**Caesor Cipher:**

import java.util.Scanner;

public class caesor {

public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

public static String decrypt(String cipherText, int shiftKey)

{

cipherText = cipherText.toLowerCase();

String message = "";

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

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

int keyVal = (charPosition - shiftKey) % 26; //decryption

if (keyVal < 0) {

keyVal = ALPHABET.length() + keyVal; //same for both

}

char replaceVal = ALPHABET.charAt(keyVal);

message += replaceVal;

}

return message;

}

public static String encrypt(String cipherText, int shiftKey)

{

cipherText = cipherText.toLowerCase();

String message = "";

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

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

int keyVal = (charPosition + shiftKey) % 26; //encryption

if (keyVal < 0) {

keyVal = ALPHABET.length() + keyVal; //same for both

}

char replaceVal = ALPHABET.charAt(keyVal);

message += replaceVal;

}

return message;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String message = new String();

System.out.print("Enter the String for Encryption n Decryption: ");

message = sc.next();

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

int key = sc.nextInt();

String demsg=encrypt(message,key);

System.out.println("Encrpyted msg: "+ demsg);

System.out.println("\nDecrypted Message:" + decrypt(demsg, key));

}

}

**Monoalphabetic:**

import java.util.Scanner;

public class Monoalph {

public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

public static String Encrypt(String plainText,int Key)

{

plainText = plainText.toLowerCase();

String cipherText = "";

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

{

int charPositionPT = ALPHABET.indexOf(plainText.charAt(i));

int finalPosition = charPositionPT+Key;

if(finalPosition > 25)

{

finalPosition = finalPosition - 26;

}

char replaceVal = ALPHABET.charAt(finalPosition);

cipherText += replaceVal;

}

return cipherText;

}

public static String Decrypt(String cipherText,int Key)

{

cipherText = cipherText.toLowerCase();

String plainText = "";

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

{

int charPositionCT = ALPHABET.indexOf(cipherText.charAt(i));

int finalPosition = charPositionCT-Key;

if(finalPosition < 0)

{

finalPosition = finalPosition + 26;

}

char replaceVal = ALPHABET.charAt(finalPosition);

plainText += replaceVal;

}

return plainText;

}

public static void main(String[] args) {

//To read input from the keyboard

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 for Encryption: ");

int key = sc.nextInt();

System.out.println("Encrypted Text:");

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

System.out.println("Decrypted Text:");

System.out.println(Decrypt((Encrypt(message, key)), key));

sc.close();

}

}

**Rail Fence Cipher:**

import java.util.Arrays;

import java.util.Scanner;

public class rail {

public static void Encrypt(String str, int n)

{

if (n == 1) {

System.out.print(str);

return ;

}

char[] str1 = str.toCharArray();

int len = str.length();

String[] arr = new String[n];

Arrays.fill(arr, "");

int row = 0;

boolean down = true;

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

{

arr[row] = arr[row] + (str1[i]);

if (row == n - 1)

{

down = false;

}

else if (row == 0)

{

down = true;

}

if (down)

{

row++;

}

else

{

row--;

}

}

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

{

System.out.print(arr[i]);

}

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

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

String str = new String();

str = sc.next(); //plaintext from user

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

int n = sc.nextInt(); //key / rows

System.out.println("Encrypted String: ");

Encrypt(str, n);

}

}

**Simple Columnar Technique:**

import java.util.Scanner;

public class sc {

public static void Encrypt(String plainText, String key)

{

String pt[]=plainText.split("");

String ky[]=key.split("");

int columns = key.length();

int rows;

if(plainText.length() % columns == 0)

{

rows = (plainText.length())/columns + 1;

}

else

{

rows = (plainText.length()/columns) + 2;

}

String[][] mat = new String[rows][columns];

String var="";

int c = 0;

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

{

mat[0][i] = ky[i];

}

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

{

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

{

if(c != plainText.length())

{

var = pt[c];

mat[i][j]=var;

c++;

}

else

break;

}

}

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

{

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

{

if(mat[i][j] == null)

{ mat[i][j]="X";}}}

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

{

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

{

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

}

System.out.println();

}

String output="";

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

{

for(int j = 1; j< rows ; j++)

{

output=output+mat[j][i];

}

}

System.out.println("Encrypted String: "+output);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str = new String();

str = sc.next();

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

String key = new String();

key = sc.next();

Encrypt(str,key);

}}

RSA:

import java.util.Scanner;

public class rsa2 {

public static int pvtkey;

public static void rsa(int p,int q){

int N=p\*q;

int m=(p-1)\*(q-1);

Scanner sc =new Scanner(System.in);

System.out.println("Enter value for public key");

int e=sc.nextInt();

boolean chprime= checkprime(e);

if (chprime){

int [] a = new int[25];

int [] b = new int[25];

int [] d = new int[25];

int [] k = new int[25];

a[0]=1;

b[0]=0;

d[0]=m;

k[0]=0;

a[1]=0;

b[1]=1;

d[1]=e;

k[1]=d[0]/d[1];

System.out.println("a" + "\t " + "b" + "\t " +" d" + "\t " + "k");

System.out.println(a[0] + "\t " + b[0] + "\t " + d[0] + "\t " + k[0]);

System.out.println(a[1] + "\t " + b[1] + "\t " + d[1] + "\t " + k[1]);

int i=2;

while(d[i]!=1){

a[i]=a[i-2]-(a[i-1]\*k[i-1]);

b[i]=b[i-2]-(b[i-1]\*k[i-1]);

d[i]=d[i-2]-(d[i-1]\*k[i-1]);

k[i]=d[i-1]/d[i];

System.out.println(a[i] + "\t " + b[i] + "\t " + d[i] + "\t " + k[i]);

if (d[i]==1)

pvtkey = b[i];

else

i++;

}

if (pvtkey>m)

pvtkey %= m;

else if(pvtkey<0)

pvtkey += m;

else

System.out.println("The public key you chose is not a prime number.");

}

else

System.out.println("The public key you chose is bigger than derived number " +m);

System.out.println();

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

System.out.println("Public Key: " + e);

System.out.println("Private Key: " + pvtkey);

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

}

public static boolean checkprime(int input){

boolean prime=true;

if (input<=1)

prime=false;

else{

for (int i=2; i<=input; i++)

{

if (input%1!=0){

prime=false;

break;

}

}

}

return prime;

}

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

System.out.println("Enter the value of p ");

int p = sc.nextInt();

boolean isPrime=checkprime(p);

if(isPrime){

System.out.println("Enter the value of q ");

int q = sc.nextInt();

isPrime=checkprime(q);

if(isPrime)

rsa(p,q);

else

System.out.println("q is not a prime number ");

}

else

System.out.println("p is not a prime number ");

}

}

**Diffie-Hellman:**

import java.util.Scanner;

public class diffie {

public static void ComputeDF(double q, double p)

{

//Step 2: Compute CipherKey for Ram and Shyam

Scanner sc = new Scanner(System.in);

System.out.println("Enter private key for Ram:");

double pvtRam = sc.nextInt();

System.out.println("Enter private key for Shyam:");

double pvtShyam = sc.nextInt();

double cipherKeyRam, cipherKeyShyam;

cipherKeyRam = Math.pow(p, pvtRam) % q;

cipherKeyShyam = Math.pow(p, pvtShyam)% q;

System.out.println("Cipher Key of Ram: "+cipherKeyRam);

System.out.println("Cipher Key of Shyam: "+cipherKeyShyam);

//Step 3: Compute Shared Secret Key

double SecretKeyRam = Math.pow(cipherKeyShyam, pvtRam) %q;

double SecretKeyShyam = Math.pow(cipherKeyRam, pvtShyam) %q;

if (SecretKeyRam == SecretKeyShyam)

System.out.println("Shared Secret Key = " + (int)SecretKeyRam);

else

System.out.println("Your values don't match. Please try again.");

//sc.close();

}

public static boolean checkprime(double input){

boolean prime=true;

if (input<=1){

prime=false;

return prime;

}

else{

for (int i=2; i<=input/2; i++)

{

if (input%i==0){

prime=false;

break;

}

}

return prime;

}

}

public static void main(String[] args) {

//Step 1 : take q and p as input

Scanner sc = new Scanner(System.in);

System.out.println("Enter value for p(prime no.)- ");

double p = sc.nextInt();

boolean isPrime = checkprime(p);

if (isPrime)

{

System.out.println("Enter value for g(primitive root of q)- ");

double g = sc.nextInt();

ComputeDF(p,g);

}

else

System.out.println("The value you entered for q is not prime number. Please try again.");

}

}

MD5:

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;

public class md {

public static String getMd5(String input)

{

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] messageDigest = md.digest(input.getBytes());

BigInteger no = new BigInteger(1, messageDigest);

String hashtext = no.toString(16);

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

}

catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

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

{

Scanner sc = new Scanner(System.in);

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

String msg = sc.next();

System.out.println("Your HashCode Generated by MD5 is: " + getMd5(msg));

}

}