Cesar Cipher

```
import java.util.Scanner;
public class CesarCipher {
   public static void main(String[] args) {
       String message, encryptmessage = "", decryptmessage= "";
       int key;
       char ch;
       Scanner sc = new Scanner (System.in);
       System.out.println("Enter a plain text : ");
       message = sc.nextLine();
       System.out.println("Enter key : ");
       key = sc.nextInt();
       for(int i = 0; i<message.length(); i++) {
          ch = message.charAt(i);
          if(Character.isLowerCase(ch)) {
             ch = (char)(ch+key);
             if(ch>'z') {
                ch=(char)(ch - 'z' + 'a' - 1); }
                encryptmessage = encryptmessage+ch; }
          else if(Character.isUpperCase(ch)) {
             ch = (char)(ch+key);
             if(ch>'Z') {
                ch=(char)(ch - 'Z' + 'A' - 1); }
                encryptmessage = encryptmessage+ch; }
          else {
                encryptmessage = encryptmessage+ch; }
       }
       System.out.println("EncryptMessage : " +encryptmessage);

       for(int i = 0; i<encryptmessage.length(); i++) {
          ch = encryptmessage.charAt(i);
          if(Character.isLowerCase(ch)) {
             ch = (char)(ch-key);
             if(ch<'a') {
                ch=(char)(ch - 'a' + 'z' + 1); }
                decryptmessage = decryptmessage+ch; }
          else if(Character.isUpperCase(ch)) {
             ch = (char)(ch-key);
             if(ch<'A') {
                ch=(char)(ch - 'A' + 'Z' + 1); }
                decryptmessage = decryptmessage+ch; }
          else {
                decryptmessage = decryptmessage+ch; }
          }
```

```
        System.out.println("DecryptMessage : " +decryptmessage);
    }
}
```

Simple columnar

```
import java.util.Scanner;
public class columnarTransposition{
public static void main(String[] args) {
String text; int key1; int key[] = new int[4];
Scanner sc = new Scanner(System.in);
System.out.println("Enter a String: ");
text = sc.nextLine();
char a[][] = new char[50][4];
int l = text.length();
int row;
if(l%4==0){row = l/4;}
else{row = (l/4)+1;}
int k = 0;
System.out.println("\nMatrix: ");
for(int i = 0; i<row; i++){
for(int j = 0; j<4; j++){
a[i][j]=text.charAt(k);
k++;
System.out.print(a[i][j] + " ");
if(l==k){break;}
}
System.out.println("\n");
}
String s = "";
System.out.println("Enter a key: ");
for(int i = 0; i<4; i++){ key[i] = sc.nextInt();}
for(int i = 0; i<4; i++){
key1 = key[i];
for(int j = 0; j<row; j++){
String c = a[j][key1] + " ";
if(c!="\0"){s =s+c;}
}
}
System.out.println("Cipher Text: " + s);
}
}
```

Mono alpha

```
import java.util.Scanner;
public class monoalpha {
public static char p[] = {'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 ch[] = {'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 doEncryption(String s){
char c[]= new char[s.length()];
for(int i=0; i<s.length();i++){
for(int j=0; j<27; j++){
if(p[j]==s.charAt(i)){
c[i] = ch[j];
break;
}
}
}
return new String(c);
}
public static String doDecryption(String s){
char p1[]= new char[s.length()];
for(int i=0; i<s.length();i++){
for(int j=0; j<27; j++){
if(ch[j]==s.charAt(i)){
p1[i] = p[j];
break;
}
}
}
return new String(p1);
}
public static void main(String[]args){
Scanner sc = new Scanner(System.in);
System.out.println("Enter Message: ");
String en = doEncryption(sc.nextLine().toLowerCase());
System.out.println("Encrypted Message: " + en);
System.out.println("Decrypted Message: " + doDecryption(en));
sc.close();
}
}
```

Rail Fence

```
import java.util.Scanner;
public class RailFence {
```

```
    public static void main(String[] args) {
        String s="";
       Scanner sc=new Scanner(System.in);
System.out.println("Enter a message plain text:");
String rf=sc.nextLine();
System.out.println("\nFirst Half:");
for(int i=0;i<rf.length();i=i+4){
   char c=rf.charAt(i);
   s=s+c;
   System.out.print(c);
  }

System.out.println("\nSecond Half:");
 for(int i=1;i<rf.length();i=i+2){
    char c=rf.charAt(i);
   s=s+c;
   System.out.print(c);
  }
 System.out.println("\nThird Half:");
 for(int i=2;i<rf.length();i=i+4){
    char c=rf.charAt(i);
   s=s+c;
   System.out.print(c);
  }
  System.out.println("\nrail fence tehnique output:"+s);
    }
   }
```

DiffieHellman

```
import java.math.BigInteger;
import java.util.Scanner;
public class DiffieHellman {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter the value of q anda:");
BigInteger q = sc.nextBigInteger();
BigInteger a = sc.nextBigInteger();
System.out.println("Enter Alice's secretkey:");
BigInteger xa = sc.nextBigInteger();
System.out.println("Enter Bob's secretkey:");
BigInteger xb = sc.nextBigInteger();
BigInteger ya = a.modPow(xa, q);
BigInteger yb = a.modPow(xb, q);
System.out.println("Ya = " + ya);
```

```
System.out.println("Yb = " + yb);
BigInteger Ka = yb.modPow(xa, q);
BigInteger Kb = ya.modPow(xb, q);
System.out.println("Secret value of Alice: " +Ka);
System.out.println("Secret value of Bob: " +Kb); }
}
```

—----------

RSA

```
import java.util.Scanner;
import java.util.Random;
import java.math.BigInteger;
import java.nio.charset.StandardCharsets;

public class RSA {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter plain text: ");
        String plainText = sc.nextLine();
        BigInteger pt = new BigInteger(plainText.getBytes(StandardCharsets.UTF_8));
        System.out.println("Enter 2 Prime Numbers: ");
        BigInteger p = new BigInteger(sc.next());
        BigInteger q = new BigInteger(sc.next());
        BigInteger n = p.multiply(q);
        BigInteger one = BigInteger.ONE;
        BigInteger phi = (p.subtract(one)).multiply(q.subtract(one));
        BigInteger e;
        do {
            e = new BigInteger(2 * 512, new Random());
        } while ((e.compareTo(phi) != 1) || (e.gcd(phi).compareTo(one) != 0));
        System.out.println("Public Key: " + e);
        BigInteger d = e.modInverse(phi);
        System.out.println("Private Key: " + d);
        BigInteger ct = pt.modPow(e, n);
        System.out.println("Cipher Text: " + ct);
        BigInteger pt1 = ct.modPow(d, n);
        String decryptedMessage = new String(pt1.toByteArray(), StandardCharsets.UTF_8);
        System.out.println("Decrypted Plain Text: " + decryptedMessage);
        sc.close();
    }
}
```

—----------

SSL/TSL server client

Serverside:

```
import java.io.*;
```

```
import java.net.*;
import java.util.*;
public class Servercode {
public static void main(String args[]) throws IOException{
Socket socket;
ServerSocket serversocket=new ServerSocket(1111);
java.io.InputStreamReader insr=null;
java.io.OutputStreamWriter oswr=null;
java.io.BufferedReader br=null;
java.io.BufferedWriter bw=null;
while (true){
try{
socket=serversocket.accept();
insr=new InputStreamReader(socket.getInputStream());
oswr=new OutputStreamWriter(socket.getOutputStream());
br=new BufferedReader(insr);
bw=new BufferedWriter(oswr);
while(true){
String msgfromclient=br.readLine();
System.out.println(msgfromclient);
bw.newLine();
bw.flush();
if(msgfromclient.equalsIgnoreCase("BYE")){
break;
}
}
bw.close();
}
catch(Exception e){
System.out.println("Exception:"+e);
}

}
}
}
```

Clientside

```
import java.util.*;
import java.io.*;
import java.net.*;
public class Clientcode{
public static void main(String args[]){
Socket socket=null;
java.io.InputStreamReader insr=null;
```

```
java.io.OutputStreamWriter oswr=null;
java.io.BufferedReader br=null;
java.io.BufferedWriter bw=null;
try{
socket=new Socket("localhost",1111);
insr=new InputStreamReader(socket.getInputStream());
oswr=new OutputStreamWriter(socket.getOutputStream());
br=new BufferedReader(insr);
bw=new BufferedWriter(oswr);
Scanner sc=new Scanner(System.in);
System.out.println("Sending messge to server....");
while(true){
String msgtosend=sc.nextLine();
bw.write(msgtosend);
bw.newLine();
bw.flush();
if(msgtosend.equalsIgnoreCase("BYE")){
break;
}
}
bw.close();
}
catch(Exception e){
System.out.println("Error "+e);
}
}
}
```

Vernam cipher

```
import java.util.Scanner;
public class Vernamcipher {
    public static String Encrypt(String PTchange,String keychange){
        String ciphertxt="";
        int len=keychange.length();
        int cipher[]=new int[len];
        for(int i=0;i<len;i++){
            cipher[i]=PTchange.charAt(i)-'A'+keychange.charAt(i)-'A';
          if (cipher[i]>25){cipher [i]=cipher[i]-26;}
         int x=cipher[i]+'A';
           ciphertxt=ciphertxt+(char)x;
          }
        return ciphertxt;
    }
 public static String Decrypt(String enctxt,String keychange){
```

```
        String plaintxt="";
        int len=keychange.length();
        int plain[]=new int[len];
        for(int i=0;i<len;i++){
            plain[i]=(enctxt.charAt(i)-'A')-(keychange.charAt(i)-'A');
          if (plain[i]<0){plain [i]=plain[i]+26;}
         int x=plain[i]+'A';
           plaintxt=plaintxt+(char)x;
          }
        return plaintxt;
    }
public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Plain Text and key of same size");
        String PT=sc.nextLine ();
        String key=sc.nextLine ();
        String PTchange=PT.toUpperCase();
        String keychange=key.toUpperCase();
        String encryptedmsg=Encrypt(PTchange,keychange);
        System.out.println("Encrypted msg="+encryptedmsg);
        String decryptedmsg=Decrypt(encryptedmsg,keychange);
        System.out.println("Decrypted msg="+decryptedmsg);
    }  }
```

—----------

Digital Signature

```
import java.security.*;
import java.nio.file.*;
import java.lang.*;
public class DigitalSignature{
public static void main(String[] args) {
try{
KeyPairGenerator keygen=KeyPairGenerator.getInstance("RSA");
keygen.initialize(2048);
KeyPair kp=keygen.generateKeyPair();
PrivateKey privkey=kp.getPrivate();
PublicKey pubkey=kp.getPublic();
//for one file creating digital signature

byte
data[]=Files.readAllBytes(Paths.get("C:\\Users\\DELL\\OneDrive\\Desktop\\Project
Backup\\text.txt"));
//creating sign for document

java.security.Signature signature =
```

```
Signature.getInstance("SHA256withRSA");
signature.initSign(privkey);
signature.update(data);
byte DGSIGN[]=signature.sign();
Files.write(Paths.get("C:\\Users\\DELL\\OneDrive\\Desktop\\Project
Backup\\text.txt"),DGSIGN,StandardOpenOption.APPEND);
//verify sign now with public key
signature.initVerify(pubkey);
signature.update(data);
boolean check=signature.verify(DGSIGN);
if(check==true){

System.out.println("System Verified");
}
else{
System.out.println("System Not Verified");
}
}
catch(Exception e){
System.out.println(e);
}
}
}
```

—----------

MAC(message authentication code)

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.*;
public class MAC{
 public static String encyptThisString(String input){
       try{
          MessageDigest md=MessageDigest.getInstance("SHA-1");
          byte[] messageDigest=md.digest(input.getBytes());
          BigInteger no=new BigInteger(1,messageDigest);
          String hashtext=no.toString(16);
          return hashtext;
       }
       catch(NoSuchAlgorithmException e){
         throw new RuntimeException(e);
       }
   }
 public static void main(String args[]){
```

```
    System.out.println("Hashcode by SHA-1 for");
    String s1="hello world";
    System.out.println("\n"+s1+": "+encyptThisString(s1));
}
}
```