for(int k = 0; k < keyString.length(); k++)
62 {
63 boolean repeat = false;
64 boolean used = false;
65 for(int i = 0; i < 5; i++)
66 {
67 for(int j = 0; j < 5; j++)
68 {
69 if(playfairTable[i][j].equals("" + keyString.charAt(k)))
70 {
71 repeat = true;
72 }
73 else if(playfairTable[i][j].equals(anObject:"") && !repeat && !used)
74 {
75 playfairTable[i][j] = "" + keyString.charAt(k);


File Edit Selection View Go Run Terminal Help
PlayfairCipher.java - Crypography - Visual Studio Code
PlayfairCipher.java 1
PlayfairCipher.java > PlayfairCipher > cipherTable(String)
74 {
75 playfairTable[i][j] = "" + keyString.charAt(k);
76 used = true;
77 }
78 }
79 }
80 }
81 return playfairTable;
82 }
83 //cipher: takes input (all upper-case), encodes it, and returns the output
84 private String cipher(String in)
85 {
86 length = (int) in.length() / 2 + in.length() % 2;
87 //insert x between double-letter digraphs & redefines "length"
88
89 for(int i = 0; i < (length - 1); i++)
90 {
91 if(in.charAt(2 * i) == in.charAt(2 * i + 1))
92 {
93 in = new StringBuffer(in).insert(2 * i + 1, c:'X').toString();
94 length = (int) in.length() / 2 + in.length() % 2;
95 }
96 }
97 //------------makes plaintext of even length--------------
98 //creates an array of digraphs


File Edit Selection View Go Run Terminal Help PlayfairCipher.java - Crypography - Visual Studio Code
PlayfairCipher.java 1
PlayfairCipher.java > PlayfairCipher > cipherTable(String)
//loop iterates over the plaintext
for(int j = 0; j < length ; j++)
{
//checks the plaintext is of even length or not
if(j == (length - 1) && in.length() / 2 == (length - 1))
//if not addends X at the end of the plaintext
in = in + "X";
digraph[j] = in.charAt(2 * j) +""+ in.charAt(2 * j + 1);
}
//encodes the digraphs and returns the output
String out = "";
String[] encDigraphs = new String[length];
encDigraphs = encodeDigraph(digraph);
for(int k = 0; k < length; k++)
out = out + encDigraphs[k];
return out;
}
//---------------encryption logic-----------------
//encodes the digraph input with the cipher's specifications
private String[] encodeDigraph(String di[])
{
String[] encipher = new String[length];
for(int i = 0; i < length; i++)
{
char a = di[i].charAt(index:0);
char b = di[i].charAt(index:1);


File Edit Selection View Go Run Terminal Help PlayfairCipher.java - Crypography - Visual Studio Code
PlayfairCipher.java 1
PlayfairCipher.java > PlayfairCipher > encodeDigraph(String[])
char a = di[i].charAt(index:0);
char b = di[i].charAt(index:1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
//executes if the letters of digraph appear in the same row
//in such case shift columns to right
if(r1 == r2)
{
c1 = (c1 + 1) % 5;
c2 = (c2 + 1) % 5;
}
//executes if the letters of digraph appear in the same column
//in such case shift rows down
else if(c1 == c2)
{
r1 = (r1 + 1) % 5;
r2 = (r2 + 1) % 5;
}
//executes if the letters of digraph appear in the different row and different column
//in such case swap the first column with the second column
else
{
int temp = c1;
c1 = c2;


c2 = temp;
}
//performs the table look-up and puts those values into the encoded array
encipher[i] = table[r1][c1] + "" + table[r2][c2];
}
return encipher;
}
//----------------------decryption logic--------------------
// decodes the output given from the cipher and decode methods (opp. of encoding process)
private String decode(String out)
{
String decoded = "";
for(int i = 0; i < out.length() / 2; i++)
{
char a = out.charAt(2*i);
char b = out.charAt(2*i+1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
if(r1 == r2)
{
c1 = (c1 + 4) % 5;
c2 = (c2 + 4) % 5;
}
else if(c1 == c2)
{
r1 = (r1 + 4) % 5;
r2 = (r2 + 4) % 5;
}
else
{
//swapping logic
int temp = c1;
c1 = c2;




```
    }
    decoded = decoded + table[r1][c1] + table[r2][c2];
    }
    //returns the decoded message
    return decoded;
    }
    // returns a point containing the row and column of the letter
    private Point getPoint(char c)
    {
    Point pt = new Point(x:0,y:0);
    for(int i = 0; i < 5; i++)
    for(int j = 0; j < 5; j++)
    if(c == table[i][j].charAt(index:0))
    pt = new Point(i,j);
    return pt;
    }
    //function prints the key-table in matrix form for playfair cipher
    private void keyTable(String[][] printTable)
    {
    System.out.println(x:"Playfair Cipher Key Matrix: ");
    System.out.println();
    //loop iterates for rows
    for(int i = 0; i < 5; i++)
    {
    //loop iterates for column
    for(int j = 0; j < 5; j++)
    {
    //prints the key-table in matrix form
    System.out.print(printTable[i][j]+" ");
    }
    System.out.println();
    }
    System.out.println();
    }
    //method that prints all the results
    private void printResults(String encipher, String dec)
    {
    System.out.print(s:"Encrypted Message: ");
    //prints the encrypted message
    System.out.println(encipher);
    System.out.println();
    System.out.print(s:"Decrypted Message: ");
    //prints the decrypted message
    System.out.println(dec);
    }
```

## Output:

```
4 errors
PS C:\Users\priyush\Desktop\Crypography> cd "c:\Users\priyush\Desktop\Crypography\" ; if ($?) { javac PlayfairCiph
java } ; if ($?) { java PlayfairCipher }
Enter the key for playfair cipher: PRIYUSH
Enter the plaintext to be encipher: NUTAN
Playfair Cipher Key Matrix:

P R I Y U
S H A B C
D E F G K
L M N O Q
T V W X Z

Encrypted Message: QIWSOW

Decrypted Message: NUTANX
PS C:\Users\priyush\Desktop\Crypography>
```

## Conclusion: -

Thus, we have done Cryptanalysis or decoding Playfair, Vigenère cipher