**Ex.No:1 IMPLEMENTATION OF CAESAR CIPHER**

**Date:**

**AIM:**

Write a Java program to perform encryption and decryption using Caesar cipher.

**ALGORITHM:**

**STEP1:**Read the plaintext and Replace the letters of plaintext by other letters

**STEP 2 :** Caesar cipher involves replacing each letter of the alphabet with the letter standing 3

places further down the alphabet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

**STEP3:** The formula used for encrypting the message in Caesar cipher such that

C = E(p) = (p+3) mod 26

shift may be any amount, so that general Caesar algorithm is

C = E (p) = (p+k) mod 26 Where k takes on a value in the range 1 to 25.

**STEP 4 :**The formula used for decrypting the Caesar cipher is P = D(C) = (C-3) mod 26

**STEP 5:** Display the ciphertext

**PROGRAM:**

**import** java.util.Scanner;

**public** **class** CaesarCipher

{

**public** **static** **final** String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

**public** **static** String encrypt(String plainText, **int** shiftKey)

{

plainText = plainText.toLowerCase();

String cipherText = "";

**for** (**int** i = 0; i < plainText.length(); i++)

{

**int** charPosition = ALPHABET.indexOf(plainText.charAt(i));

**int** keyVal = (shiftKey + charPosition) % 26;

**char** replaceVal = ALPHABET.charAt(keyVal);

cipherText += replaceVal;

}

**return** cipherText;

}

**public** **static** String decrypt(String cipherText, **int** shiftKey)

{

cipherText = cipherText.toLowerCase();

String plainText = "";

**for** (**int** i = 0; i < cipherText.length(); i++)

{

**int** charPosition = ALPHABET.indexOf(cipherText.charAt(i));

**int** keyVal = (charPosition - shiftKey) % 26;

**if** (keyVal < 0)

{

keyVal = ALPHABET.length() + keyVal;

}

**char** replaceVal = ALPHABET.charAt(keyVal);

plainText += replaceVal;

}

**return** plainText;

}

**public** **static** **void** main(String[] args)

{

Scanner sc = **new** Scanner(System.in);

System.out.println("Enter the String for Encryption: ");

String message = **new** String();

message = sc.next();

System.out.println(encrypt(message, 3));

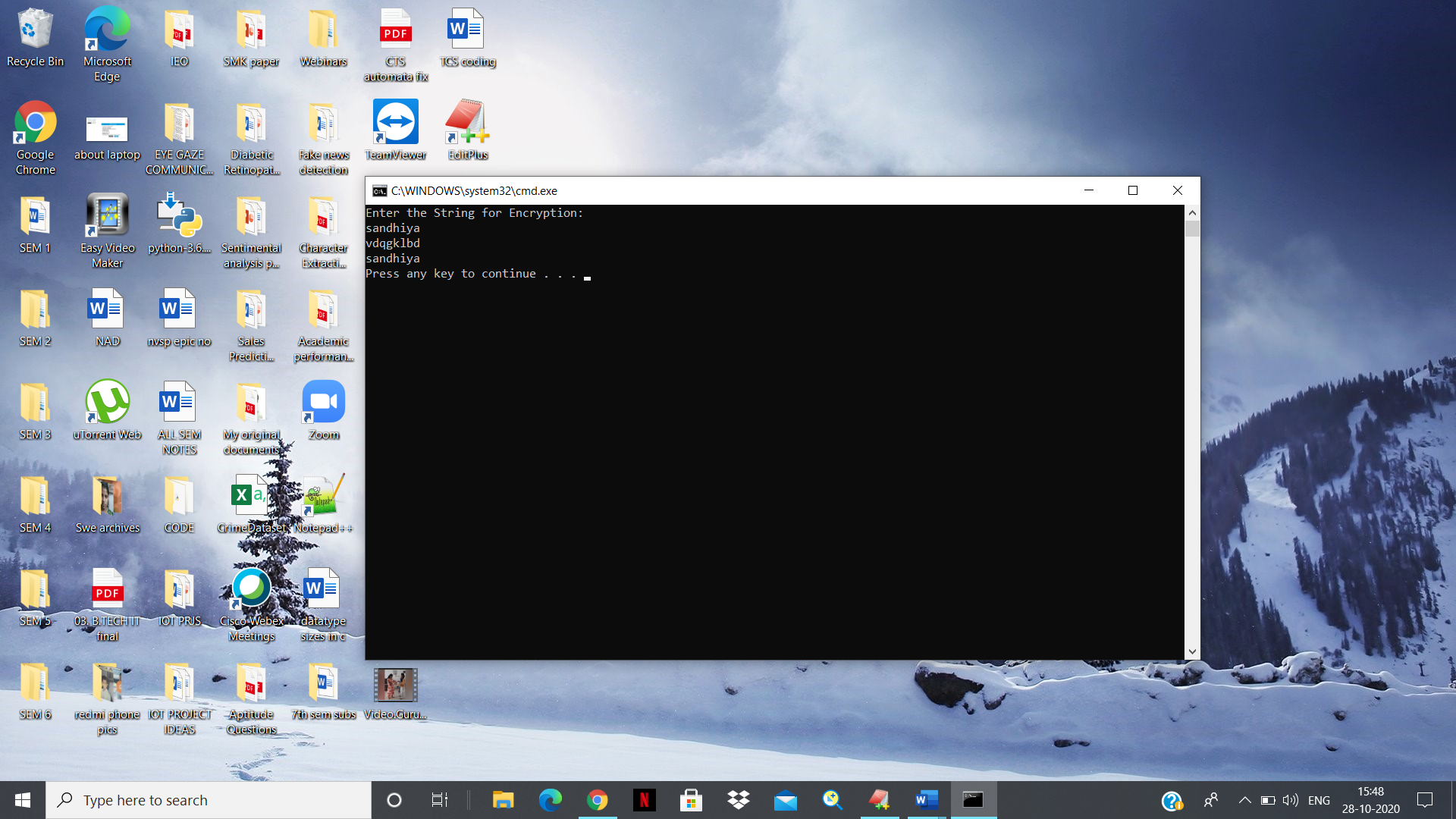
System.out.println(decrypt(encrypt(message, 3), 3));

sc.close();

}

}

**OUTPUT:**



**RESULT:**

The Caesar cipher was implemented using Java.

**Ex.No:2 IMPLEMENTATION OF PLAYFAIR CIPHER**

**Date:**

**AIM:**

Write a Java program to encrypt the given plaintext to Playfair cipher text using keyword “MONARCHY.

**ALGORITHM:**

**STEP 1:** Construct a Playfair cipher which is based on 5 X 5 matrix containing a key word or phrase.

**STEP 2:** To Generate the key matrix, fill in the spaces in the matrix with the letters of the keyword (dropping any duplicate letters), then fill the remaining spaces with the rest of the letters of the alphabet in order (put both "I" and "J" in the same space) to reduce the alphabet to fit.. The key can be written in the top rows of the table, from left to right .

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

**STEP 3:** To encrypt a message, first break the message into groups of 2 letters such that, for example, "HelloWorld" becomes "HE LL OW OR LD", and map them out on the key matrix. If needed, append an uncommon monogram to complete the final diagram.

**STEP4:**Encrypt two letters of plaintext at a time and Apply the following 4 rules, in order, to each pair of letters in the plaintext:

**STEP 4.1:** If both letters are the same ,they are separated by filter letter such as x.For eg, the plaintext balloon would be treated as ba lx lo on.

**STEP4.2**:Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right,with the first element of the row circularly following the last.For example, ar is encryptede as RM.

**STEP 4.3:**If two plaintext letters that fall in the same column are each replaced by the letter beneath,with the top element of the column circularly following the last .For example ,mu is encrypted as CM.

**STEP 4.4:** If the letters are not on the same row or column, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintextletter.Thus ,hs becomes BP and ea becomes IM.

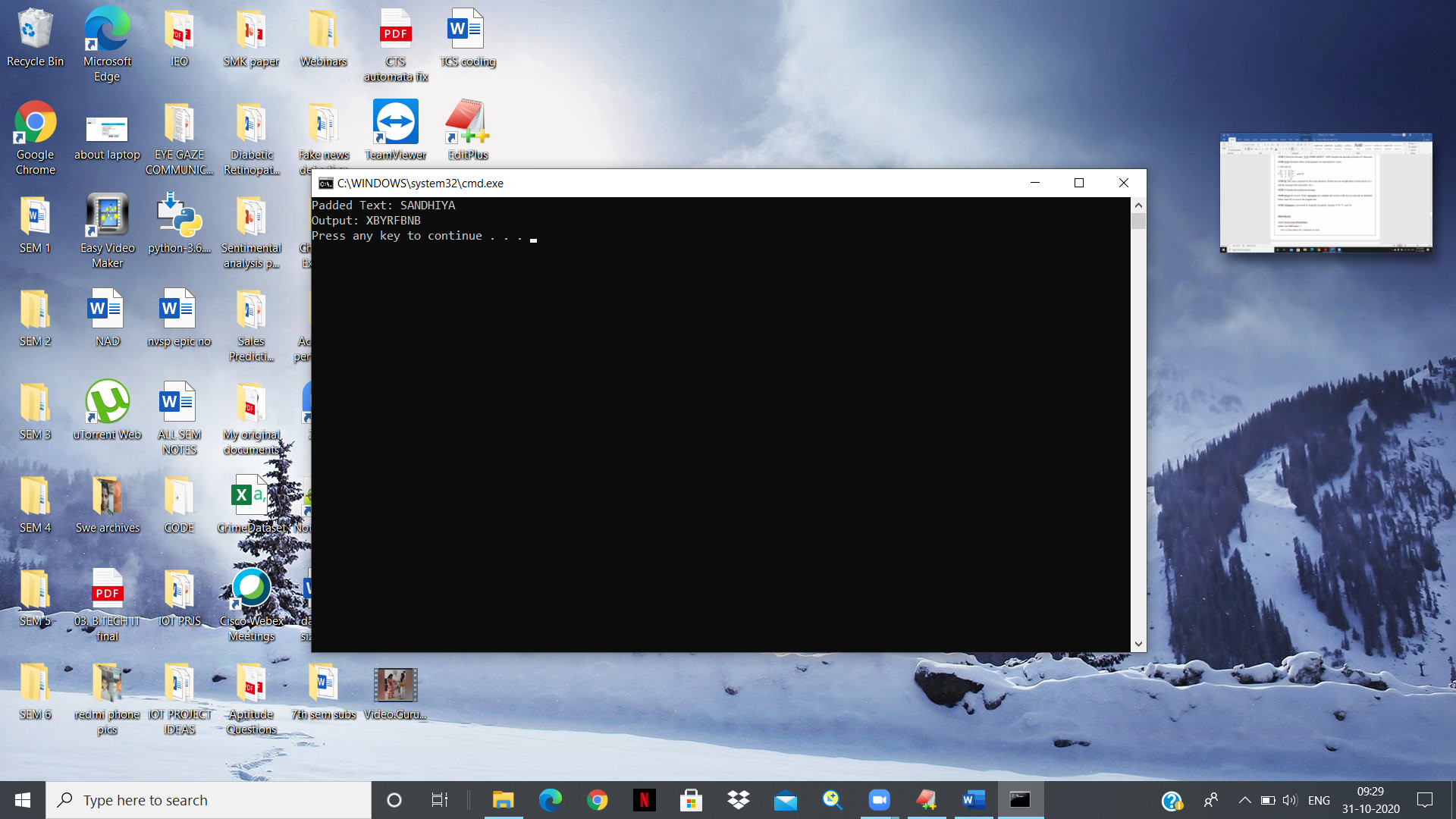
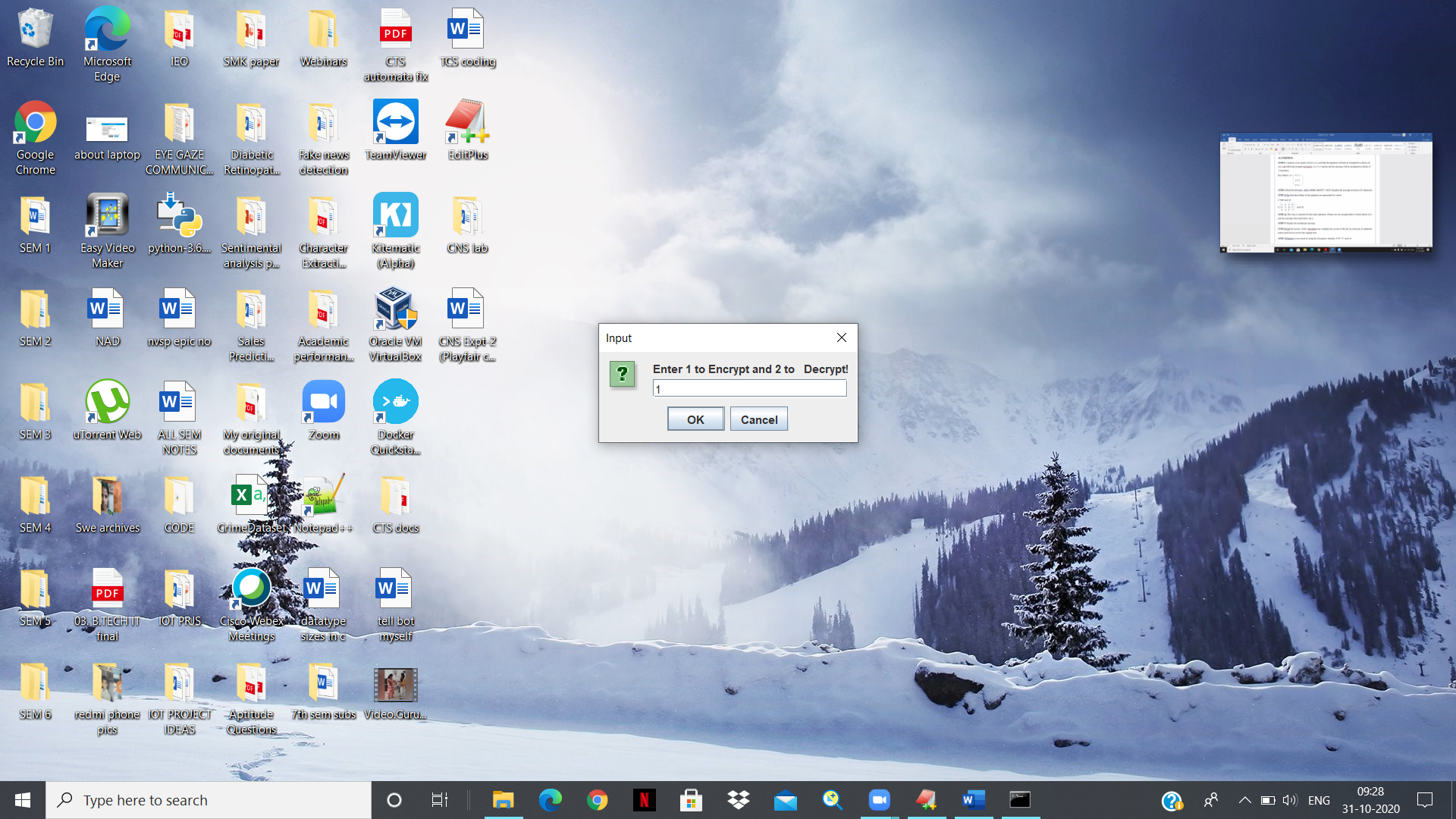
**STEP 5:** Display the ciphertext produced from the plaintext.

**PROGRAM:**

import javax.swing.JOptionPane;  
  
public class PlayFair {  
    //the key matrix  
    public static char[][] keymat = new char[][]{  
                { '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 static String trans = "J"; //for translating J to I  
    public static char filler = 'X'; //filler letter is X  
    public static void main(String args[])

{  
        String text, outtext="";  
        int ch;  
        ch = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter 1 to Encrypt and 2 to Decrypt!"));  
        text = JOptionPane.showInputDialog(null, "Enter plain/cipher text to encrypt/decrypt?");  
        text = text.toUpperCase();  
        text = text.replaceAll("\\s",""); //removing spaces  
        text = text.replace(trans, "I"); //changing J with I  
        text = text.replaceAll("([A-Z])\\1+","$1"+filler+"$1"); //handling repeated letters  
        if(text.length() % 2 !=0)  
            text+= filler;  
        char[] ptextchars = text.toCharArray();  
        System.out.println("Padded Text: "+text);  
                switch(ch){  
            case 1:  
                    for(int i=0;i< text.length(); i+=2){  
                        outtext += encrypt(ptextchars[i],ptextchars[i+1]);  
                    }  
                    break;  
            case 2:  
                    for(int i=0;i< text.length(); i+=2){  
                        outtext += decrypt(ptextchars[i],ptextchars[i+1]);  
                    }  
                    break;  
            default: System.out.println("Invalid Choice!");  
        }  
        System.out.println("Output: "+outtext);  
    }  
  
    private static String encrypt(char c1, char c2) {  
        int[] posa = new int[2];  
        int[] posb = new int[2];  
        String frag = "";  
        posa = search(c1);  
        posb = search(c2);  
        if(posa[0] == posb[0]){//same row  
            c1 = keymat[posa[0]][(posa[1]+1)%5];  
            c2 = keymat[posb[0]][(posb[1]+1)%5];  
            frag = (""+c1+c2+"");  
        }  
        else if(posa[1] == posb[1]){ //same column  
            c1 = keymat[(posa[0]+1)%5][posa[1]];  
            c2 = keymat[(posb[0]+1)%5][posb[1]];  
            frag = (""+c1+c2+"");  
        }  
        else{  
            c1 = keymat[posb[0]][posa[1]];  
            c2 = keymat[posa[0]][posb[1]];  
            frag = (""+c1+c2+"");  
        }  
        return frag;  
    }  
  
    private static String decrypt(char c1, char c2) {  
        int[] posa = new int[2];  
        int[] posb = new int[2];  
        String frag = "";  
        posa = search(c1);  
        posb = search(c2);  
        if(posa[0] == posb[0]){//same row  
            c1 = keymat[posa[0]][(posa[1]-1)%5];  
            c2 = keymat[posb[0]][(posb[1]-1)%5];  
            frag = (""+c1+c2+"");  
        }  
        else if(posa[1] == posb[1]){ //same column  
            c1 = keymat[(posa[0]-1)%5][posa[1]];  
            c2 = keymat[(posb[0]-1)%5][posb[1]];  
            frag = (""+c1+c2+"");  
        }  
        else{  
            c1 = keymat[posb[0]][posa[1]];  
            c2 = keymat[posa[0]][posb[1]];  
            frag = (""+c1+c2+"");  
        }  
        return frag;  
    }  
  
    private static int[] search(char c) {  
        int i,j;  
        int[] pos = new int[2];  
        for (i = 0; i < 5; i++) {  
            for (j = 0; j < 5; j++) {  
                if (keymat[i][j] == c){  
                    pos[0] = i;  
                    pos[1] = j;  
                    break;  
                }  
            }  
        }  
        return pos;  
    }  
   
}

**OUTPUT:**



**RESULT:**

The Playfair cipher was implemented using Java.

**Ex.No:3 IMPLEMENTATION OF HILL CIPHER**

**Date:**

**AIM:**

Write a Java program to perform encryption and decryption using Hill Cipher.

**ALGORITHM:**

**STEP1: Construct a key matrix of size n x n such that the** plaintext will then be enciphered in blocks of size *n*.In following example keymatrix  is a 3 x 3 matrix and the message will be enciphered in blocks of 3 characters.

Key Matrix: A =  1 2 1

2 3 2

2 2 1

**STEP 2:** Read the Message “ pAY MORE MONEY” and Encipher the message in blocks of 3 characters

**STEP 3:**The first three letters of the plaintext are represented by vector

C=KP mod 26

C = mod 26

**STEP 4:.** This step is repeated for the entire plaintext. If there are not enough letters to form blocks of 3, pad the message with some letter, say x.

**STEP 5:** Display the enciphered message.

**STEP 6**:Find the inverse of the keymatrix and multiply the inverse of the key by each pair of ciphertext letters (mod 26) to recover the original text.

**STEP 7:**Plaintext is recovered by using the decryption formula.. P=C mod 26

**PROGRAM:**

import javax.swing.JOptionPane;

public class HillCipher1 {

//the 3x3 key matrix for 3 characters at once

public static int[][] keymat = new int[][]

{

{ 1, 2, 1 },

{ 2, 3, 2 },

{ 2, 2, 1 },

};

public static int[][] invkeymat = new int[][]

{

{ -1, 0, 1 },

{ 2, -1, 0 },

{ -2, 2, -1},

};

public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

public static void main(String[] args)

{

String text,outtext ="";

int ch, n;

ch = Integer.parseInt(JOptionPane.showInputDialog(null, "Enter 1 to Encrypt and 2 to Decrypt!"));

text = JOptionPane.showInputDialog(null, "Enter plain/cipher text to encrypt?");

text = text.toUpperCase();

text = text.replaceAll("\\s",""); //removing spaces

n = text.length() % 3;

if(n!=0){

for(int i = 1; i<= (3-n);i++){

text+= 'X';

}

}

System.out.println("Padded Text:" + text);

char[] ptextchars = text.toCharArray();

switch(ch)

{

case 1:

for(int i=0;i< text.length(); i+=3){

outtext += encrypt(ptextchars[i],ptextchars[i+1],ptextchars[i+2]);

}

break;

case 2:

for(int i=0;i< text.length(); i+=3){

outtext += decrypt(ptextchars[i],ptextchars[i+1],ptextchars[i+2]);

}

break;

default: System.out.println("Invalid Choice!");

}

System.out.println("Output: " + outtext);

}

private static String encrypt(char a, char b, char c)

{

String ret = "";

int x,y, z;

int posa = (int)a - 65;

int posb = (int)b - 65;

int posc = (int)c - 65;

x = posa \* keymat[0][0] + posb \* keymat[1][0] + posc \* keymat[2][0];

y = posa \* keymat[0][1] + posb \* keymat[1][1] + posc \* keymat[2][1];

z = posa \* keymat[0][2] + posb \* keymat[1][2] + posc \* keymat[2][2];

a = key.charAt(x%26);

b = key.charAt(y%26);

c = key.charAt(z%26);

ret = "" + a + b + c;

return ret; }

private static String decrypt(char a, char b, char c)

{

String ret = "";

int x,y,z;

int posa = (int)a - 65;

int posb = (int)b - 65;

int posc = (int)c - 65;

x = posa \* invkeymat[0][0]+ posb \* invkeymat[1][0] + posc \* invkeymat[2][0];

y = posa \* invkeymat[0][1]+ posb \* invkeymat[1][1] + posc \* invkeymat[2][1];

z = posa \* invkeymat[0][2]+ posb \* invkeymat[1][2] + posc \* invkeymat[2][2];

a = key.charAt((x%26<0)?(26+x%26):(x%26));

b = key.charAt((y%26<0)?(26+y%26):(y%26));

c = key.charAt((z%26<0)?(26+z%26):(z%26));

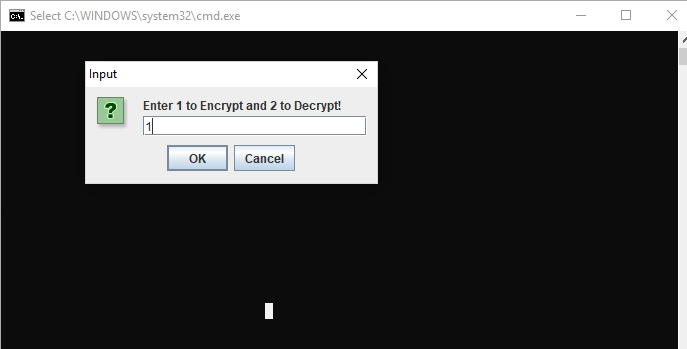
ret = "" + a + b + c;

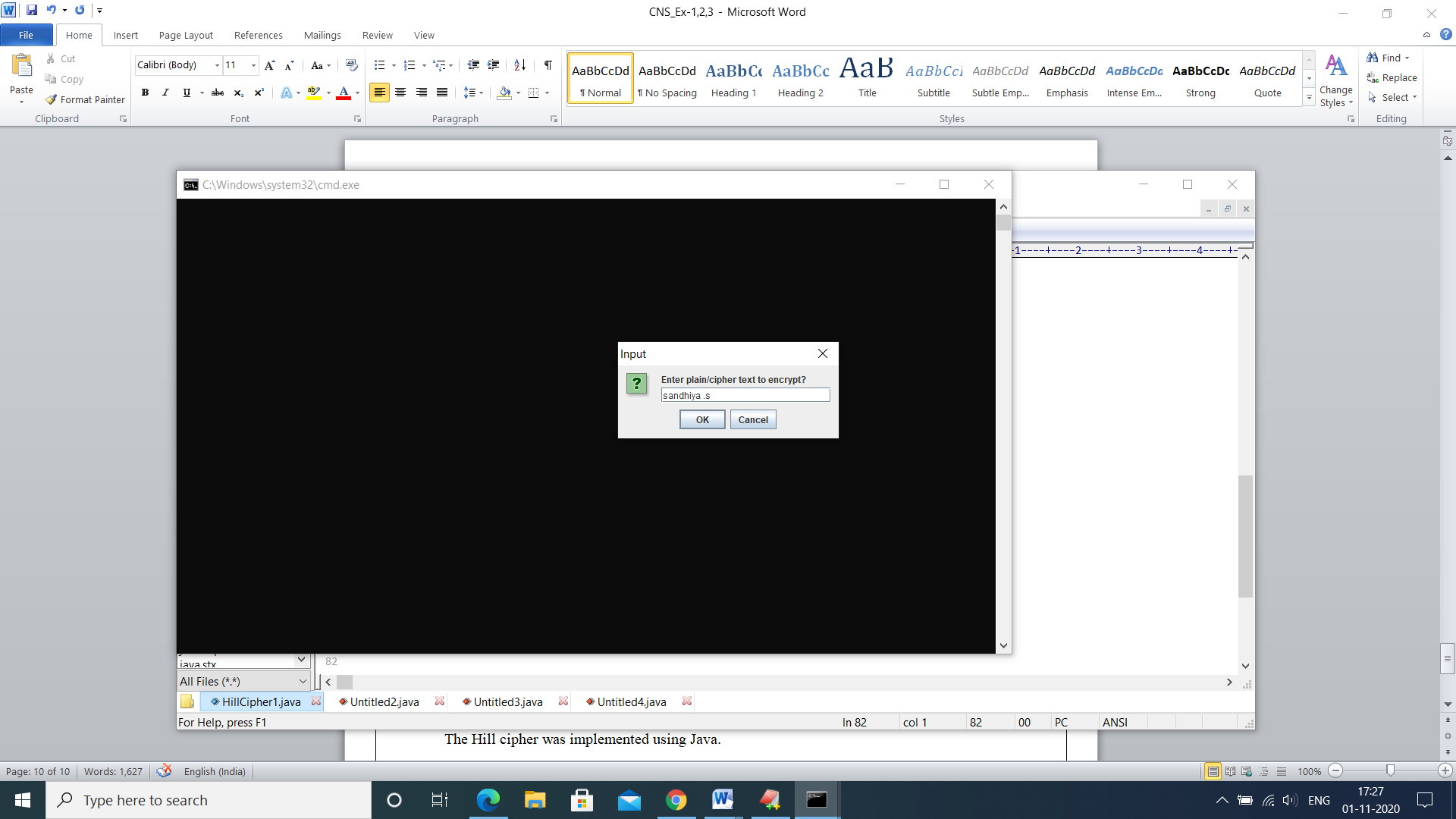
return ret;

}

}

**OUTPUT:**







**RESULT:**

The Hill cipher was implemented using Java.