**a) Caesar Cipher:-**

import java.util.Scanner;

public class Caesar

{

public static final String ALPHABET= "abcdefghijklmnopqrstuvwxyz";

public static String encryptData(String inputStr, int shiftKey)

{

inputStr = inputStr.toLowerCase();

String encryptStr = "";

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

{

int pos = ALPHABET.indexOf(inputStr.charAt(i));

int encryptPos = (shiftKey + pos) % 26;

char encryptChar = ALPHABET.charAt(encryptPos);

encryptStr += encryptChar;

}

return encryptStr;

}

public static String decryptData(String inputStr, int shiftKey)

{

inputStr = inputStr.toLowerCase();

String decryptStr = " ";

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

{

int pos = ALPHABET.indexOf(inputStr.charAt(i));

int decryptPos = (pos - shiftKey) % 26;

if (decryptPos < 0)

{

decryptPos = ALPHABET.length() + decryptPos;

}

char decryptChar = ALPHABET.charAt(decryptPos);

decryptStr += decryptChar;

}

return decryptStr;

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string for encryption using Caesar Cipher:");

String inputStr = sc.nextLine();

System.out.println("Enter the value by which each character in the plaintext message gets shifted: ");

int shiftKey = Integer.valueOf(sc.nextLine());

System.out.println("Encrypted Data ===> "+encryptData(inputStr, shiftKey));

System.out.println("Decrypted Data ===> "+decryptData(encryptData(inputStr, shiftKey), shiftKey));

sc.close();

}

}

**b) Monoalphabetic Cipher:-**

import java.io.\*;

class monoalphabetic

{

public static char normalChar[] = { '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' };

public static char codedChar[] = { 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P',

'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'Z', 'X', 'C', 'V', 'B', 'N', 'M' };

public static String stringEncryption(String s)

{

String encryptedString = "";

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

{

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

{

if (s.charAt(i) == normalChar[j])

{

encryptedString += codedChar[j];

break;

}

if (s.charAt(i) < 'a' || s.charAt(i) > 'z')

{

encryptedString += s.charAt(i);

break;

}

}

}

return encryptedString;

}

public static String stringDecryption(String s)

{

String decryptedString = " ";

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

{

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

{

if (s.charAt(i) == codedChar[j])

{

decryptedString += normalChar[j];

break;

}

if (s.charAt(i) < 'A' || s.charAt(i) > 'Z')

{

decryptedString += s.charAt(i);

break;

}

}

}

return decryptedString;

}

public static void main(String args[])

{

String str = "Welcome to geeksforgeeks";

System.out.println("Plain text: " + str);

String encryptedString = stringEncryption(str.toLowerCase());

System.out.println("Encrypted message: "+ encryptedString);

System.out.println("Decrypted message: "

+stringDecryption(encryptedString));

}

}

**a) Rail Fence Cipher:-**

import java.io.\*;

public class Railfence

{

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

{

String pt,ct="";

pt="TYCSBNB";

String even="",odd="";

for(int i=0;i<pt.length();i++)3

{

if(i%2==0)

{

even=even+pt.charAt(i);

}

else

{

odd=odd+pt.charAt(i);

}

ct=even+odd;

}

System.out.println("Plain Text:" +pt);

System.out.println("Cipher Text:" +ct);

}

}

**Diffie-Hellman Key**

import java.math.BigInteger;

import java.util.\*;

public class DiffieHellman

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

BigInteger n,g,x,y,k1,k2,A,B;

System.out.println("Enter two Prime Numbers: ");

n=new BigInteger(sc.next());

g=new BigInteger(sc.next());

System.out.println("Person A:Enter your Secret Number");

x=new BigInteger(sc.next());

A=g.modPow(x,n);

System.out.println("Person B:Enter your Secret Number");

y=new BigInteger(sc.next());

B=g.modPow(y,n);

k1=B.modPow(x,n);

k2=A.modPow(y,n);

System.out.println("A's Secret Key:" +k1);

System.out.println("B's Secret Key:" +k2);

}

}

**MD5 algorithm**

import java.util.Scanner;

import javax.xml.bind.DatatypeConverter;

import java.security.MessageDigest;

public class MD5

{

public static void main(String[] args)

{

Scanner sn=new Scanner(System.in);

System.out.print("Please Enter Data for which MD5 is required:");

String data=sn.nextLine();

MD5 md=new MD5();

String hash=md.getMD5Hash(data);

System.out.println("The MD5 (hexadecimalencoded)hash is:"+hash);

}

private String getMD5Hash(String data){

String result=null;

try

{

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

byte[] hash=digest.digest(data.getBytes("UTF-8"));

return bytesToHex(hash);

}

catch(Exception ex)

{

ex.printStackTrace();

}

return result;

}

private String bytesToHex(byte[] hash)

{

return DatatypeConverter.printHexBinary(hash);

}

}

**HMAC**

import java.security.SignatureException;

import java.util.\*;

import java.security.\*;

import javax.crypto.\*;

import java.security.InvalidKeyException;

import javax.crypto.spec.\*;

import javax.crypto.spec.SecretKeySpec;

public class Msgobj

{

private static final String HMAC\_SHA1\_ALGORITHM="hMACsha1";

private static String toHexString(byte[] bytes)

{

Formatter formatter=new Formatter();

for(byte b:bytes)

{

formatter.format("%02x", b);

}

return formatter.toString();

}

public static String calculateRFC2104HMAC(String data,String key)throws SignatureException,

NoSuchAlgorithmException,InvalidKeyException

{

SecretKeySpec signingKey=new

SecretKeySpec(key.getBytes(),HMAC\_SHA1\_ALGORITHM);

Mac mac=Mac.getInstance(HMAC\_SHA1\_ALGORITHM);

mac.init(signingKey);

return toHexString(mac.doFinal(data.getBytes()));

}

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

{

String hmac=calculateRFC2104HMAC("data","key") ;

System.out.println(hmac);

}

}