**Aim: Implementation of Caesar Cipher.**

**Program Code:**

import java.util.Scanner;

public class Caesar {

public static void main(String[] args){

String cip=Caesar.encrypt();

Caesar.decrypt(cip);

}

// Caesar Encryption Function

private static String encrypt() {

char chars[] = {'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'};

String empty = "empty";

Scanner input = new Scanner(System.in);

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

String plainText = input.nextLine();

String cipher = null;

char[] plain = plainText.toCharArray();

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

for(int j = 0 ; j<=25;j++){

if(j<=22){

if(plain[i]==chars[j]){

plain[i] = chars[j+3];

break;

}

}//End nested If

else if(plain[i] == chars[j]){

plain[i] = chars [j-23];

} //End else

} //End nested for loop

} //End of For loop

cipher = String.valueOf(plain);

System.out.println(" cipher text is "+cipher);

Scanner in = new Scanner(System.in);

System.out.println("To Decrypt plaintext enter 1");

int choice = in.nextInt();

if(choice == 1){

return cipher;

}

else{

System.out.println("Thank you");}

return empty;

}

// Caesar Decryption Function

private static String decrypt(String cip) {

char chars[] = {'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'};

String cipher = null;

String empty = "empty";

char[] cipher1 = cip.toCharArray();

if(cip .equals(empty)){

System.out.println(" No text is Decrypted");

}

else{ //char[] cipher1 = cip.toCharArray();

for(int i = 0;i<cipher1.length;i++){

for(int j = 0 ; j<25;j++){

if(j>=3 && cipher1[i]==chars[j]){

cipher1[i] = chars[j-3];

break;

}

if(cipher1[i] == chars[j] && j<3){

cipher1[i] = chars[23+j];

break;

} //End IF

} //End nested for loop

} //End of For loop

}

cipher=String.valueOf(cipher1);

System.out.println(" Plain text is '"+cipher+"'");

return cipher;

}

}

**Output:**



**Aim: Implementation of Vernam Cipher**.

**Program Code:**

import java.lang.Math;

public class Xor {

public static void main(String args[]) {

// This would be the text we encrypt (in this case "hello")

// We convert it to a character array

String text = new String("hello");

char[] arText = text.toCharArray();

// This would be our vernam cipher (should be same length as our text)

// Here we use the same letters, but theoretically should be random

// characters generated on the fly. USE RANDOM LETTERS!

String cipher = new String("XYZHG");

char[] arCipher = cipher.toCharArray();

// Array to hold our encryption (again same length)

char[] encoded = new char[5];

// Encrypt the text by using XOR (exclusive OR) each character

// of our text against cipher.

System.out.println("Encoded " + text + " to be... ");

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

encoded[i] = (char) (arText[i] ^ arCipher[i]);

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

}

System.out.println("\nDecoded to be... ");

// Run through the encrypted text and against the cipher again

// This decrypts the text.

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

char temp = (char) (encoded[i] ^ arCipher[i]);

System.out.print(temp);

}

}

}

**Output:**



**Aim: Implementation of Rail Fence Cipher.**

**Program Code:**

public class Railfence {

public static void main(String args[])

{

String input = "inputstring";

String output = "";

int len = input.length(),flag = 0;

System.out.println("Input String : " + input);

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

output += input.charAt(i);

}

for(int i=1;i<len;i+=2) {

output += input.charAt(i);

}

System.out.println("Ciphered Text : "+output);

}

}

**Output:**



**Aim: Implementation of Modified Caesar Cipher.**

**Program Code:**

import java.io.\*;

public class Caesarcipher

{

public String encrypt(int shift, String line)

{

String result="";

int offset;

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

{

offset=((int)line.charAt(i)+shift)%256;

result+=(char)(offset);

}

return result;

}

public String decrypt(int shift, String line)

{

String result="";

int offset;

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

{

offset=((int)line.charAt(i)-shift)%256;

if(offset<0)

offset+=256;

result+=(char)(offset);

}

return result;

}

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

{

Caesarcipher obj=new Caesarcipher();

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

int choice;

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

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

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

int shift=Integer.parseInt(in.readLine());

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

String line=in.readLine();

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

switch(choice)

{

case 1:System.out.println(obj.encrypt(shift,line));

break;

case 2:System.out.println(obj.decrypt(shift,line));

break;

default:

System.out.println("Invalid input!");

break;

}

}

}

**Output:**



**Aim: Implementation of DES Algorithm.**

**Program Code:**

import javax.crypto.\*;

import java.io.\*;

import java.security.InvalidAlgorithmParameterException;

import java.security.spec.\*;

import javax.crypto.spec.IvParameterSpec;

import java.lang.\*;

public class DesEncrypter {

Cipher ecipher;

Cipher dcipher;

DesEncrypter(SecretKey key) {

try {

ecipher = Cipher.getInstance("DES");

dcipher = Cipher.getInstance("DES");

ecipher.init(Cipher.ENCRYPT\_MODE, key);

dcipher.init(Cipher.DECRYPT\_MODE, key);

} catch (javax.crypto.NoSuchPaddingException e) {

} catch (java.security.NoSuchAlgorithmException e) {

} catch (java.security.InvalidKeyException e) {

}

}

public String encrypt(String str) {

try {

// Encode the string into bytes using utf-8

byte[] utf8 = str.getBytes("UTF8");

// Encrypt

byte[] enc = ecipher.doFinal(utf8);

// Encode bytes to base64 to get a string

return new sun.misc.BASE64Encoder().encode(enc);

} catch (javax.crypto.BadPaddingException e) {

} catch (IllegalBlockSizeException e) {

} catch (UnsupportedEncodingException e) {

} catch (java.io.IOException e) {

}

return null;

}

public String decrypt(String str) {

try {

// Decode base64 to get bytes

byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(str);

// Decrypt

byte[] utf8 = dcipher.doFinal(dec);

// Decode using utf-8

return new String(utf8, "UTF8");

} catch (javax.crypto.BadPaddingException e) {

} catch (IllegalBlockSizeException e) {

} catch (UnsupportedEncodingException e) {

} catch (java.io.IOException e) {

}

return null;

}

public static void main(String args[])

{

System.out.println();

System.out.println("----\*--Encrypting string using DES--\*----");

System.out.println();

try {

// Generate a temporary key. In practice, you would save this key.

SecretKey key =

KeyGenerator.getInstance("DES").generateKey();

// Create encrypter/decrypter class

DesEncrypter encrypter = new DesEncrypter(key);

String s="Don't tell anybody!";

String d="Hello";

// Encrypt

String encrypted = encrypter.encrypt(s);

// Decrypt

String decrypted = encrypter.decrypt(encrypted);

System.out.println("Original string is : " + s);

System.out.println("Encrypted string is: " + encrypted);

System.out.println("Decrypted string is: "+decrypted);

}

catch (Exception e) {}

}

}

**Output:**



**Aim: Implementation of AES Algorithm**.

**Program Code:**

import java.security.\*;

import javax.crypto.\*;

import java.io.\*;

public class AES\_StringEncrypter

{

Cipher ecipher;

Cipher dcipher;

AES\_StringEncrypter(SecretKey key)

{

try

{

ecipher = Cipher.getInstance("AES");

dcipher = Cipher.getInstance("AES");

ecipher.init(Cipher.ENCRYPT\_MODE, key);

dcipher.init(Cipher.DECRYPT\_MODE, key);

}

catch (Exception e) {}

}

public String encrypt(String str)

{

try

{

// Encode the string into bytes using utf-8

byte[] utf8 = str.getBytes("UTF8");

// Encrypt

byte[] enc = ecipher.doFinal(utf8);

// Encode bytes to base64 to get a string

return new sun.misc.BASE64Encoder().encode(enc);

}

catch(Exception e) {}

return null;

}

public String decrypt(String str)

{

try

{

// Decode base64 to get bytes

byte[]dec=new sun.misc.BASE64Decoder().decodeBuffer(str);

// Decrypt

byte[] utf8 = dcipher.doFinal(dec);

// Decode using utf-8

return new String(utf8, "UTF8");

}

catch(Exception e) {}

return null;

}

public static void main(String args[])

{

SecretKey key=null;

try

{

// Generate a AES key

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

key = keyGen.generateKey();

}

catch (Exception e)

{

e.printStackTrace();

}

AES\_StringEncrypter dese = new AES\_StringEncrypter(key);

String o = "Welcome to UDCS.....";

String en = dese.encrypt(o);

String de = dese.decrypt(en);

System.out.println("Original Text:"+o);

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

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

}

}

**Output:**



**Aim: Implementation of RSA Algorithm.**

**Program Code:**

import java.math.\*;

import java.security.\*;

public class Rsa3

{

BigInteger p,q,n,d,e,ph,t;

SecureRandom r;

public Rsa3()

{

r=new SecureRandom();

p=new BigInteger(512,100,r);

q=new BigInteger(512,100,r);

System.out.println("prime nos p and q are "+p.intValue()+" , "

+q.intValue());

n=p.multiply(q);

ph=(p.subtract(new BigInteger("1")));

ph=ph.multiply(q.subtract(new BigInteger("1")));

e=new BigInteger("2");//initial

while(ph.gcd(e).intValue() > 1 || e.compareTo(ph) !=-1)

e = e.add(new BigInteger("1"));//or "2" when bug

d=e.modInverse(ph);

System.out.println("public key is ("+n.intValue()+" , "

+e.intValue()+")");

System.out.println("pvt key is ("+n.intValue()+" , " +

d.intValue()+")");

BigInteger msg= new BigInteger("15");

System.out.println("\nMessage is: "+msg);

BigInteger enmsg=encrypt(msg,e,n);

System.out.println("\nEncrypted msg is: "+enmsg.intValue());

BigInteger demsg=decrypt(enmsg,d,n);

System.out.println("\nDecrypted msg is: "+demsg.intValue());

}

BigInteger encrypt(BigInteger msg,BigInteger e,BigInteger n)

{

return msg.modPow(e, n);

}

BigInteger decrypt(BigInteger msg,BigInteger d,BigInteger n)

{

return msg.modPow(d, n);

}

public static void main(String[] args)

{

new Rsa3();

}

}

**Output:**



**Aim: Implementation of RC4 Algorithm.**

**Program Code:**

class RC4Demo

{

String strPlainText;

static char cipher[];

RC4Demo(String strPlainText,int []key)

{

this.strPlainText = strPlainText;

int S[] = new int[255];

cipher = new char[strPlainText.length()];

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

{

S[i] = i;

}

int i=0;

int j=0;

for (int k=0;k < strPlainText.length();k++)

{

int modk = (k%key.length);

int Kc = key[modk];

j = (S[i] + j + Kc) % 256 + 1;

int temp = S[i];

S[i] = S[j];

S[j] = temp;

int Sc = (S[i]+S[j]) % 256;

int Ck = S[Sc];

cipher[k] = (char) (Ck ^ (int)strPlainText.charAt(k));

i = i+1;

}

}

public static void main(String[] args)

{

int K[] = {1, 2, 3, 4, 5};

String strOriginal = "Hello World";

System.out.println("Original String--> "+strOriginal);

new RC4Demo(strOriginal,K);

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

{ System.out.print(" "+cipher[i]); }

}

}

**Output:**

