**EX1:**

**CAESAR**

import java.util.Scanner;

public class Main {

static Boolean givenInput;

static String inputString;

static int key;

static StringBuffer encryptedString;

static StringBuffer decryptedString;

public Main() {

givenInput = false;

encryptedString = new StringBuffer();

decryptedString = new StringBuffer();

}

public static void Cipher() {

encryptedString = new StringBuffer(); ---initialize string buffer

for (int i = 0; i < inputString.length(); i++) { ----function .length()

if (Character.isUpperCase(inputString.charAt(i))) { ----.isUpperCase, charAt()

char temp = (char) (((int) inputString.charAt(i) + key - 65) % 26 + 65);

---temp char (char)[i+key-65/26+65

encryptedString.append(temp); -----append temp to buffer

} else {

char temp = (char) (((int) inputString.charAt(i) + key - 97) % 26 + 97);

encryptedString.append(temp);

}

}

}

public static void De\_Cipher() {

decryptedString = new StringBuffer();

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

if (Character.isUpperCase(encryptedString.charAt(i))) {

char temp = (char) (((int) encryptedString.charAt(i) - key - 65) % 26 + 65);

decryptedString.append(temp);

} else {

char temp = (char) (((int) encryptedString.charAt(i) - key - 97) % 26 + 97);

decryptedString.append(temp);

}

} ------same as encrypt but (-key)

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in); ------input new scanner in System.in

System.out.println("CAESAR CIPHER");

int choice = 0;

while (choice != -1) {

System.out.println("1. Input text");

System.out.println("2. Input key");

System.out.println("3. Encrypt");

System.out.println("4. Decrypt");

System.out.println("5. Exit");

choice = scan.nextInt();

switch (choice) {

case 3:

if (givenInput) { -----condition tester(givenInput)

------call Cipher()

Cipher();

System.out.println(encryptedString); ------print variable (buffer)

} else {

choice = 3;

}

break;

case 4:

if (givenInput) { -----condition tester

De\_Cipher(); -----call DeCipher()

System.out.println(decryptedString); -----print variable(buffer)

} else {

choice = 3;

}

break;

case 1:

System.out.println("Enter the text - ");

inputString = scan.next();

givenInput = true; ------marker boolean

-----.next() i/p and break

// System.out.println(inputString);

break;

case 2:

System.out.println("Enter the Key - "); .-----next() i/p

key = scan.nextInt(); -------marker set to true and break

givenInput = true;

// System.out.println(key);

break;

case 5:

choice = -1;

break;

default:

System.out.println("Please enter one of the above mentioned options :");

}

}

}

}

**B. PLAYFAIR**

import java.util.Scanner;

public class Main

{

private String KeyWord=new String();

private String Key=new String();

private char key\_matrix[][]= 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 ()

{

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

int counter=0;

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

{

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

{

key\_matrix[i][j]=Key.charAt(counter);

System.out.printf("%s ",key\_matrix[i][j]);

counter++;

}

System.out.println("\n");

}

}

private String format(String old\_text)

{

int i = 0;

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

continue;

}

else

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

}

len = text.length();

for (i = 0; i < len-1; i = i + 2)

{

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

{

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

}

else

{}

}

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);

System.out.print(x[i]+'\t');

counter=counter+2;

}

System.out.println();

return x;

}

public int[] GetDimensions(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(key\_matrix[i][j] == letter)

{

key[0]=i;

key[1]=j;

break;

}

}

}

return key;

}

public String Encrypt(String Source)

{

System.out.println("Encryption Start");

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 = GetDimensions(one);

part2 = GetDimensions(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 + key\_matrix[part1[0]][part1[1]] + key\_matrix[part2[0]][part2[1]];

}

System.out.println("Encrypted Text: "+Code+'\n');

return Code;

}

public String Decrypt (String Code, int s)

{

System.out.println("Decryption Start");

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 = GetDimensions(one);

part2 = GetDimensions(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 + key\_matrix[part1[0]][part1[1]] + key\_matrix[part2[0]][part2[1]];

}

String result=new String();

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

char c=Original.charAt(j);

result=result+c;

}

System.out.println("Decrypted Text: "+result+"\n");

return result;

}

public static void main(String[] args)

{

Main x=new Main();

int choice = 0;

Scanner sc = new Scanner(System.in);

String key\_input="";

String Encrypted="";

int s=0;

while(choice!=-1){

System.out.println("Enter your choice:\n1.Enter keyword\n2.Enter text to be encrypted\n3.Encrypt\n4.Decrypt\n5.Exit");

choice=sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter keyword for Cipher: ");

String keyword = sc.nextLine();

x.setKey(keyword);

x.KeyGen();

break;

case 2:

System.out.print("Enter text: ");

key\_input = sc.nextLine();

s=key\_input.length();

break;

case 3:

Encrypted= x.Encrypt(key\_input);

break;

case 4:

x.Decrypt(Encrypted,s);

break;

case 5:

choice=-1;

break;

default:

System.out.println("Please enter one of the above mentioned options :");

}

}

}

}

**HILL**

import java.util.\*;

import java.io.\*;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

static int[] lm;

static int[][] keyMatrix;

static int[] rm;

static int choice;

static int [][] inverseKeyMatrix;

static int casevariable;

static String line="";

// Display function to print a ma1trix

public static void displayMatrix(int A[][],int len) {

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

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

System.out.print(A[i][j] + " ");

System.out.println();

}

}

// Perform encryption/decryption

public static void performEncryptionOrDecryption(String temp, int s)

{

while (temp.length() > s)

{

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

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

findColumnMatrix(line);

if(choice ==1){

multiplyColumnByKey(line.length());

showResult(line.length());

}

else if(choice==2){

multiplyColumnByInverseKey(line.length());

showResult(line.length());

}

}

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

if(choice ==1){

findColumnMatrix(temp);

multiplyColumnByKey(temp.length());

showResult(temp.length());

}

else if(choice==2){

findColumnMatrix(temp);

multiplyColumnByInverseKey(temp.length());

showResult(temp.length());

}

}

else if (temp.length() < s)

{

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

temp = temp + 'x';

if(choice ==1){

findColumnMatrix(temp);

multiplyColumnByKey(temp.length());

showResult(temp.length());

}

else if(choice==2){

findColumnMatrix(temp);

multiplyColumnByInverseKey(temp.length());

showResult(temp.length());

}

}

}

// Compute the key matrix

public static void findKeyMatrix(String key, int len)

{

keyMatrix = new int[len][len];

int k = 0;

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

{

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

{

keyMatrix[i][j] = ((int) key.charAt(k)) - casevariable;

k++;

}

}

if(choice==1)

{

System.out.println("\nKEY MATRIX");

System.out.println("----------");

displayMatrix(keyMatrix,len);

System.out.print("\nCipher Text : ");

}

}

// Take each group of input variables and put them into a col matrix

public static void findColumnMatrix(String line)

{

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

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

{

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

}

}

public static void multiplyColumnByKey(int len)

{

rm = new int[len];

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

{

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

{

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

}

rm[i] %= 26;

}

}

public static void multiplyColumnByInverseKey(int len)

{

rm = new int[len];

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

{

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

{

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

}

rm[i] %= 26;

}

}

public static void showResult(int len)

{

String result = "";

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

{

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

}

System.out.print(result);

}

public static int findDeterminant(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]

\* findDeterminant(m, N - 1);

}

break;

}

return resultOfDet;

}

public static void findCoFactor(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) \* findDeterminant(b, f - 1);

}

}

findTranspose(fac, f);

}

static void findTranspose(int fac[][], int r)

{

int i, j;

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

b = new int[r][r];

inv = new int[r][r];

int d = findDeterminant(keyMatrix, 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:");

//matrixtoinverseKeyMatrixey(inv, r);

inverseKeyMatrix = inv;

if(choice==2)

{

System.out.println("\nINVERSE KEY MATRIX");

System.out.println("------------------");

displayMatrix(inverseKeyMatrix,r);

System.out.print("\nOriginal Text : ");

}

}

public static 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);

}

// Check if key matrix is invertible

public static boolean check(String key, int len)

{

findKeyMatrix(key, len);

int d = findDeterminant(keyMatrix, 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

{

String key="";

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

System.out.println("\nOPTIONS");

System.out.println("-------");

System.out.println("1. Encrypt\n2. Decrypt\n3. Exit\n");

choice = -1;

while(choice!=3)

{

System.out.print("Enter option : ");

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

if(choice==1)

{

System.out.print("Enter the Plain Text to Encrypt : ");

line = in.readLine();

System.out.print("Enter the Key : ");

key = in.readLine();

}

else if(choice==2)

{

System.out.print("Enter the Cipher Text to Decrypt : ");

line = in.readLine();

System.out.print("Enter the Key : ");

key = in.readLine();

}

line = line.replaceAll("\\s+","");

if(Character.isUpperCase(line.charAt(0)))

{

casevariable = 65;

}

else

{

casevariable = 97;

}

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

if (sq != (long) sq)

System.out.println("Cannot Form a Square Matrix !\n");

else

{

int size = (int) sq;

if (check(key, size))

{

findCoFactor(keyMatrix, size);

performEncryptionOrDecryption(line, size);

System.out.println("\n");

}

}

}

}

}

**VIGNERE**

import java.util.Scanner;

class Main

{

static String generateKey(String str, String key)

{

int x = str.length();

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

{

if (x == i)

i = 0;

if (key.length() == str.length())

break;

key+=(key.charAt(i));

}

return key;

}

// This function returns the encrypted text

// generated with the help of the key

static String cipherText(String str, String key)

{

String cipher\_text="";

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

{

// converting in range 0-25

int x = (str.charAt(i) + key.charAt(i)) %26;

// convert into alphabets(ASCII)

x += 'A';

cipher\_text+=(char)(x);

}

return cipher\_text;

}

// This function decrypts the encrypted text

// and returns the original text

static String originalText(String cipher\_text, String key)

{

String orig\_text="";

for (int i = 0 ; i < cipher\_text.length() &&

i < key.length(); i++)

{

// converting in range 0-25

int x = (cipher\_text.charAt(i) -

key.charAt(i) + 26) %26;

// convert into alphabets(ASCII)

x += 'A';

orig\_text+=(char)(x);

}

return orig\_text;

}

static void print\_matrix ()

{

char c,d;

for (c = 'A' ; c <= 'Z' ; c++)

{

for ( d=c; d <= 'Z' ; d++)

{

System.out.print(d + " ");

}

for ( d= 'A'; d < c ; d++)

{

System.out.print(d + " ");

}

System.out.println();

}

}

// Driver code

public static void main(String[] args)

{

int choice = 0;

Scanner sc = new Scanner(System.in);

String key\_input="";

String key = "";

String Encrypted="";

int n =0;

while(true){

System.out.println("Enter your choice:\n1.Input PlainText \n2.Input keyword \n3.Print Matrix \n4.Encrypt \n5.Decrypt \n6.Exit");

choice=sc.nextInt();

sc.nextLine();

if(choice == 6){

break;

}

switch (choice) {

case 2:

System.out.print("Enter keyword : ");

String keyword = sc.nextLine();

key = generateKey(key\_input, keyword);

break;

case 1:

System.out.print("Enter PlainText: ");

key\_input = sc.nextLine();

break;

case 3:

print\_matrix();

break;

case 4:

Encrypted = cipherText(key\_input, key);

System.out.println("Ciphertext : "+ Encrypted + "\n");

break;

case 5:

System.out.println("Original/Decrypted Text :" + originalText(Encrypted, key));

break;

default:

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

}

}

}

}

**RAIL**

import java.util.\*;

import java.io.\*;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

public static void main(String arg[]){

Scanner inn=new Scanner (System.in);

System.out.println("Rail Fence cipher:");

int choice = 0;

while (choice != -1) {

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

System.out.println("2. Decrypt");

System.out.println("3. Exit");

choice = inn.nextInt();

switch (choice) {

case 1:

System.out.println("Enter the plaintext for encryption");

String plaintext=inn.next();

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

int rails=inn.nextInt();

encryption(plaintext,rails);

break;

case 2:

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

rails=inn.nextInt();

System.out.println("Enter the ciphertext for decryption:");

String ciphertext=inn.next();

decryption(ciphertext,rails);

System.out.println();

break;

case 3:

choice = -1;

break;

default:

System.out.println("Please enter one of the above mentioned options :");

}

}

// System.out.println("Enter the plaintext for encryption");

// Scanner inn=new Scanner (System.in);

// String plaintext=inn.next();

// plaintext = plaintext.replaceAll("\\s+", "");

// System.out.println(plaintext);

// System.out.println("Decryption process start:");

}

public static void encryption(String str,int rails){

boolean checkdown=false; //check whether it is moving downward or upward

int j=0;

int row=rails; // no of row is the no of rails entered by user

int col=str.length(); //column length is the size of string

char[][] a=new char[row][col];

//we create a matrix of a of row \*col size

for(int i=0;i<col;i++){

//matrix visiting in rails order and putting the character of plaintext

if(j==0||j==row-1)

checkdown=!checkdown;

a[j][i]=str.charAt(i);

if(checkdown){

j++;

}

else

j--;

}

//visiting the matrix in usual order to get ciphertext

for(int i=0;i<row;i++){

for(int k=0;k<col;k++){

System.out.print(a[i][k]+" ");

}

System.out.println();

}

String en="";

System.out.println("----------------------");

for(int i=0;i<row;i++){

for(int k=0;k<col;k++){

if(a[i][k]!=0)

en=en+a[i][k];

}

}

System.out.println(en);//printing the ciphertext

}

public static void decryption(String str,int rails){

boolean checkdown=false;

int j=0;

int row=rails;

int col=str.length();

char[][] a=new char[row][col];

//first of all mark the rails position by \* in the matrix

for(int i=0;i<col;i++){

if(j==0||j==row-1)

checkdown=!checkdown;

a[j][i]='\*';

if(checkdown)j++;

else j--;

}

//now enter the character of cipheetext in the matrix positon that have \* symbol

int index=0;

for(int i=0;i<row;i++){

for(int k=0;k<col;k++){

if(a[i][k]=='\*'&&index<str.length()){

a[i][k]=str.charAt(index++);

}

}

}

// visit each character in rails order as character are put in the encryption function

for(int i=0;i<row;i++){

for (int k=0;k<col;k++){

System.out.print(a[i][k]+ "\t");

}

System.out.println();

}

checkdown=false;

String s="";

j=0;

for(int i=0;i<col;i++){

if( j==0||j==row-1)

checkdown=!checkdown;

s+=a[j][i];

if(checkdown)j++;

else j--;

}

System.out.print(s);//print the plaintext that was decrypted by rail fence cipher

}

}

**ROW:**

import java.util.\*;

class Main{

static String message;

static String key;

static int columnCount;

static int rowCount;

static int plainText[][];

static int cipherText[][];

static String ct;

public static void main(String sap[]){

Scanner sc = new Scanner(System.in);

System.out.println("Row Transposition Cipher:");

int choice = 0;

while (choice != -1) {

System.out.println("1. Input plaintext");

System.out.println("2. Input key ");

System.out.println("3. Encrypt");

System.out.println("4. Decrypt");

System.out.println("5. Exit");

choice = sc.nextInt();

switch (choice) {

case 1:

System.out.print("\nEnter plaintext(enter in lower case): ");

message = sc.next();

break;

case 2:

System.out.print("\nEnter key in numbers: ");

key = sc.next();

break;

case 3:

columnCount = key.length();

rowCount = (message.length()+columnCount)/columnCount;

plainText = new int[rowCount][columnCount];

cipherText =new int[rowCount][columnCount];

cipherText = encrypt(plainText, cipherText, message, rowCount, columnCount, key);

ct = "";

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

{

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

{

if(cipherText[j][i] == 0)

ct = ct + 'x';

else{

ct = ct + (char)cipherText[j][i];

}

}

}

System.out.print("\nCipher Text: " + ct);

System.out.println();

break;

case 4:

plainText = decrypt(plainText, cipherText, ct, rowCount, columnCount, key);

String pt = "";

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

{

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

{

if(plainText[i][j] == 0)

pt = pt + "";

else{

pt = pt + (char)plainText[i][j];

}

}

}

System.out.print("\nPlain Text: " + pt);

System.out.println();

break;

case 5:

choice = -1;

break;

default:

System.out.println("Please enter one of the above mentioned options :");

}

}

}

static int[][] encrypt(int plainText[][], int cipherText[][], String message, int rowCount, int columnCount, String key){

int i,j;

int k=0;

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

{

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

{

if(k < message.length())

{

plainText[i][j] = (int)message.charAt(k);

k++;

}

else

{

break;

}

}

}

System.out.print("PlainText Array: \n");

for(i=0;i<rowCount;i++){

for(j=0;j<columnCount;j++){

System.out.print((char)plainText[i][j]+"\t");

}

System.out.println();

}

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

{

int currentCol= ( (int)key.charAt(i) - 48 ) -1;

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

{

cipherText[j][i] = plainText[j][currentCol];

}

}

System.out.print("Cipher Array(read column by column): \n");

for(i=0;i<rowCount;i++){

for(j=0;j<columnCount;j++){

System.out.print((char)cipherText[i][j]+"\t");

}

System.out.println();

}

return cipherText;

}

static int[][] decrypt(int plainText[][], int cipherText[][], String message, int rowCount, int columnCount, String key){

int i,j;

int k=0;

System.out.print("Cipher Array: \n");

for(i=0;i<rowCount;i++){

for(j=0;j<columnCount;j++){

System.out.print((char)cipherText[i][j]+"\t");

}

System.out.println();

}

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

{

int currentCol= ( (int)key.charAt(i) - 48 ) -1;

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

{

plainText[j][currentCol] = cipherText[j][i];

}

}

System.out.print("Plain Array(read row by row): \n");

for(i=0;i<rowCount;i++){

for(j=0;j<columnCount;j++){

System.out.print((char)plainText[i][j]+"\t");

}

System.out.println();

}

return plainText;

}

}

**3. AES:**

import javax.swing.\*;

import java.security.SecureRandom;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Random ;

class Main {

byte[] skey = new byte[1000];

String skeyString;

static byte[] raw;

String inputMessage,encryptedData,decryptedMessage,keyMessage;

public Main() {

try {

// generateSymmetricKey();

inputMessage=JOptionPane.showInputDialog(null,"Enter message to encrypt");

byte[] ibyte = inputMessage.getBytes();

keyMessage=JOptionPane.showInputDialog(null,"Enter key for encryption");

raw = keyMessage.getBytes();

byte[] ebyte=encrypt(raw, ibyte);

String encryptedData = new String(ebyte);

System.out.println("Encrypted message "+encryptedData);

JOptionPane.showMessageDialog(null,"Encrypted Data "+"\n"+encryptedData);

byte[] dbyte= decrypt(raw,ebyte);

String decryptedMessage = new String(dbyte);

System.out.println("Decrypted message "+decryptedMessage);

JOptionPane.showMessageDialog(null,"Decrypted Data "+"\n"+decryptedMessage);

}

catch(Exception e) {

System.out.println(e);

}

}

void generateSymmetricKey() {

try {

Random r = new Random();

int num = r.nextInt(10000);

String knum = String.valueOf(num);

byte[] knumb = knum.getBytes();

skey=getRawKey(knumb);

skeyString = new String(skey);

System.out.println("AES Symmetric key = "+skeyString);

}

catch(Exception e) {

System.out.println(e);

}

}

private static byte[] getRawKey(byte[] seed) throws Exception {

KeyGenerator kgen = KeyGenerator.getInstance("AES");

SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");

sr.setSeed(seed);

kgen.init(128, sr); // 192 and 256 bits may not be available

SecretKey skey = kgen.generateKey();

raw = skey.getEncoded();

return raw;

}

private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {

SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.ENCRYPT\_MODE, skeySpec);

byte[] encrypted = cipher.doFinal(clear);

return encrypted;

}

private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {

SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.DECRYPT\_MODE, skeySpec);

byte[] decrypted = cipher.doFinal(encrypted);

return decrypted;

}

public static void main(String args[]) {

Main aes = new Main();

}}