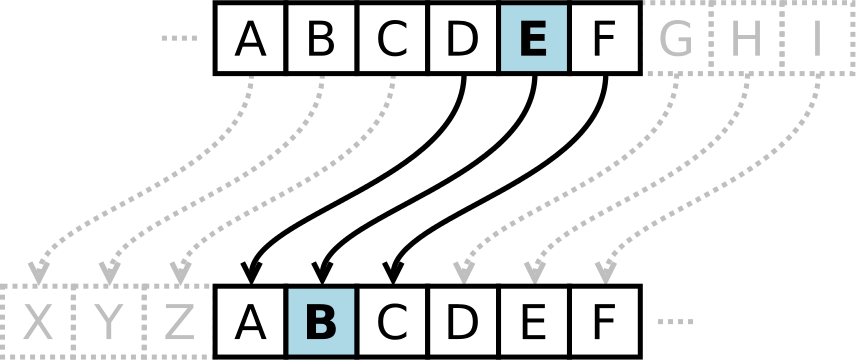
***Aim:*** Write a program to implement Classical/ traditional ciphers techniques.

***Theory:***

***Caesar Cipher***

The Caesar cipher, also known as a shift cipher, is one of the simplest forms of encryption. It is a substitution cipher where each letter in the original message (called the plaintext) is replaced with a letter corresponding to a certain number of letters up or down in the alphabet.



In this way, a message that initially was quite readable, ends up in a form that can not be understood at a simple glance.

For example, here's the Caesar Cipher encryption of a message, using a right shift of 3.

Plaintext: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG  
Ciphertext: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD

As unreadable as the resulting ciphertext may appear, the Caesar Cipher is one of the weakest forms of encryption one can employ.

* The key space is very small. Using a brute force method, one could easily try all (25) possible combinations in order to decrypt the message without initially knowing the key.
* The structure of the original plaintext remains intact. This makes the encryption method vulnerable to frequency analysis - by looking at how often certain characters or sequences of characters appear, one can discover patterns and potentially discover the key without having to perform a full brute force search.

The Caesar Cipher can be expressed in a more mathematical form as follows:



In plain terms, this means that the encryption of a letter *x* is equal to a shift of *x + n*, where n is the number of letters shifted. The result of the process is then taken under modulo division, essentially meaning that if a letter is shifted past the end of the alphabet, it wraps around to the beginning.

Decryption of the encrypted text (*ciphertext*) would be defined similarly, with instead a subtraction of the shift amount.



First used by Julius Caesar, the Caesar Cipher is one of the more well known older historical encryption methods. While you certainly wouldn't want to use it in today's modern world, a long time ago it might have done the trick.

***Rail-fence Cipher***

The rail fence cipher is a very simple, easy to crack cipher. It is a transposition cipher that follows a simple rule for mixing up the characters in the plaintext to form the ciphertext. The rail fence cipher offers essentially no communication security, and it will be shown that it can be easily broken even by hand.

Although weak on its own, it can be combined with other ciphers, such as a substitution cipher, the combination of which is more difficult to break than either cipher on it's own. Many websites claim that the rail-fence cipher is a simpler "write down the columns, read along the rows" cipher. This is equivalent to using an un-keyed columnar transposition cipher.

## ***Example***

The key for the rail fence cipher is just the number of rails. To encrypt a piece of text, e.g.

defend the east wall of the castle

We write it out in a special way on a number of rails (the key here is 3)

d . . . n . . . e . . . t . . . l . . . h . . . s . . .

. e . e . d . h . e . s . w . l . o . t . e . a . t . e  
. . f . . . t . . . a . . . a . . . f . . . c . . . l .

The ciphertext is read off along the rows:

dnetlhseedheswloteateftaafcl

With a key of 4:

d . . . . . t . . . . . t . . . . . f . . . . . s . . .  
. e . . . d . h . . . s . w . . . o . t . . . a . t . .  
. . f . n . . . e . a . . . a . l . . . h . c . . . l .  
. . . e . . . . . e . . . . . l . . . . . e . . . . . e

The ciphertext is again read off along the rows:

dttfsedhswotatfneaalhcleelee

***Program:***

***Caeser Cipher:***

import java.util.Scanner;

public class CaeserCipher

{

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("Enter the Key: ");

int key = sc.nextInt();

System.out.print("Cipher Text: ");

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

System.out.print("Plain Text: ");

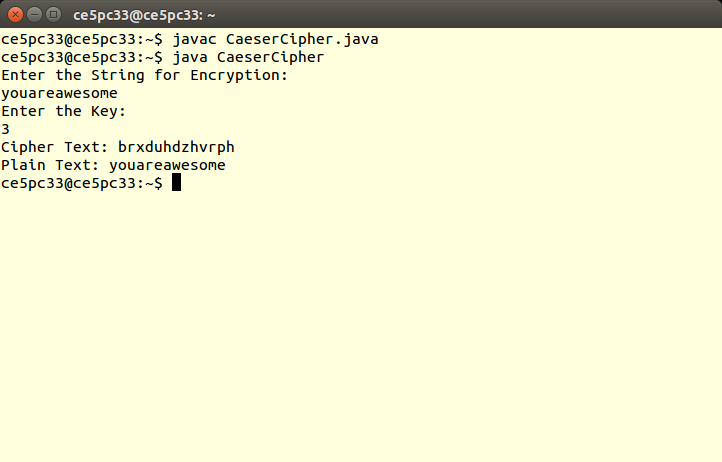
System.out.println(decrypt(encrypt(message, key), key));

sc.close();

}

}

***Output:***



***Rail Fence Cipher***

import java.util.\*;

class RailFenceBasic{

int depth;

String Encryption(String plainText,int depth)throws Exception

{

int r=depth,len=plainText.length();

int c=len/depth;

char mat[][]=new char[r][c];

int k=0;

String cipherText="";

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

{

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

{

if(k!=len)

mat[j][i]=plainText.charAt(k++);

else

mat[j][i]='X';

}

}

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

{

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

{

cipherText+=mat[i][j];

}

}

return cipherText;

}

String Decryption(String cipherText,int depth)throws Exception

{

int r=depth,len=cipherText.length();

int c=len/depth;

char mat[][]=new char[r][c];

int k=0;

String plainText="";

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

{

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

{

mat[i][j]=cipherText.charAt(k++);

}

}

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

{

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

{

plainText+=mat[j][i];

}

}

return plainText;

}

}

class RailFence{

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

{

RailFenceBasic rf=new RailFenceBasic();

Scanner scn=new Scanner(System.in);

int depth;

String plainText,cipherText,decryptedText;

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

plainText=scn.nextLine();

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

depth=scn.nextInt();

cipherText=rf.Encryption(plainText,depth);

System.out.println("Encrypted text is:\n"+cipherText);

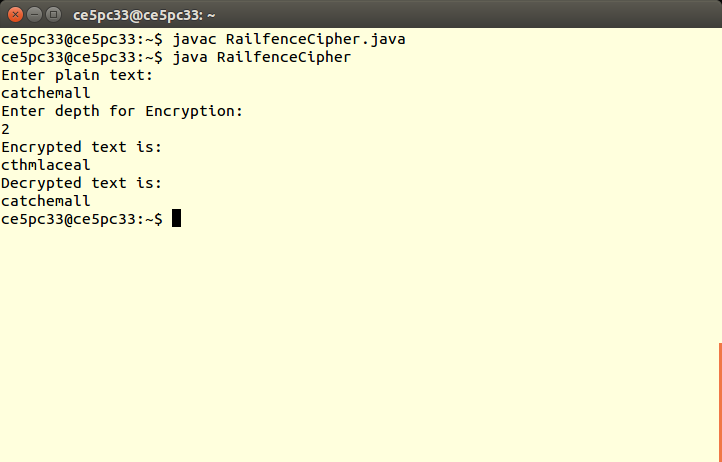
decryptedText=rf.Decryption(cipherText, depth);

System.out.println("Decrypted text is:\n"+decryptedText);

}

}

***Output:***

****

***Conclusion:***  Hence, Successfully studied & implement Classical/traditional ciphers techniques (*Caeser and Rail fence*)