**Substitution Cipher**

1. **Caesar Cipher:**

**Program:**

//import java.system.\*;

import java.util.\*;

public class Caesar\_Cipher{

void shift\_mod(String str,int key,int a){

String text = "abcdefghijklmnopqrstuvwxyz";

int len,i=0,j,k=0,val=0;

len = str.length();

char[] e =new char[len+1];

char[] d =new char[len+1];

while(i<len)

{

j=text.indexOf(str.charAt(i));

val = (j+key)%26;

e[k] = text.charAt(val);

i++;

k++;

}

String en,de;

for(i=len-1;i>=a;i--)

{

e[i+1]=e[i];

}

e[a]=' ';

en = String.valueOf(e);

System.out.println("Encrypted Word:"+en);

i=0;k=0;j=0;

//String str1 = String.valueOf(e);

String str1 = String.valueOf(e);

String st;

st = str1.replaceAll("\\s", "");

st = st.toLowerCase();

while(i<len)

{

j=text.indexOf(st.charAt(i));

val = (j-key)%26;

if(val<0)

{

val = val\*(-1);

val = 26-val;

}

d[k] = text.charAt(val);

i++;

k++;

}

for(i=len-1;i>=a;i--)

{

d[i+1]=d[i];

}

d[a]=' ';

de = String.valueOf(d);

System.out.println("Decrypted Word:"+de);

}

void shift(String str,int key,int a){

String text = "abcdefghijklmnopqrstuvwxyz";

int len,i=0,j,k=0,val=0;

len = str.length();

char[] e =new char[len+1];

char[] d =new char[len+1];

while(i<len)

{

j=text.indexOf(str.charAt(i));

val = (j+key);

if(val>26)

{

val = val-26;

}

e[k] = text.charAt(val);

i++;

k++;

}

String en,de;

for(i=len-1;i>=a;i--)

{

e[i+1]=e[i];

}

e[a]=' ';

en = String.valueOf(e);

System.out.println("Encrypted Word:"+en);

i=0;k=0;j=0;

String str1 = String.valueOf(e);

String st;

st = str1.replaceAll("\\s", "");

st = st.toLowerCase();

while(i<len)

{

j=text.indexOf(st.charAt(i));

val = (j-key);

if(val<0)

{

val = val\*(-1);

val = 26-val;

}

d[k] = text.charAt(val);

i++;

k++;

}

for(i=len-1;i>=a;i--)

{

d[i+1]=d[i];

}

d[a]=' ';

de = String.valueOf(d);

System.out.println("Decrypted Word:"+de);

}

void thank(){

System.out.println("ThankYou!!!");

System.exit(0);

}

public static void main(String[] args) {

String s,str;

int key,i;

Caesar\_Cipher cc = new Caesar\_Cipher();

char x=' ';

int ch,op;

do{

System.out.println("Enter The String:");

Scanner scan = new Scanner(System.in);

s=scan.nextLine();

i = s.indexOf(" ");

str = s.replaceAll("\\s", "");

str = str.toLowerCase();

//System.out.println(i);

//System.out.println(str);

System.out.println("Enter The Key:");

key=scan.nextInt();

System.out.println("Enter");

System.out.println("1.Shifting");

System.out.println("2.Shifting Using MOD");

ch=scan.nextInt();

switch(ch){

case 1: cc.shift(str,key,i);

break;

case 2: cc.shift\_mod(str,key,i);

break;

case 3: cc.thank();

break;

default:

System.out.println("Invalid Choice");

}

System.out.println("Enter -1 to Exit");

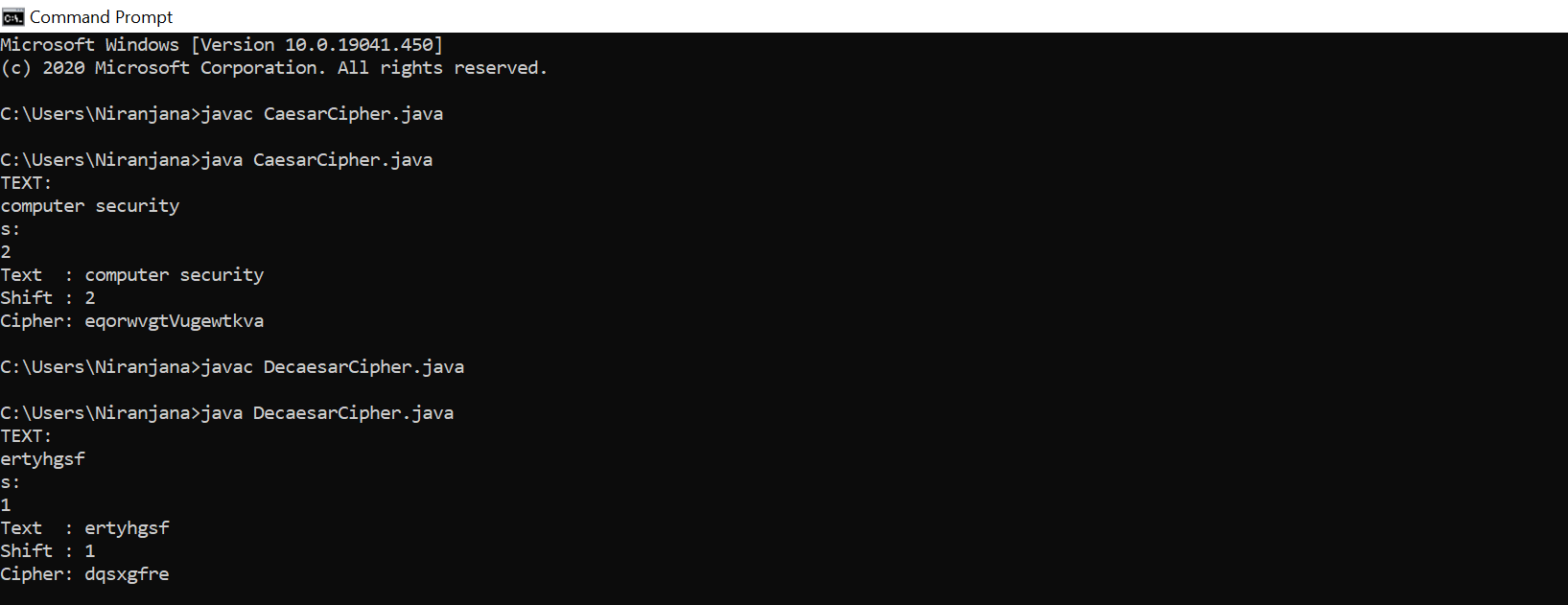
op = scan.nextInt();

}while(op!=-1);

}

}

**Screenshot:**



1. **Vigener:**

**Program:**

import java.util.\*;

public class vigener{

public static void main(String[] args){

String text = "abcdefghijklmnopqrstuvwxyz";

int len,len1,i=0,j=0,x=0,k=0,val=0,in=0;

String s,str,s1,str1,en,de;

int ch,op;

System.out.println("Enter The String:");

Scanner scan = new Scanner(System.in);

s=scan.nextLine();

str = s.replaceAll("\\s", "");

str = str.toLowerCase();

System.out.println("Enter The Key Stream:");

s1=scan.nextLine();

str1 = s1.replaceAll("\\s", "");

str1 = str1.toLowerCase();

len = str.length();

len1 = str1.length();

char[] e =new char[len];

char[] d =new char[len];

while(i<len)

{

if(in<len1)

{

j=text.indexOf(str.charAt(i));

x=text.indexOf(str1.charAt(in));

val = j+x;

if(val>=26)

val=val-26;

//System.out.println(val);

e[k] = text.charAt(val);

i++;

k++;

in++;

}

else

in=0;

}

en = String.valueOf(e);

System.out.println("Encrypted Word:"+en);

i=0;k=0;j=0;x=0;val=0;in=0;

String str2 = String.valueOf(e);

while(i<len)

{

if(in<len1)

{

j=text.indexOf(str2.charAt(i));

x=text.indexOf(str1.charAt(in));

val = j-x;

if(val<0)

val=26+val;

d[k] = text.charAt(val);

i++;

k++;

in++;

}

else

in=0;

}

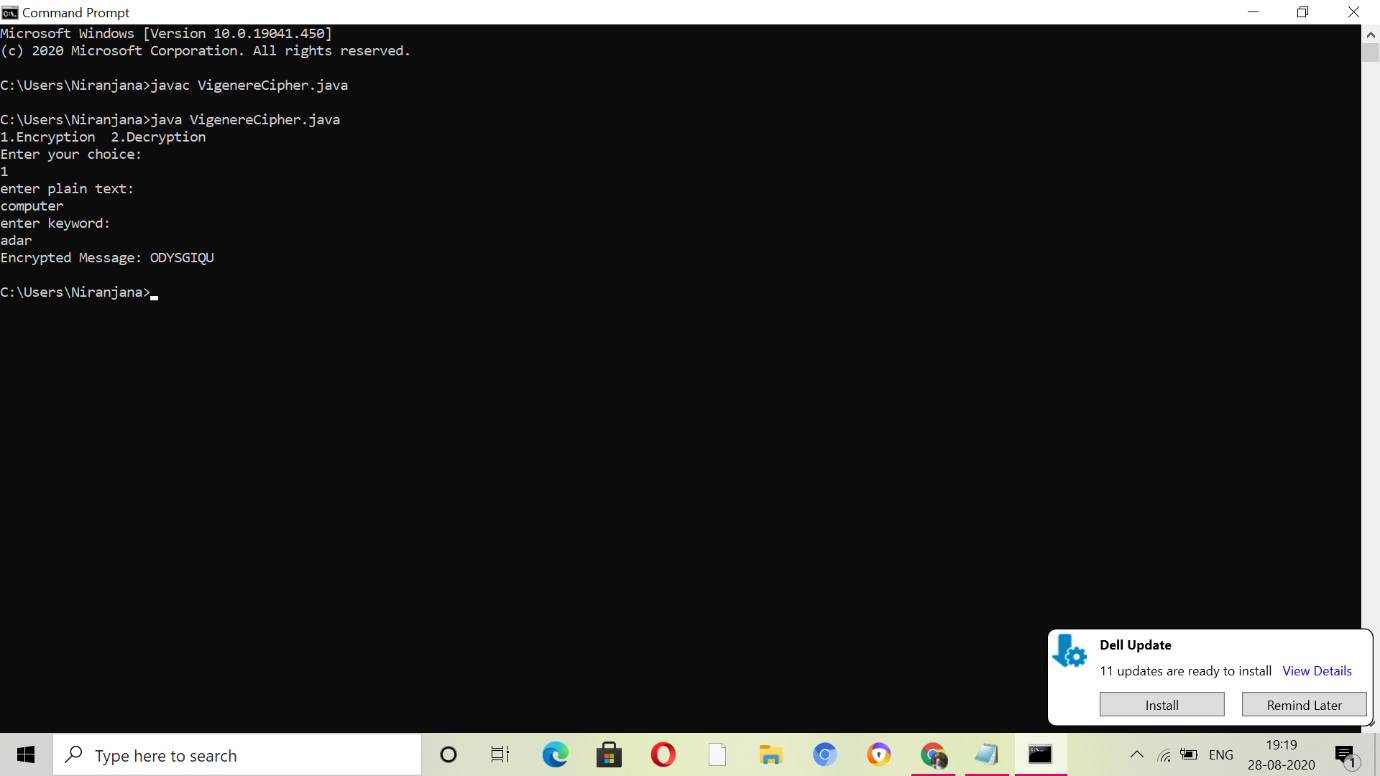
de = String.valueOf(d);

System.out.println("Decrypted Word:"+de);

}

}

**Screenshot:**



1. **Play Fair:**

**Program:**

import java.util.Scanner;

public class playfair

{

private String KeyWord = new String();

private String Key = new String();

private char matrix\_arr[][] = new char[5][5];

public void setKey(String k)

{

String K\_adjust = new String();

boolean flag = false;

K\_adjust = K\_adjust + k.charAt(0);

for (int i = 1; i < k.length(); i++)

{

for (int j = 0; j < K\_adjust.length(); j++)

{

if (k.charAt(i) == K\_adjust.charAt(j))

{

flag = true;

}

}

if (flag == false)

K\_adjust = K\_adjust + k.charAt(i);

flag = false;

}

KeyWord = K\_adjust;

}

public void KeyGen()

{

boolean flag = true;

char current;

Key = KeyWord;

for (int i = 0; i < 26; i++)

{

current = (char) (i + 97);

if (current == 'j')

continue;

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

{

if (current == KeyWord.charAt(j))

{

flag = false;

break;

}

}

if (flag)

Key = Key + current;

flag = true;

}

System.out.println(Key);

matrix();

}

private void matrix()

{

int counter = 0;

for (int i = 0; i < 5; i++)

{

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

{

matrix\_arr[i][j] = Key.charAt(counter);

System.out.print(matrix\_arr[i][j] + " ");

counter++;

}

System.out.println();

}

}

private String format(String old\_text)

{

int i = 0;

int len = 0;

String text = new String();

len = old\_text.length();

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

{

if (old\_text.charAt(tmp) == 'j')

{

text = text + 'i';

}

else

text = text + old\_text.charAt(tmp);

}

len = text.length();

for (i = 0; i < len; i = i + 2)

{

if (text.charAt(i + 1) == text.charAt(i))

{

text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);

}

}

return text;

}

private String[] Divid2Pairs(String new\_string)

{

String Original = format(new\_string);

int size = Original.length();

if (size % 2 != 0)

{

size++;

Original = Original + 'x';

}

String x[] = new String[size / 2];

int counter = 0;

for (int i = 0; i < size / 2; i++)

{

x[i] = Original.substring(counter, counter + 2);

counter = counter + 2;

}

return x;

}

public int[] GetDiminsions(char letter)

{

int[] key = new int[2];

if (letter == 'j')

letter = 'i';

for (int i = 0; i < 5; i++)

{

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

{

if (matrix\_arr[i][j] == letter)

{

key[0] = i;

key[1] = j;

break;

}

}

}

return key;

}

public String encryptMessage(String Source)

{

String src\_arr[] = Divid2Pairs(Source);

String Code = new String();

char one;

char two;

int part1[] = new int[2];

int part2[] = new int[2];

for (int i = 0; i < src\_arr.length; i++)

{

one = src\_arr[i].charAt(0);

two = src\_arr[i].charAt(1);

part1 = GetDiminsions(one);

part2 = GetDiminsions(two);

if (part1[0] == part2[0])

{

if (part1[1] < 4)

part1[1]++;

else

part1[1] = 0;

if (part2[1] < 4)

part2[1]++;

else

part2[1] = 0;

}

else if (part1[1] == part2[1])

{

if (part1[0] < 4)

part1[0]++;

else

part1[0] = 0;

if (part2[0] < 4)

part2[0]++;

else

part2[0] = 0;

}

else

{

int temp = part1[1];

part1[1] = part2[1];

part2[1] = temp;

}

Code = Code + matrix\_arr[part1[0]][part1[1]]

+ matrix\_arr[part2[0]][part2[1]];

}

return Code;

}

public String decryptMessage(String Code)

{

String Original = new String();

String src\_arr[] = Divid2Pairs(Code);

char one;

char two;

int part1[] = new int[2];

int part2[] = new int[2];

for (int i = 0; i < src\_arr.length; i++)

{

one = src\_arr[i].charAt(0);

two = src\_arr[i].charAt(1);

part1 = GetDiminsions(one);

part2 = GetDiminsions(two);

if (part1[0] == part2[0])

{

if (part1[1] > 0)

part1[1]--;

else

part1[1] = 4;

if (part2[1] > 0)

part2[1]--;

else

part2[1] = 4;

}

else if (part1[1] == part2[1])

{

if (part1[0] > 0)

part1[0]--;

else

part1[0] = 4;

if (part2[0] > 0)

part2[0]--;

else

part2[0] = 4;

}

else

{

int temp = part1[1];

part1[1] = part2[1];

part2[1] = temp;

}

Original = Original + matrix\_arr[part1[0]][part1[1]]

+ matrix\_arr[part2[0]][part2[1]];

}

return Original;

}

public static void main(String[] args)

{

playfair x = new playfair();

Scanner sc = new Scanner(System.in);

System.out.println("Enter a keyword:");

String keyword = sc.next();

keyword = keyword.replaceAll("\\s","");

keyword = keyword.toLowerCase();

x.setKey(keyword);

x.KeyGen();

System.out.println("Enter word to encrypt: (Make sure length of message is even)");

String key\_input = sc.next();

key\_input = key\_input.replaceAll("\\s","");

key\_input = key\_input.toLowerCase();

if (key\_input.length() % 2 == 0)

{

System.out.println("Encryption: " + x.encryptMessage(key\_input));

System.out.println("Decryption: " + x.decryptMessage(x.encryptMessage(key\_input)));

}

else

{

System.out.println("Message length should be even");

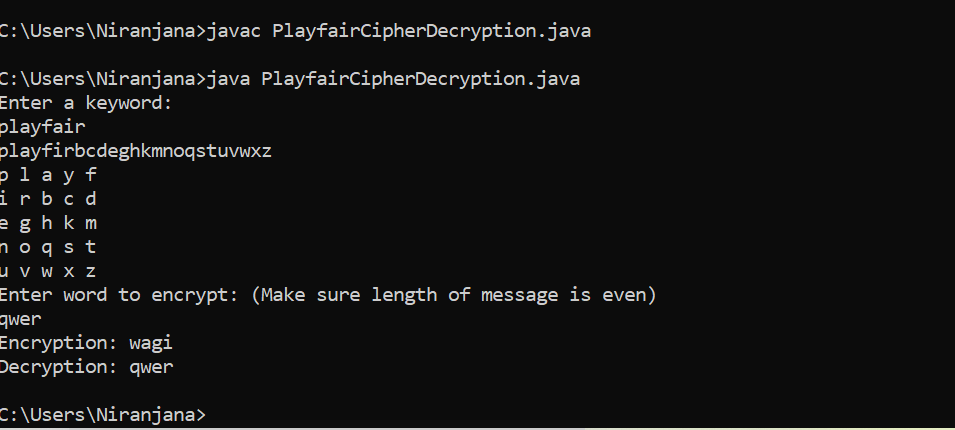
}

sc.close();

}

}

**ScreenShot:**



1. **HillCipher:**

**Program:**

import java.util.\*;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class hillcipher {

int[] lm;

int[][] km;

int[] rm;

static int choice;

int [][] invK;

public void performDivision(String temp, int s)

{

while (temp.length() > s)

{

String line = temp.substring(0, s);

temp = temp.substring(s, temp.length());

calLineMatrix(line);

if(choice ==1){

multiplyLineByKey(line.length());

}else{

multiplyLineByInvKey(line.length());

}

showResult(line.length());

}

if (temp.length() == s){

if(choice ==1){

calLineMatrix(temp);

multiplyLineByKey(temp.length());

showResult(temp.length());

}

else{

calLineMatrix(temp);

this.multiplyLineByInvKey(temp.length());

showResult(temp.length());

}

}

else if (temp.length() < s)

{

for (int i = temp.length(); i < s; i++)

temp = temp + 'x';

if(choice ==1){

calLineMatrix(temp);

multiplyLineByKey(temp.length());

showResult(temp.length());

}

else{

calLineMatrix(temp);

multiplyLineByInvKey(temp.length());

showResult(temp.length());

}

}

}

public void calKeyMatrix(String key, int len)

{

km = new int[len][len];

int k = 0;

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

{

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

{

km[i][j] = ((int) key.charAt(k)) - 97;

k++;

}

}

}

public void calLineMatrix(String line)

{

lm = new int[line.length()];

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

{

lm[i] = ((int) line.charAt(i)) - 97;

}

}

public void multiplyLineByKey(int len)

{

rm = new int[len];

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

{

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

{

rm[i] += km[i][j] \* lm[j];

}

rm[i] %= 26;

}

}

public void multiplyLineByInvKey(int len)

{

rm = new int[len];

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

{

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

{

rm[i] += invK[i][j] \* lm[j];

}

rm[i] %= 26;

}

}

public void showResult(int len)

{

String result = "";

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

{

result += (char) (rm[i] + 97);

}

System.out.print(result);

}

public int calDeterminant(int A[][], int N)

{

int resultOfDet;

switch (N) {

case 1:

resultOfDet = A[0][0];

break;

case 2:

resultOfDet = A[0][0] \* A[1][1] - A[1][0] \* A[0][1];

break;

default:

resultOfDet = 0;

for (int j1 = 0; j1 < N; j1++)

{

int m[][] = new int[N - 1][N - 1];

for (int i = 1; i < N; i++)

{

int j2 = 0;

for (int j = 0; j < N; j++)

{

if (j == j1)

continue;

m[i - 1][j2] = A[i][j];

j2++;

}

}

resultOfDet += Math.pow(-1.0, 1.0 + j1 + 1.0) \* A[0][j1]

\* calDeterminant(m, N - 1);

} break;

}

return resultOfDet;

}

public void cofact(int num[][], int f)

{

int b[][], fac[][];

b = new int[f][f];

fac = new int[f][f];

int p, q, m, n, i, j;

for (q = 0; q < f; q++)

{

for (p = 0; p < f; p++)

{

m = 0;

n = 0;

for (i = 0; i < f; i++)

{

for (j = 0; j < f; j++)

{

b[i][j] = 0;

if (i != q && j != p)

{

b[m][n] = num[i][j];

if (n < (f - 2))

n++;

else

{

n = 0;

m++;

}

}

}

}

fac[q][p] = (int) Math.pow(-1, q + p) \* calDeterminant(b, f - 1);

}

}

trans(fac, f);

}

void trans(int fac[][], int r)

{

int i, j;

int b[][], inv[][];

b = new int[r][r];

inv = new int[r][r];

int d = calDeterminant(km, r);

int mi = mi(d % 26);

mi %= 26;

if (mi < 0)

mi += 26;

for (i = 0; i < r; i++)

{

for (j = 0; j < r; j++)

{

b[i][j] = fac[j][i];

}

}

for (i = 0; i < r; i++)

{

for (j = 0; j < r; j++)

{

inv[i][j] = b[i][j] % 26;

if (inv[i][j] < 0)

inv[i][j] += 26;

inv[i][j] \*= mi;

inv[i][j] %= 26;

}

}

//System.out.println("\nInverse key:");

//matrixtoinvkey(inv, r);

invK = inv;

}

public int mi(int d)

{

int q, r1, r2, r, t1, t2, t;

r1 = 26;

r2 = d;

t1 = 0;

t2 = 1;

while (r1 != 1 && r2 != 0)

{

q = r1 / r2;

r = r1 % r2;

t = t1 - (t2 \* q);

r1 = r2;

r2 = r;

t1 = t2;

t2 = t;

}

return (t1 + t2);

}

public void matrixtoinvkey(int inv[][], int n)

{

String invkey = "";

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

{

for (int j = 0; j < n; j++)

{

invkey += (char) (inv[i][j] + 97);

}

}

System.out.print(invkey);

}

public boolean check(String key, int len)

{

calKeyMatrix(key, len);

int d = calDeterminant(km, len);

d = d % 26;

if (d == 0)

{

System.out.println("Key is not invertible");

return false;

}

else if (d % 2 == 0 || d % 13 == 0)

{

System.out.println("Key is not invertible");

return false;

}

else

{

return true;

}

}

public static void main(String args[]) throws IOException

{

hillcipher obj = new hillcipher();

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Menu:\n1: Encryption\n2: Decryption");

choice = Integer.parseInt(in.readLine());

System.out.println("Enter the line: ");

String line = in.readLine();

System.out.println("Enter the key: ");

String key = in.readLine();

double sq = Math.sqrt(key.length());

if (sq != (long) sq)

System.out.println("Cannot Form a square matrix");

else

{

int size = (int) sq;

if (obj.check(key, size))

{

System.out.println("Result:");

obj.cofact(obj.km, size);

obj.performDivision(line, size);

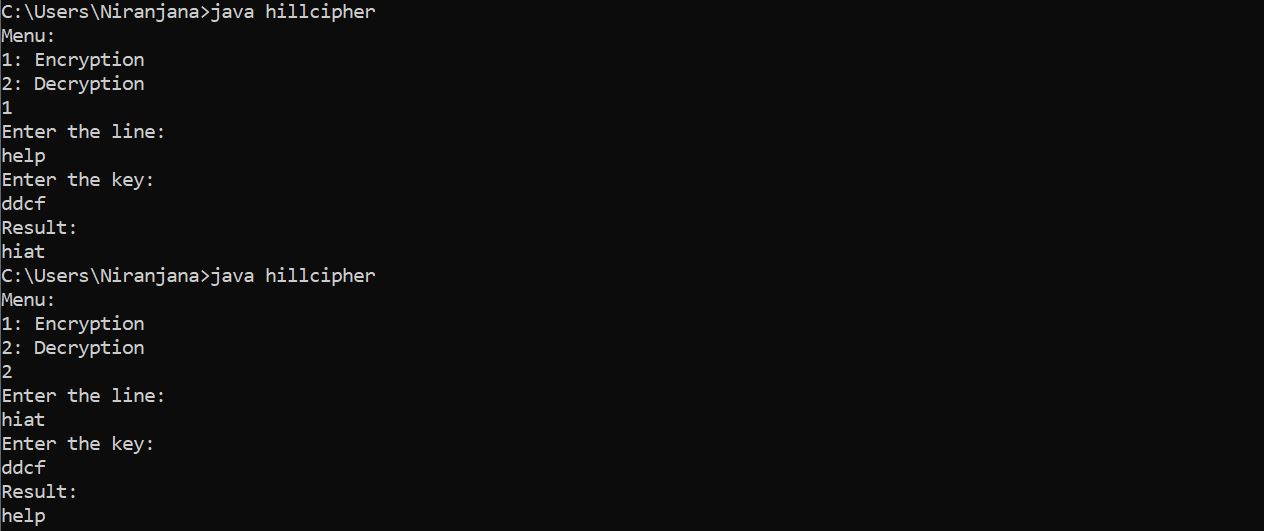
}

}

}

}

**ScreenShot:**



1. **OTP:**

**Program:**

import java.util.\*;

public class OTPCipher{

public static void main(String[] args){

String text ;

Scanner scan = new Scanner(System.in);

System.out.println("Enter a plain text");

text = scan.next();

System.out.println("Enter the key");

//String key = RandomAlpha(text.length());

String key = scan.next();

String enc = OTPEncryption(text,key);

System.out.println("Plaintext : "+text);

System.out.println("Encrypted : "+enc);

System.out.println("Decrypted : "+OTPDecryption(enc,key));

}

public static String RandomAlpha(int len){

Random r = new Random();

String key = "";

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

key = key + (char) (r.nextInt(26) + 'A');

return key;

}

public static String OTPEncryption(String text,String key){

String alphaU = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

String alphaL = "abcdefghijklmnopqrstuvwxyz";

int len = text.length();

String sb = "";

for(int x=0;x<len;x++){

char get = text.charAt(x);

char keyget = key.charAt(x);

if(Character.isUpperCase(get)){

int index = alphaU.indexOf(get);

int keydex = alphaU.indexOf(Character.toUpperCase(keyget));

int total = (index + keydex) % 26;

sb = sb+ alphaU.charAt(total);

}

else if(Character.isLowerCase(get)){

int index = alphaL.indexOf(get);

int keydex = alphaU.indexOf(Character.toLowerCase(keyget));

int total = (index + keydex) % 26;

sb = sb+ alphaL.charAt(total);

}

else{

sb = sb + get;

}

}

return sb;

}

public static String OTPDecryption(String text,String key){

String alphaU = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

String alphaL = "abcdefghijklmnopqrstuvwxyz";

int len = text.length();

String sb = "";

for(int x=0;x<len;x++){

char get = text.charAt(x);

char keyget = key.charAt(x);

if(Character.isUpperCase(get)){

int index = alphaU.indexOf(get);

int keydex = alphaU.indexOf(Character.toUpperCase(keyget));

int total = (index - keydex) % 26;

total = (total<0)? total + 26 : total;

sb = sb+ alphaU.charAt(total);

}

else if(Character.isLowerCase(get)){

int index = alphaL.indexOf(get);

int keydex = alphaU.indexOf(Character.toLowerCase(keyget));

int total = (index - keydex) % 26;

total = (total<0)? total + 26 : total;

sb = sb+ alphaL.charAt(total);

}

else{

sb = sb + get;

}

}

return sb;

}

}

**ScreenShot:**

