Practical No – 01

Implementing Substitution and Transposition Ciphers

Aim : Design and implement algorithms to encrypt and decrypt messages using classical substitution and transposition techniques.

A. Caesar Cipher
B. Monoalphabetic Cipher
C. Rail Fence Cipher
D. Simple Columnar Technique
E. Vernam Cipher

Source Code :

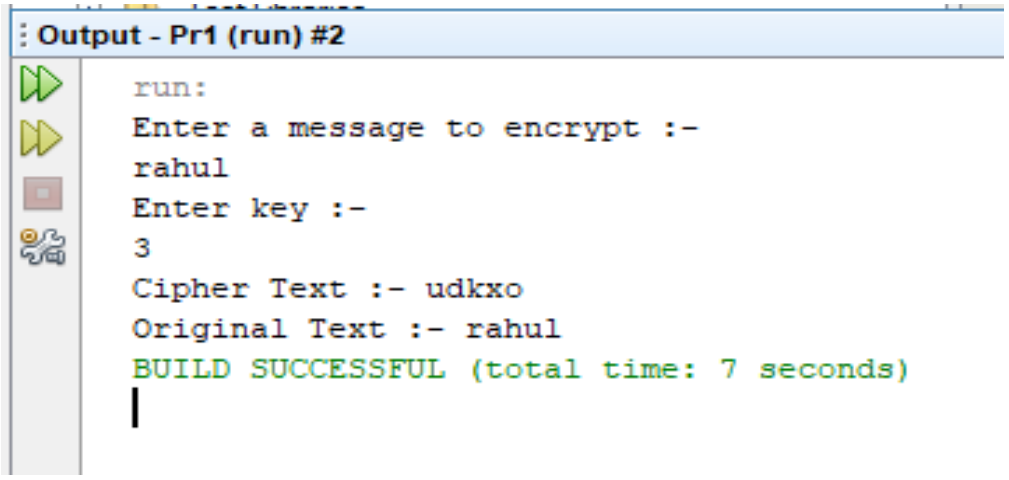
A. CaesarCipher.java

```
import    java.util.Scanner;

public class CeasarCipher {

   String message;

   static int key;

   static String encryptCeasar(String message1, int key1){

      char ch;

      String encryptedMessage = "";

      for(int i = 0; i < message1.length(); ++i){

         ch = message1.charAt(i);

         if(ch >= 'a' && ch <= 'z'){

            ch = (char) (ch + key1);

            if(ch > 'z'){

               ch = (char) (ch - 'z' + 'a' - 1);

            }

            encryptedMessage += ch;

         }

         else if(ch >= 'A' && ch <= 'Z'){

            ch = (char) (ch + key1);

            if(ch > 'Z'){

               ch = (char) (ch - 'Z' + 'A' - 1);
```

```
            }
            encryptedMessage += ch;
        }
        else{
            encryptedMessage += ch;
        }
    }
    return encryptedMessage;
}
static String decryptCeasar(String message1, int key1){
    char ch;
    String decryptedMessage = "";
    for(int i = 0; i < message1.length(); ++i){
        ch = message1.charAt(i);
        if(ch >= 'a' && ch <= 'z'){
            ch = (char) (ch - key1);
            if(ch < 'a'){
                ch = (char) (ch + 'z' - 'a' + 1);
            }
            decryptedMessage += ch;
        }
        else if(ch >= 'A' && ch <= 'Z'){
            ch = (char) (ch - key1);
            if(ch < 'A'){
                ch = (char) (ch + 'Z' - 'A' + 1);
            }
            decryptedMessage += ch;
        }
        else{
            decryptedMessage += ch;
        }
```

```
    }
    return decryptedMessage;
    }
  public static void main(String args[]) {
    String plainText;
    int key;
    String CipherText;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter a message to encrypt :- ");
    plainText = sc.nextLine();
    System.out.println("Enter key :- ");
    key = sc.nextInt();
    CipherText = encryptCeasar(plainText, key);
    System.out.println("Cipher Text :- " + CipherText);
    System.out.println("Original Text :- " + decryptCeasar(CipherText, key));
  }
```

Output :



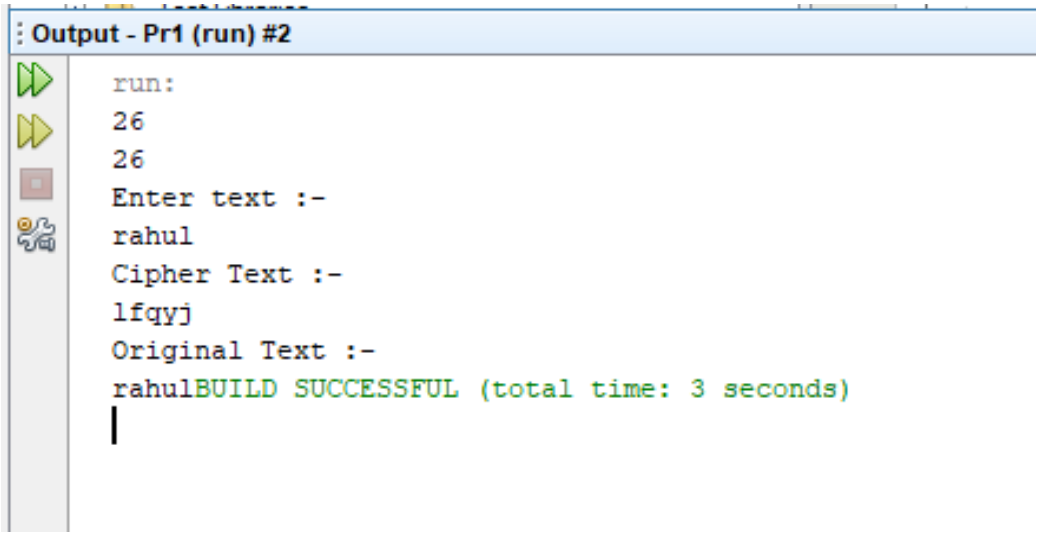
Source Code :

B. Monoalphabetic Cipher

```
import java.util.Scanner;

public class MonoalphabetCipher {

   public static void main(String args[]) {

      final char  RALPHAEBETS[] = {'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'};

      final char  MALPHAEBETS[] = {'f', 'g', 'a', 'b', 'u', 'v', 'k', 'q', 'r', 'h', 'i', 'j', 's', 't', 'c', 'd', 'e', 'l','w', 'x',
'y', 'm', 'n', 'o', 'p', 'z'};

     Scanner sc = new  Scanner(System.in);

      System.out.println(RALPHAEBETS.length);

      System.out.println(MALPHAEBETS.length);

      String plText;

       char citext[] = new char[20];

       char detext[] = new char[20];

       int i, j, l;

       System.out.println("Enter text :- ");

       plText = sc.nextLine();

       l = (plText.length());

       plText = plText.toLowerCase();

       for(i =0; i < l; i++){

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

             if(RALPHAEBETS[j] == plText.charAt(i)){

                citext[i] = MALPHAEBETS[j];

                break;

             }

          }

       }

       System.out.println("Cipher Text :- ");

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

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

       }
```

```
        String b = new String(citext);
        for(i =0; i < l; i++){
            for(j = 0; j < 26; j++){
                if(MALPHAEBETS[j] == b.charAt(i)){
                    detext[i] = RALPHAEBETS[j];
                    break;
                }
            }
        }
        System.out.println("Original Text :- ");
        for(i = 0; i < plText.length(); i++){
            System.out.print(detext[i]);
        }
    }
```

Output :



Source Code :

C. Rail Fence Cipher

```
import java.util.*;
public class Railfence {
   String Encryption(String plainText, int depth) throws Exception{
      int r = depth, len = plainText.length();
      int c = len / depth;
      c = c + 1;
      char mat[][] = new char[r][c];
      int k = 0;
      String cipherText = "";
      for(int i = 0; i < c; i++){
         for(int j = 0; j < r; j++){
            if(k != len){
               mat[j][i] = plainText.charAt(k++);
               System.out.println("mat[" + j + "][" + i + "] = " + mat[j][i]);
            }
         }
      }
      for(int i = 0; i < r; i++){
         for(int j = 0; j < c; j++){
            cipherText += mat[i][j];
         }
      }
      return cipherText;
   }
   String Decryption(String cipherText, int depth) throws Exception{
      int r = depth, len = cipherText.length();
      int c = len / depth;
      char mat[][] = new char[r][c];
```

```
        int k = 0;
        String plainText = "";
        for(int i = 0; i < r; i++){
            for(int j = 0; j < c; j++){
                if(k != len){
                    mat[i][j] = cipherText.charAt(k++);
                }
            }
        }
        for(int i = 0; i < c; i++){
            for(int j = 0; j < r; j++){
                plainText += mat[j][i];
            }
        }
        return plainText;
    }
}
class RailfenceB{
    public static void main(String[] args) throws Exception{

        Scanner sc = new Scanner(System.in);
        int depth;
        String plainText, cipherText, decreptedText;
        System.out.println("Enter plain text :- ");
        plainText = sc.nextLine();
        System.out.println("Enter depth(No of Rails) for Encryptions :- ");
        depth = sc.nextInt();
        Railfence rf = new Railfence();
        cipherText = rf.Encryption(plainText, depth);
        System.out.println("Encrypted text is : -\n" + cipherText);
            decreptedText = rf.Decryption(cipherText, depth);
```

```
    System.out.println("Decrypted text is : -\n" + decreptedText);
  }
```

Output :

```
Output - prac2 (run)
run:
Enter plain text :-
rahul
Enter depth(No of Rails) for Encryptions :-
2
mat[0][0] = r
mat[1][0] = a
mat[0][1] = h
mat[1][1] = u
mat[0][2] = l
Encrypted text is : -
rhlau
Decrypted text is : -
rahul
BUILD SUCCESSFUL (total time: 7 seconds)
```

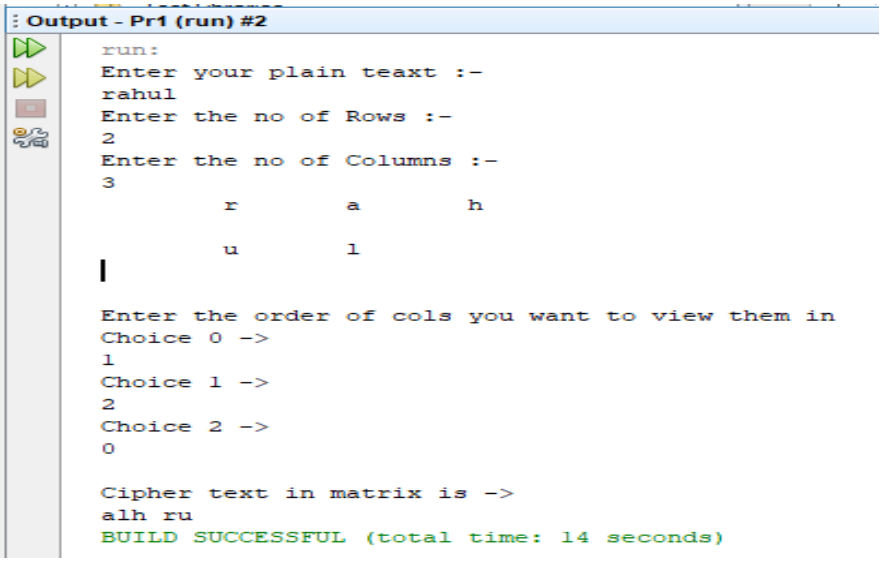
Source Code :

D. Simple Columnar Technique

```
import java.io.*;
import java.util.Scanner;
public class SCT {
    public static void main(String args[]) throws Exception{
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your plain teaxt :- ");
        String accept = sc.nextLine();
        System.out.println("Enter the no of Rows :- ");
        int r = Integer.parseInt(sc.nextLine());
        System.out.println("Enter the no of Columns :- ");
        int c = Integer.parseInt(sc.nextLine());
        int count = 0;
        char cont[][] = new char[r][c];
        for(int i = 0; i < r; i++){
            for(int j = 0; j < c; j++){
                if(count >= accept.length()){
                    cont[i][j] = ' ';
                }
                else{
                    cont[i][j] = accept.charAt(count);
                    count++;
                }
            }
        }
        for(int i = 0; i < r; i++){
            for(int j = 0; j < c; j++){
            System.out.print("\t" + cont[i][j]);
            }
```

```
        System.out.println("\n");
    }
    System.out.println("\nEnter the order of cols you want to view them in");
    int choice[] = new int[c];
    for(int k = 0; k < c; k++){
        System.out.println("Choice " + k + " -> ");
        choice[k] = Integer.parseInt(sc.next());
    }
    System.out.println("\nCipher text in matrix is -> ");
    String cipher = "";
    for(int j = 0; j < c; j ++){
        int k = choice[j];
        for(int i = 0; i < r; i++){
            cipher += cont[i][k];
        }
    }
    System.out.println(cipher);
  }
```

Output :



Source Code :

E. Vernam Cipher

```
import java.util.*;
public class Vernam {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter plain text :- ");
      String plainText = sc.nextLine();
      char[] arText = plainText.toCharArray(
      System.out.println("Enter plain text :- ");
      String key = sc.nextLine();
      char[] arkey = key.toCharArray();
       char[] cipherText = new char[plainText.length()];
      System.out.println("Encoded " + plainText + " to be...");
      for(int i = 0; i < arText.length; i++){
         cipherText[i] = (char) (arText[i] ^ arkey[i]);
         System.out.print(cipherText[i]);
      }

      System.out.println("\n");
      System.out.println("Decoded to be...");
      for(int i = 0; i < cipherText.length; i++){
         char temp = (char) (cipherText[i] ^ arkey[i]);
         System.out.print(temp);
      }

   }
}
```

Output :

```
Output
Debugger Console ×   prac2 (run) ×
run:
Enter plain text :-
rahul
Enter plain text :-
FHDSRKAJ
Encoded rahul to be...
4),&>

Decoded to be...
rahulBUILD SUCCESSFUL (total time: 10 seconds)
```