Row and Coloumn

```
import java.util.HashMap;

import java.util.Scanner;

import java.util.Set;

import java.util.regex.Matcher;

import java.util.regex.Pattern;


public class Main {

   private static char[][] matfill(char [][] matrix,char [] key){

      char [][] ans=new char[matrix.length][matrix[0].length];


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


         int index=Character.getNumericValue(key[i]);


         for(int j=0;j<matrix.length;j++){


            ans[j][i]=matrix[j][index];

         }

      }


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

//          for(int j=0;j<matrix[0].length;j++){

//             System.out.print(ans[i][j]+" ");

//          }

//          System.out.println();

//       }

      return ans;

   }


   private static String decryption(char [][] matrix,char[] key){
    String decrypted="";

    char [][] ans=matfill(matrix,key);

    for(int i=0;i<ans.length;i++){
        for(int j=0;j<ans[0].length;j++){
            decrypted+=ans[i][j];
        }
    }
    return decrypted;
}
private static char [][] fillColumnWise(char [] cipher,int klength){
    int rows=cipher.length/klength;
    int cols=klength;
    char [][] matrix=new char[rows][cols];
    int index=0;
    for(int i=0;i<cols;i++){
        for(int j=0;j<rows;j++){
            matrix[j][i]=cipher[index];
            index++;
        }
    }
    return matrix;
}
private static char[][] fillMatrix(char [] plain,int klength){
    int rows=plain.length/klength;
    int cols=klength;
    char [][] matrix=new char[rows][cols];
    int index=0;
    for(int i=0;i<rows;i++){
        for(int j=0;j<cols;j++){
            matrix[i][j]=plain[index];
            index++;
        }
    }
    return matrix;
}
private static char[][] fillMatrix1(char [] plain,int klength){
    int rows=klength;
    int cols=klength;
    char [][] matrix=new char[rows][cols];
    int index=0;
    for(int i=0;i<rows;i++){
        for(int j=0;j<cols;j++){
            matrix[i][j]=plain[index];
            index++;
        }
    }
    return matrix;
}
private static String encryptionOrder(String key){
    HashMap<Integer,Integer> map=new HashMap<>();
    for(int i=0;i<key.length();i++){
        map.put(Character.getNumericValue(key.toCharArray()[i]),i);
    }
    String keytransformed="";
    Set<Integer> set=map.keySet();
    for(Integer i: set){
        keytransformed+=map.get(i);
    }
    return keytransformed;
}
```

```
    private static String encryption(char [][] matrix,int length,String key) {
        char[] karr = key.toCharArray();
        String encrypted = "";
        for (int i = 0; i < karr.length; i++) {
            int val = Character.getNumericValue(karr[i]);

            for (int j = 0; j < matrix.length; j++) {
                encrypted += matrix[j][val];
            }
        }
        return encrypted;
    }
    private static boolean isContainsOtherCharacters(String plain){
        Pattern pattern = Pattern.compile("[^a-z]");
        Matcher matcher = pattern.matcher(plain);
        return matcher.find();
    }
    private static boolean checkForSpecialCharactersOtherThanSpaces(String [] array){
        int count=0;
        for(String i:array){
            if(isContainsOtherCharacters(i)){
                count++;
            }
        }
        return count>0;
    }
//      private static boolean containsDuplicateInKey(String key){
//
//      }
        public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
Scanner scn=new Scanner(System.in).useDelimiter("/n");
System.out.println("Enter key:-");
String key=sc.next();
System.out.println("Enter plain text");
String plain=scn.nextLine();
String gen="";
   String [] arrayOfplain=plain.split(" ");
   while(checkForSpecialCharactersOtherThanSpaces(arrayOfplain)){
      System.out.println("Re-enter Plain Text");
      plain=scn.nextLine();
      arrayOfplain=plain.split(" ");
   }
   for(String i:arrayOfplain){
      gen+=i;
   }

   char [] array=gen.toCharArray();
   char [][] matrix=fillMatrix(array,key.length());
   int cal=gen.length()/key.length();
   if(cal*key.length()<gen.length()){
      int val=(key.length()*key.length())-gen.length();
      for(int i=0;i<val;i++){
         gen+=(char)('z'-i);
      }
      System.out.println(gen);
      array=gen.toCharArray();
      matrix=fillMatrix1(array,key.length());
   }
   if(cal*key.length()>gen.length()){
      int val=(cal*key.length())-gen.length();
      for(int i=0;i<val;i++){
            gen+=(char)('z'-i);
        }
        System.out.println(gen);
        array=gen.toCharArray();
        matrix=fillMatrix(array,key.length());
    }

String ekey=encryptionOrder(key);
String encrypted;
String enc=encryption(matrix,ekey.length(),ekey);
System.out.println("Enter Number of rounds of Encryption:-");
int rounds=sc.nextInt();
int r1=1;
System.out.println("<--Rounds of RC-Encryption-->");
while(rounds!=0){
    for (int i = 0; i < array.length/key.length(); i++) {
        for (int j=0;j<key.length();j++){
            System.out.print(matrix[i][j]+" ");
        }
        System.out.println();
    }
    encrypted=encryption(matrix,ekey.length(),ekey);
    System.out.println();
    System.out.println("Encrypted After "+r1+" round:- "+encrypted);
    matrix=fillMatrix(encrypted.toCharArray(),ekey.length());
    enc=encrypted;
    System.out.println();
    r1++;
    rounds--;
}
    System.out.println();
System.out.println("Final encrypted after all rounds performed:- "+enc);
System.out.println();


System.out.println("Enter Decryption key:-");
String key1=sc.next();
String dkey="";
for(char c:key1.toCharArray()){
   dkey+=(Character.getNumericValue(c)-1);
}
System.out.println("Enter cipher:-");
String cipher=sc.next();
System.out.println("Enter the number of decryption rounds:-");
 int drounds=sc.nextInt();
System.out.println("<--Rounds of RC-Decryption-->");
char [][] dmatrix=fillColumnWise(cipher.toCharArray(),key1.length());
String decrypted;
String dec="";
int r2=1;
 while (drounds--!=0){
    for (int i = 0; i < array.length/key.length(); i++) {
       for (int j=0;j<key.length();j++){
          System.out.print(dmatrix[i][j]+" ");
       }
       System.out.println();
    }
    decrypted=decryption(dmatrix,dkey.toCharArray());
    dec=decrypted;
    System.out.println();
    System.out.println("Decrypted After "+r2+" round:- "+decrypted);
    dmatrix=fillColumnWise(decrypted.toCharArray(),key1.length());
    System.out.println();
```

```
            r2++;
          }
         System.out.println();
         System.out.println("Final Decryption After all rounds:- "+dec);
         System.out.println();
         if(gen.equals(dec)){
            System.out.println("!!!Congratulations you have found the right plain text..");
         }
         else{
            System.out.println("!!!Sorry you have to try again!!!");
         }
   }
}
```

RSA Signature Main

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Main {
   static int power(int x, int y, int p)
   {
      int res = 1;

      x = x % p;
      if (x == 0)
         return 0;

      while (y > 0)
      {
        if ((y & 1) != 0)
            res = (res * x) % p;
        y = y >> 1;
        x = (x * x) % p;
    }
    return res;
}
static boolean isPrime(int n)
{
    if (n <= 1)
    {
        return false;
    }
    if (n <= 3)
    {
        return true;
    }
    if (n % 2 == 0 || n % 3 == 0)
    {
        return false;
    }

    for (int i = 5; i * i <= n; i = i + 6)
    {
        if (n % i == 0 || n % (i + 2) == 0)
        {
            return false;
        }
    }

    return true;
```

```
}
static int modInverse(int A, int M)
{

   for (int X = 1; X < M; X++)
      if (((A % M) * (X % M)) % M == 1)
         return X;
   return 1;
}
static int gcd(int num1,int num2){
   if(num1<num2){
      int temp=num1;
      num1=num2;
      num2=temp;
   }
   int r1=num1;
   int r2=num2;
   int q;
   int r;
   while(r2>0){
      q=r1/r2;
      r=r1-q*r2;
      r1=r2;
      r2=r;
   }
   return r1;
}
public static void main(String[] args) {
   Scanner sc=new Scanner(System.in);
   System.out.println("Enter the value of p:-");
   int p=sc.nextInt();
```

```
while (!isPrime(p)){
   System.out.println("Wrong Input Enter value of p gain:-");
   p=sc.nextInt();
}
System.out.println("Enter the value of q:-");
int q=sc.nextInt();
while (!isPrime(q)){
   System.out.println("Wrong Input Enter value of q gain:-");
   q=sc.nextInt();

}
while (p==q){
   System.out.println("Please re enter any p or q value:-");
   q=sc.nextInt();
}
int n=p*q;
int phin=(p-1)*(q-1);
List<Integer> e_values=new ArrayList<>();
List<Integer> d_values=new ArrayList<>();
for(int i=1;i<=phin;i++){
   if(gcd(i,phin)==1){
      e_values.add(i);
   }
}
System.out.println("E-values:- "+e_values);
for(Integer i:e_values){
   d_values.add(modInverse(i,phin));
}
System.out.println("D-values:- "+d_values);
System.out.println("select e value:- ");
int e=sc.nextInt();
```

```
        System.out.println("select d value:- ");
        int d=sc.nextInt();
        System.out.println("Enter the msg:-");
        int msg=sc.nextInt();
        int s=power(msg,d,n);
        System.out.println("Signature Value:- "+s);
        int sdesh=power(s,e,n);
        System.out.println("S` Value:-"+sdesh);
        if(sdesh==msg){
            System.out.println("\n"+"Verification Done");
        }
        else{
            System.out.println("Not verified");
        }
    }
}
```

Vignere Cipher

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class Main {
    private static final String alphabet = "abcdefghijklmnopqrstuvwxyz";
    private static final String alphabet1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in).useDelimiter("\n");
        List<Integer> list = new ArrayList<>();
        Collections.addAll(list, 1, 3, 5, 7, 11, 15, 17, 19, 21, 23, 25);
```

```
System.out.println("1 FOR ENCRYPTION:");
System.out.println("2 FOR DECRYPTION:");
System.out.println("3 FOR BRUTEFORCE:");
System.out.println("Other Key FOR EXIT:");
int c = sc.nextInt();
boolean result = false;
String pt = "";
String ct = "";
String pt1 = "";
String ptd = "";
String ct1 = "";
String km = "";

String key = "";

int kmf = 0;
String ka = "";
int kaf = 0;
int f = 0;
ArrayList<Integer> spaces = new ArrayList<>();
switch (c) {
  case 1:
    while(!result){
      System.out.println("ENTER Plaintext : ");
      pt=sc.next();
      pt1=pt;
      pt=pt.replaceAll("\\s+","");
      result = pt.matches("[a-z]+");
      if(result==false)
        System.out.println("ENTER CORRECT STRING::");
    }
```

```
            for(int i=0;i<pt1.length();i++){
               if(pt1.charAt(i)==' ') spaces.add(i);
            }
            while (f == 0) {
               System.out.println("ENTER KEY : ");
               key = sc.next();
               result = key.matches("[a-z]+");
               if (result == false)
                  System.out.println("ENTER CORRECT KEY::");
               else
                  f = 1;
            }
            String keyL=generateKey(pt,key);
            String enc1=cipherText(pt,keyL);


            StringBuffer str1 = new StringBuffer(enc1);
//                System.out.println(spaces);
            for (int i = 0; i < pt1.length(); i++) {
               for (int j = 0; j < spaces.size(); j++) {
                  if (spaces.get(j) == i) {
                     str1.insert(i, ' ');
                  }
               }


            }
            enc1 = str1.toString();
            System.out.println("Plaintext : " + pt1);
            System.out.println("Encrypted : " + enc1);
            break;
```

```
case 2:
   while(!result){
      System.out.println("ENTER Ciphertext : ");
      ptd=sc.next();
      pt1=ptd;
      ptd=ptd.replaceAll("\\s+","");
      result = ptd.matches("[A-Z]+");
      if(result==false)
         System.out.println("ENTER CORRECT STRING::");
   }
   for(int i=0;i<pt1.length();i++){
      if(pt1.charAt(i)==' ') spaces.add(i);
   }

   for(int i=0;i<ptd.length();i++){
      int p1=alphabet1.indexOf(ptd.charAt(i));
      pt+=alphabet.charAt(p1);
   }
   while (f == 0) {
      System.out.println("ENTER KEY : ");
      key = sc.next();
      result = key.matches("[a-z]+");
      if (result == false)
         System.out.println("ENTER CORRECT KEY::");
      else
         f = 1;
   }
   String keyE=generateKey(pt,key);
   String dec1=originalText(pt,keyE);
```

```
            StringBuffer str2 = new StringBuffer(dec1);
//                System.out.println(spaces);
            for (int i = 0; i < pt1.length(); i++) {
                for (int j = 0; j < spaces.size(); j++) {
                    if (spaces.get(j) == i) {
                        str2.insert(i, ' ');
                    }
                }

            }
            dec1 = str2.toString();
            System.out.println("Plaintext : " + pt1);
            System.out.println("Encrypted : " + dec1);
            break;


        case 3:

            String ciptext = "";
            String ciptext1 = "";
            boolean ctresult = false;
            boolean outerloop = false;

            while (!outerloop) {
                result = false;
                ctresult = false;
                while (!result) {
                    System.out.println("ENTER Plaintext : ");
                    pt = sc.next();
```

```
    pt1 = pt;
    pt = pt.replaceAll("\\s+", "");
    result = pt.matches("[a-z]+");
    if (result == false)
        System.out.println("ENTER CORRECT STRING::");
}

while (!ctresult) {
    System.out.println("ENTER Ciphertext : ");
    ptd = sc.next();
    ciptext1 = ptd;
    ptd = ptd.replaceAll("\\s+", "");
    ctresult = ptd.matches("[A-Z]+");
    if (ctresult == false)
        System.out.println("ENTER CORRECT STRING::");
}

for(int i=0;i<ptd.length();i++){
    int p33=alphabet1.indexOf(ptd.charAt(i));
    ciptext+=alphabet.charAt(p33);
}

int flag = 1;
if (ciptext1.length() != pt1.length()) {
    System.out.println("Length of both are not same");
    flag = 0;
}
int flagC = 1;
if (flag == 1) {
    for (int i = 0; i < pt1.length(); i++) {
        if (pt1.charAt(i) == ' ') {
}

                if (ciptext1.charAt(i) == ' ') {

                } else {
                    flagC = 0;
                }
            }
        }
    }

    if (flagC == 0) {
        System.out.println("Spaces are not equal or at same place");
    }
    if (flagC == 1 && flag == 1) {
        outerloop = true;
    }
}

String keyBrute = "";

keyBrute=originalText(ciptext,pt);

System.out.println("KEY : "+keyBrute);

break;

  }
}
```

```
static String generateKey(String str, String key)
{
    int x = str.length();

    for (int i = 0; ; i++)
    {
        if (x == i)
            i = 0;
        if (key.length() == str.length())
            break;
        key+=(key.charAt(i));
    }
    return key;
}

static String cipherText(String str, String key)
{
    String cipher_text="";

    for (int i = 0; i < str.length(); i++)
    {
        int x = (alphabet.indexOf(str.charAt(i)) + alphabet.indexOf(key.charAt(i))) %26;
        cipher_text+=alphabet1.charAt(x);
    }
    return cipher_text;
}


static String originalText(String cipher_text, String key)
{
        String orig_text="";


        for (int i = 0 ; i < cipher_text.length() &&
                i < key.length(); i++)
        {
            int x = (alphabet.indexOf(cipher_text.charAt(i)) -
                    alphabet.indexOf(key.charAt(i)) + 26) %26;
            orig_text+=alphabet.charAt(x);
        }
        return orig_text;
    }
}
```