Date : 21/08/2020

**Practial No 1**

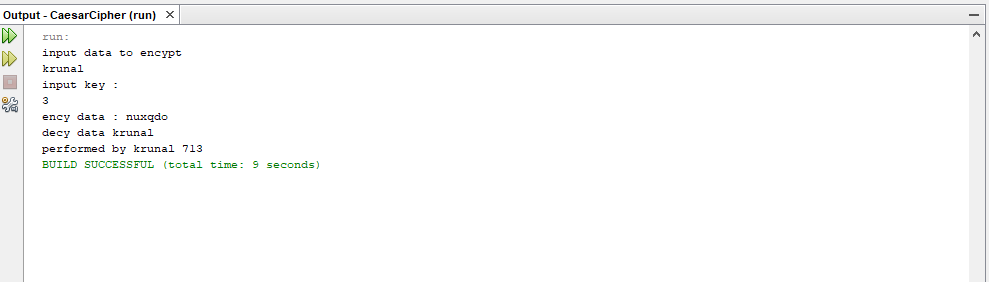
**AIM:** WAP in Java to implement the following Substitution Cipher Techniques.

**CODE**

1. **Caesar cipher :-**

|  |
| --- |
| package javaapplicationins; import java.io.\*;  import java.util.Scanner; public class CaesarCipher  {  public static void main(String[] args)  {  CaesarCipher c=new CaesarCipher(); Scanner s= new Scanner(System.in); System.out.println("Performed by krunal 713"); System.out.println("Input Data to encrypt"); String str=s.nextLine(); System.out.println("Input the key");  int key=s.nextInt();  String encrypted=c.encrypt(str,key); System.out.println("Encrypted Data:"+encrypted); String decrypted=c.decrypt(encrypted,key); System.out.println("Decrypted Data:"+decrypted);  }  String encrypt(String str,int key)  {  String encrypted="";  for(int i=0;i<str.length();i++)  {  int c=str.charAt(i); if(Character.isUpperCase(c))  {  c=c+key; if(c>'Z'){  c=c-26;  }  }  if(Character.isLowerCase(c))  {  c=c+key; if(c>'z'){ c=c-26;  }  }  encrypted +=(char) c;  }  return encrypted;  }  String decrypt(String str,int key)  {  String decrypted="";  for(int i=0;i<str.length();i++)  {  int c=str.charAt(i); if(Character.isUpperCase(c))  {  c=c-key; if(c <'A'){  c=c+26;  }  }  if(Character.isLowerCase(c))  {  c=c- key; if(c <'a'){  c = c + 26;  }  }  decrypted += (char) c;  }  return decrypted;  }  } |

**Output:**

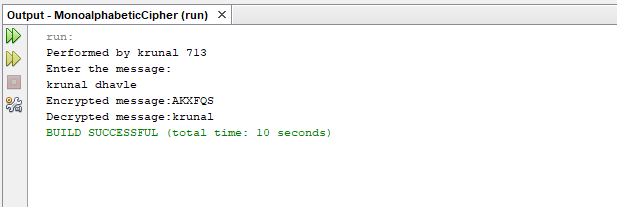
****

**B)Monoalphabetic Cipher**

**Program code:**

|  |
| --- |
| package javaapplicationins; import java.io.\*;  import java.util.Scanner;  public class MonoalphabeticCipher {  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<26;j++)  {  if(p[j]==s.charAt(i))  {  c[i]=ch[j]; break; }  }  }  return(new String(c));  }  public static String doDecryption(String s)  {  char pt[]=new char[(s.length())]; for (int i=0;i<s.length();i++)  {  for(int j=0;j<26;j++)  {  if(ch[j]==s.charAt(i))  {  pt[i]=p[j]; break; }  }  }  return(new String(pt));  }  public static void main(String args[])  {  Scanner sc=new Scanner(System.in); System.out.println("Performed by krunal 713"); System.out.println("Enter the message:");  String en=doEncryption(sc.next().toLowerCase());  System.out.println("Encrypted message:"+en); System.out.println("Decrypted message:"+doDecryption(en));  sc.close();  }  } |

**Output:**

****

Date: 02/09/2020

**Practical no 2**

**AIM:** Write program to implement the following Substitution Cipher Techniques

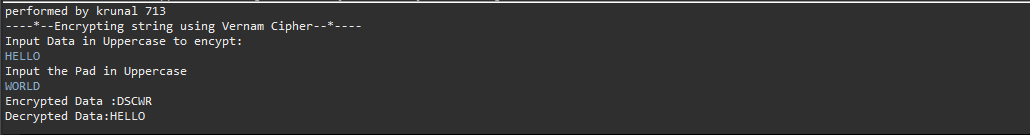
a)Vernam Cipher b)Playfair Cipher

**Code:**

**a)Vernam Cipher**

|  |
| --- |
| import java.util.Scanner;  public class Vernam {  String encrypt(String str, String pad) {  String encrypted = "";  for (int i = 0; i < str.length(); i++) {  int c = str.charAt(i);  int p = pad.charAt(i);  c = (c + p);  if (c > 'Z') {  c = c%26;  c = c+65;  }  encrypted += (char) c;  }  return encrypted;  }  String decrypt(String str, String pad) {  String decrypted = "";  for (int i = 0; i < str.length(); i++) {  int c = str.charAt(i);  int p = pad.charAt(i);  c = (c - p)+26;  if (c < 'A') {  c = (c%26);  c = c+65;  }  decrypted += (char) c;  }  return decrypted;  }  public static void main(String[] args) {  System.***out***.println("performed by krunal 713");  System.***out***.println("----\*--Encrypting string using Vernam Cipher--\*----");  Vernam v = new Vernam();  Scanner s = new Scanner(System.***in***);  System.***out***.println("Input Data in Uppercase to encypt:");  String str = s.nextLine();  System.***out***.println("Input the Pad in Uppercase");  String pad = s.nextLine();  String encrypted = v.encrypt(str, pad);  System.***out***.println("Encrypted Data :" + encrypted);  String decrypted = v.decrypt(encrypted, pad);  System.***out***.println("Decrypted Data:" + decrypted);  }  } |

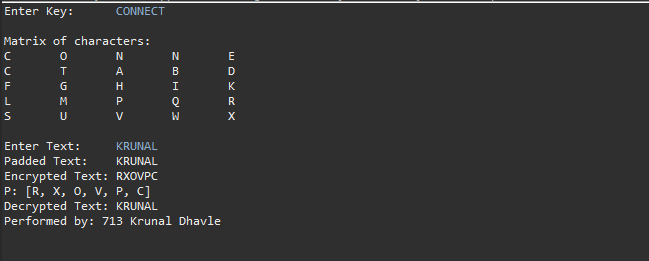
**Output:**



**b)Playfair Cipher**

|  |
| --- |
| import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.util.Arrays;  public class PlayFair {  public static char *keymat*[][] = new char[5][5];  public static String *trans* = "J";  public static char *subs* = 'X';  private static int decrem(int pos) {  if (pos < 0) {  return pos + 5;  } else {  return pos;  }  }  private static int[] srch(char c) {  int i, j;  int[] pos = new int[2];  for (i = 0; i < 5; i++) {  for (j = 0; j < 5; j++) {  if (*keymat*[i][j] == c) {  pos[0] = i;  pos[1] = j;  break;  }  }  }  return pos;  }  private static String encrypt(char c1, char c2) {  int[] pos1 = new int[2];  int[] pos2 = new int[2];  String frag = "";  pos1 = *srch*(c1);  pos2 = *srch*(c2);  if (pos1[0] == pos2[0]) { //condition for same row  c1 = *keymat*[pos1[0]][(pos1[1] + 1) % 5];  c2 = *keymat*[pos2[0]][(pos2[1] + 1) % 5];  frag = ("" + c1 + c2 + "");  } else if (pos1[1] == pos2[1]) { //condition for same column  c1 = *keymat*[(pos1[0] + 1) % 5][pos1[1]];  c2 = *keymat*[(pos2[0] + 1) % 5][pos2[1]];  frag = ("" + c1 + c2 + "");  } else { //condition for different row & column  c1 = *keymat*[pos2[0]][pos1[1]];  c2 = *keymat*[pos1[0]][pos2[1]];  frag = ("" + c1 + c2 + "");  }  return frag;  }  private static String decrypt(char c1, char c2) {  int[] pos1 = new int[2];  int[] pos2 = new int[2];  String frag = "";  pos1 = *srch*(c1);  pos2 = *srch*(c2);  if (pos1[0] == pos2[0]) { //condition for same row  c1 = *keymat*[pos1[0]][*decrem*(pos1[1] - 1) % 5];  c2 = *keymat*[pos2[0]][*decrem*(pos2[1] - 1) % 5];  frag = ("" + c1 + c2 + "");  } else if (pos1[1] == pos2[1]) { //condition for same column  c1 = *keymat*[*decrem*(pos1[0] - 1) % 5][pos1[1]];  c2 = *keymat*[*decrem*(pos2[0] - 1) % 5][pos2[1]];  frag = ("" + c1 + c2 + "");  } else { //condition for different row & column  c1 = *keymat*[pos2[0]][pos1[1]];  c2 = *keymat*[pos1[0]][pos2[1]];  frag = ("" + c1 + c2 + "");  }  return frag;  }  public static void main(String[] args) throws IOException {  BufferedReader br = new BufferedReader(new InputStreamReader(System.***in***));  String key;  int p = 0, k = 0, c = 65;  System.***out***.print("Enter Key:\t");  key = br.readLine();  for (int i = 0; i < 5; i++) {  for (int j = 0; j < 5; j++) {  if (p < key.length()) {  *keymat*[i][j] = key.charAt(p);  p++;  } else {  if ((char) c == 'J') {  c++;  }  for (; k < key.length();) {  if ((char) c == key.charAt(k)) {  k = 0;  c++;  }  k++;  }  *keymat*[i][j] = (char) c;  c++;  k = 0;  }  }  }  System.***out***.println("\nMatrix of characters:");  for (int i = 0; i < 5; i++) {  for (int j = 0; j < 5; j++) {  System.***out***.print(*keymat*[i][j] + "\t");  }  System.***out***.println();  }  String etext = "", dtext = "";  System.***out***.print("\nEnter Text: \t");  String text = br.readLine();  text = text.toUpperCase();  text = text.replaceAll("\\s", ""); //removes whitespaces  text = text.replace(*trans*, "I"); //replaces J with I  text = text.replaceAll("([A-Z])\\1+", "$1" + *subs* + "$1");  if (text.length() % 2 != 0) {  text += *subs*;  }  char[] PTC = text.toCharArray();  System.***out***.println("Padded Text:\t" + text);  for (int i = 0; i < text.length(); i += 2) {  etext += *encrypt*(PTC[i], PTC[i + 1]);  }  System.***out***.println("Encrypted Text:\t" + etext);  char[] OTC = etext.toCharArray();  System.***out***.println("P: " + Arrays.*toString*(OTC));  for (int i = 0; i < etext.length(); i += 2) {  dtext += *decrypt*(OTC[i], OTC[i + 1]);  }  System.***out***.println("Decrypted Text:\t" + dtext);  System.***out***.println("Performed by: 713 Krunal Dhavle");  }} |

**Output**



Date: 28/08/2020

**Practical no 3**

**AIM:** Write program to implement the following Transposition Cipher Techniques

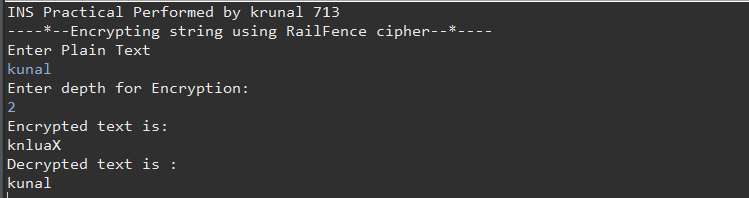
a)Rail Fence Cipher b)Simple Columnar Technique

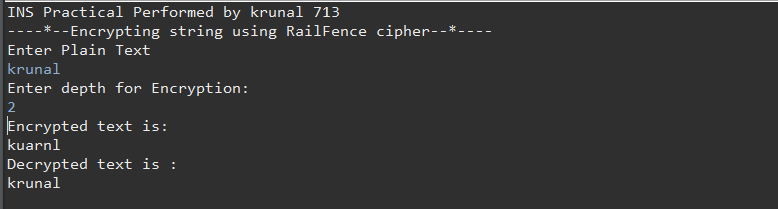
**Code:**

**a)Rail Fence Cipher**

|  |
| --- |
| import java.util.Scanner;  import java.util.logging.Level;  import java.util.logging.Logger;  public class Rails {  String Encrypytion(String plainText,int depth) throws Exception{  int r=depth, len = plainText.length();  int c= len/depth;  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++);  }  }  }  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++){  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 ;  }  public static void main(String[] args) {  try {  System.***out***.println("INS Practical Performed by krunal 713");  System.***out***.println("----\*--Encrypting string using RailFence cipher--\*----");  Rails rf = new Rails();  Scanner scn = new Scanner(System.***in***);  int depth;  String plainText,cipherText,decryptedText;  System.***out***.println("Enter Plain Text");  plainText=scn.nextLine();  System.***out***.println("Enter depth for Encryption:");  depth=scn.nextInt();  while(plainText.length()%depth!=0){  plainText+='X';  }  cipherText=rf.Encrypytion(plainText, depth);  System.***out***.println("Encrypted text is:\n" + cipherText);  decryptedText=rf.Decryption(cipherText, depth);  decryptedText=decryptedText.replace("X","");  System.***out***.println("Decrypted text is :\n"+decryptedText);  }catch (Exception ex){  Logger.*getLogger*(Rails.class.getName()).log(Level.***SEVERE***,null,ex);  }  }  } |

**Output:**





**b) Simple Columnar Technique**

|  |
| --- |
| package prac3b;  import java.io.BufferedReader;  import java.io.\*;  import java.io.InputStreamReader;  import java.util.logging.Level;  import java.util.logging.Logger;  public class Sct {  public static void main(String[] args) {    try {  System.***out***.println("INS Practical performed by krunal dhavle ");  System.***out***.println("---simple column transposition ");  BufferedReader br = new BufferedReader(new InputStreamReader(System.***in***));  System.***out***.println("Enter your plain text");  String accept = br.readLine();  System.***out***.println("Enter of rows ");  int r = Integer.*parseInt*(br.readLine());  System.***out***.println("Enter the cols");  int c = Integer.*parseInt*(br.readLine());  int count = 0;  char table[][] = new char[r][c];  for (int i = 0; i < r; i++)  {  for (int j = 0; j < c; j++)  {  table[i][j] = accept.charAt(count);  count++;  }  }  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*(br.readLine());  }  String cipher = "", plain = "";  for (int j = 0; j < c; j++)  {  int k = choice[j];  for (int i = 0; i < r; i++)  {  cipher += table[i][k];  }  }  cipher = cipher.trim();  System.***out***.println("Cipher Text: "+cipher);  char mat[][] = new char[r][c];  int t = 0;  for (int j = 0; j < c; j++)  {  int k = choice[j];  for (int i = 0; i < r; i++)  {  mat[i][k] = cipher.charAt(t++);  }  }  for (int i = 0; i < r; i++)  {  for (int j = 0; j < c; j++)  {  plain += mat[i][j];  }  }  plain = plain.trim();  System.***out***.println("Plain text: "+plain);  }  catch (IOException ex)  {  Logger.*getLogger*(Sct.class.getName()).log(Level.***SEVERE***, null, ex); }}} |

**Output:**

