CAESAR

```
class caesarCipher
{
public static String encode(String enc,int offset)
{

offset = offset %26 +26;
StringBuilder encoded = new StringBuilder();

for(char i:enc.toCharArray())
{
if(Character.isLetter(i))
{
if(Character.isUpperCase(i))
{
encoded.append((char)('A'+(i-'A'+offset)%26));
}
else
{
encoded.append((char)('a'+(i-'a'+offset)%26));
}}
else
{
encoded.append(i);
}
}
return encoded.toString();
```

```
}

public static String decode(String enc,int offset)
{
return encode(enc,26-offset);
}

public static void main(String[] args)throws Exception
{
String msg = "security";
System.out.println("MESSAGE : "+msg);
System.out.println(caesarCipher.encode(msg,3));
System.out.println(caesarCipher.decode(caesarCipher.encode(msg,3),3));
}
}
```

PLAY & HILL

```
class hillCipher {
/* 3x3 key matrix for 3 characters at once */
public static int[][] keymat = new int[][] { { 1, 2, 1 }, { 2, 3, 2 },
{ 2, 2, 1 } }; /* key inverse matrix */
public static int[][] invkeymat = new int[][] { { -1, 0, 1 }, { 2, -1, 0 }, { -2, 2, -1
} };
public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
private static String encode(char a, char b, char c) {
String ret = "";
int x, y, z;
int posa = (int) a - 65;
```

```
int posb = (int) b - 65;
int posc = (int) c - 65;
x = posa * keymat[0][0] + posb * keymat[1][0] + posc * keymat[2][0];
y = posa * keymat[0][1] + posb * keymat[1][1] + posc * keymat[2][1];
z = posa * keymat[0][2] + posb * keymat[1][2] + posc * keymat[2][2];
a = key.charAt(x % 26);
b = key.charAt(y % 26);
c = key.charAt(z % 26);
ret = "" + a + b + c;
return ret;
}
private static String decode(char a, char b, char c) {
String ret = "";
int x, y, z;
int posa = (int) a - 65;
int posb = (int) b - 65;
int posc = (int) c - 65;
x = posa * invkeymat[0][0] + posb * invkeymat[1][0] + posc *
invkeymat[2][0];
y = posa * invkeymat[0][1] + posb * invkeymat[1][1] + posc *
invkeymat[2][1];
z = posa * invkeymat[0][2] + posb * invkeymat[1][2] + posc *
invkeymat[2][2];
a = key.charAt((x % 26 < 0) ? (26 + x % 26) : (x % 26));
b = key.charAt((y % 26 < 0) ? (26 + y % 26) : (y % 26));
c = key.charAt((z % 26 < 0) ? (26 + z % 26) : (z % 26));
ret = "" + a + b + c;
```

```
return ret;

}

public static void main(String[] args) throws java.lang.Exception {

String msg;

String enc = "";

String dec = "";

int n;

msg = ("SecurityLaboratory");

System.out.println("simulation of Hill Cipher\n ---------------------- ");

System.out.println("Input message : " + msg);

msg = msg.toUpperCase();

msg = msg.replaceAll("\\s", "");

/* remove spaces */ n = msg.length() % 3;

/* append padding text X */ if (n != 0) {

   for (int i = 1; i <= (3 - n); i++) {

msg += 'X';

}

}

System.out.println("padded message : " + msg);

char[] pdchars = msg.toCharArray();

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

enc += encode(pdchars[i], pdchars[i + 1], pdchars[i + 2]);

}

System.out.println("encoded message : " + enc);

char[] dechars = enc.toCharArray();

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

dec += decode(dechars[i], dechars[i + 1], dechars[i + 2]);
```

```
}

System.out.println("decoded message : " + dec);

}

}
```

VIGENERE

```
public class vigenereCipher
{
    static String encode(String text, final String key)
    {
        String res="";
        text=text.toUpperCase();
        for(int i=0,j=0;i<text.length();i++)
        {
            char c = text.charAt(i);
            if(c<'A' || c>'Z')
            {
                continue;
            }
            res +=(char)((c+key.charAt(j)-2*'A')%26+'A');
            j=++j%key.length();
        }
        return res;
    }
    static String decode(String text,final String key)
    {
        String res="";
        text=text.toUpperCase();
        for(int i=0,j=0;i<text.length();i++)
        {
            char c = text.charAt(i);
            if(c<'A' || c>'Z')
            {
                continue;
            }
            res +=(char)((c-key.charAt(j)+26)%26+'A');
            j=++j%key.length();
        }
        return res;
    }
    public static void main(String[] args)throws java.lang.Exception{
        String key ="VIGENERECIPHERE";
        String msg= "securityLaboratory\n";
        System.out.println("ciphere\n");
        System.out.println("i/p "+msg);
```

```
        String enc = encode(msg,key);
        System.out.println("\nenc msg "+enc);
        System.out.println("\ndec msg "+decode(enc,key));
    }
}
```

RAILFENCE & ROW COL

```
class railfenceCipherHelper

{



        int depth;

        String encode(String msg, int depth) throws Exception

        {

                int r=depth;

                int l=msg.length();

                int c=l/depth;

                int k=0;

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

                String enc="";

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

                {

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

                        {

                                if(k!=l)

                                {

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

                                }

                                else

                                {

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

```
                        }
                }
        }
        for(int i=0;i<r;i++)
        {
                for(int j=0;j<c;j++)
                {
                        enc+=mat[i][j];
                }
        }
        return enc;
}

String decode(String encmsg, int depth) throws Exception
        {
                int r=depth;
                int l=encmsg.length();
                int c=l/depth;
                int k=0;
                char mat[][]=new char[r][c];
                String dec="";
                for(int i=0;i<r;i++)
                {
                        for(int j=0;j<c;j++)
                        {
                                mat[i][j]=encmsg.charAt(k++);
                        }
```

```
                }
                for(int i=0;i<c;i++)
                {
                        for(int j=0;j<r;j++)
                        {
                                dec+=mat[j][i];
                        }
                }
                return dec;
        }
}

class railFence
{
        public static void main(String args[]) throws java.lang.Exception
        {
                railfenceCipherHelper rf=new railfenceCipherHelper();
                String msg, enc, dec;
                msg="Anna University, Chennai";
                int depth=2;
                enc=rf.encode(msg,depth);
                dec=rf.decode(msg,depth);
                System.out.println("Input Message: " + msg);
                System.out.println("Encrypted Message: " + enc);
                System.out.println("Decrypted Message: " + dec);
        }
}
```

DES

```
import javax.crypto.BadPaddingException;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESKeySpec;

import java.util.*;

import java.io.*;

import java.security.NoSuchAlgorithmException;

import java.security.InvalidKeyException;

import java.security.spec.InvalidKeySpecException;


public class DES

{

public static void main(String args[])throws BadPaddingException,IllegalBlockSizeException,

NoSuchPaddingException,NoSuchAlgorithmException,InvalidKeyException

{


String msg ="This is confidential message";

byte[] message = msg.getBytes();


KeyGenerator kg = KeyGenerator.getInstance("DES");

SecretKey sk = kg.generateKey();
```

```
Cipher c = Cipher.getInstance("DES");

c.init(Cipher.ENCRYPT_MODE,sk);
byte[] encbytes = c.doFinal(message);

c.init(Cipher.DECRYPT_MODE,sk);
byte[] decbytes = c.doFinal(encbytes);

String encdata = new String(encbytes);
String decdata = new String(decbytes);

System.out.println("Message : "+msg);
System.out.println("Encrypted : "+encdata);
System.out.println("Decrypted : "+decdata);
}
}
```

RSA

```
import java.math.*;
import java.util.*;

class RSA {
   public static void main(String args[])
   {
      int p, q, n, z, d = 0, e, i;

      // The number to be encrypted and decrypted
      int msg = 12;
```

```
double c;
BigInteger msgback;

// 1st prime number p
p = 3;

// 2nd prime number q
q = 11;
n = p * q;
z = (p - 1) * (q - 1);
System.out.println("the value of z = " + z);

for (e = 2; e < z; e++) {

   // e is for public key exponent
   if (gcd(e, z) == 1) {
      break;
   }
}
System.out.println("the value of e = " + e);

      for (i = 0; i <= 9; i++) {
   int x = 1 + (i * z);

   // d is for private key exponent
   if (x % e == 0) {
      d = x / e;
```

```
            break;
         }
      }
      System.out.println("the value of d = " + d);

      c = (Math.pow(msg, e)) % n;
      System.out.println("Encrypted message is : " + c);

      // converting int value of n to BigInteger
      BigInteger N = BigInteger.valueOf(n);

      // converting float value of c to BigInteger
      BigInteger C = BigDecimal.valueOf(c).toBigInteger();

      msgback = (C.pow(d)).mod(N);
      System.out.println("Decrypted message is : "
                  + msgback);
   }

   static int gcd(int e, int z)
   {
      if (e == 0)
         return z;
      else
         return gcd(z % e, e);
   }
}
```

DIFFEE

```
class DiffeeHelman
{
public static void main(String args[])
{

int p = 23, g=5, x=4, y=3;

double aliceSends = (Math.pow(g,x))%p;
double bobComputes = (Math.pow(aliceSends,y))%p;
double bobSends = (Math.pow(g,y))%p;
double aliceComputes = (Math.pow(bobSends,x))%p;
double SecretKey = (Math.pow(g,(x*y)))%p;

System.out.println("DIFFEE HELMAN .....\n");
System.out.println("\nALICE SEND : "+aliceSends);
System.out.println("\nBOB COMPUTES : "+bobComputes);
System.out.println("\nBOB SENDS : "+bobSends);
System.out.println("\nALICE COMPUTES : "+aliceComputes);
System.out.println("\n SECRET KEY : "+SecretKey);

if((aliceComputes==SecretKey) && (aliceComputes==bobComputes))
{
System.out.println("Success");
}
else
{
```

```
System.out.println("Failure");
}
}
}
```

SHA 1

```
import java.security.*;

public class SHA1
{
public static void main(String args[])
{
try
{
MessageDigest md = MessageDigest.getInstance("SHA1");
System.out.println("MESSAGE DIGEST .....\n");
System.out.println("ALGORITHM = \n"+md.getAlgorithm());
System.out.println("ToString = \n"+md.toString());

String input = "";
md.update(input.getBytes());
byte[] output = md.digest();
System.out.println();
System.out.println("SHA 1 "+ByteToHex(output));

input = "abc";
md.update(input.getBytes());
output = md.digest();
```

```
System.out.println();
System.out.println("SHA 1 "+ByteToHex(output));

input = "abcdefghijklmnopqrstuvwxyz";
md.update(input.getBytes());
output = md.digest();
System.out.println();
System.out.println("SHA 1 "+ByteToHex(output));
System.out.println();
}
catch(Exception e)
{
System.out.println("Exception"+e);
}
}

private static String ByteToHex(byte[] b)
{
char hexDigit[] = {'0','1','2','3','4','5','6','7','8','9','0','A','B','C','D','E','F'};
StringBuffer buf = new StringBuffer();

for(byte aB:b){
buf.append(hexDigit[(aB>>4)&(0x0f)]);
buf.append(hexDigit[aB & 0x0f]);
}
return buf.toString();
}
```

```
}
```

DSS

```
import java.security.KeyPairGenerator;
import java.security.KeyPair;
import java.security.Signature;
import java.security.PrivateKey;
import java.util.Scanner;

public class DSS
{
public static void main(String args[])throws Exception
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter the message : ");
String msg = sc.nextLine();

KeyPairGenerator kp = KeyPairGenerator.getInstance("DSA");
kp.initialize(2048);
KeyPair pair = kp.generateKeyPair();

PrivateKey pv = pair.getPrivate();

Signature sign = Signature.getInstance("SHA256withDSA");
sign.initSign(pv);

byte[] bytes = "msg".getBytes();
sign.update(bytes);
```

```
byte[] Signature = sign.sign();


System.out.println("DSS : "+new String(Signature,"UTF-8"));

}

}
```