Practical 1

Aim: Write programs to implement the following Substitution Cipher Techniques: - Caesar Cipher , Monoalphabetic Cipher.

MonoAlphabeticCIpher

Code:

import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

public class MonoAlphabeticCipher {

    public static void main(String[] args) {

        Scanner sc = *new* Scanner(System.in);

        System.out.println("Enter plaintext:");

        String plaintext = sc.nextLine();

        String lower = "abcdefghijklmnopqrstuvwxyz";

        String upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        ArrayList<Integer> p = *new* ArrayList();

*for* (int i = 0; i < 26; i++)

            p.add(i);

*for* (int i = 0; i < 26; i++) {

            System.out.print(p.get(i) + "");

        }

        Collections.shuffle(p);

*for* (int i = 0; i < 26; i++) {

            System.out.print(p.get(i) + "");

        }

        String key = "", KEY = "";

*for* (int i = 0; i < 26; i++) {

            key += lower.charAt(p.get(i));

            key += upper.charAt(p.get(i));

        }

        String ciphertext = "";

        int i, j;

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

*for* (j = 0; j < lower.length(); j++) {

*if* (plaintext.charAt(i) == lower.charAt(j)) {

                    ciphertext += key.charAt(j);

*break*;

                }

*if* (plaintext.charAt(i) == upper.charAt(j)) {

                    ciphertext += KEY.charAt(j);

*break*;

                }

            }

*if* (j == upper.length())

                ciphertext += plaintext.charAt(i);

        }

        String decrtptedtext = "";

        i = 0;

        j = 0;

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

*for* (j = 0; j < key.length(); j++) {

*if* (ciphertext.charAt(i) == key.charAt(j)) {

                    decrtptedtext += lower.charAt(j);

*break*;

                }

*if* (ciphertext.charAt(i) == key.charAt(j)) {

                    decrtptedtext += upper.charAt(j);

*break*;

                }

            }

*if* (j == KEY.length())

                decrtptedtext += ciphertext.charAt(i);

        }

        System.out.println("nMonoalphabectic Cipher");

        System.out.println("plain text:" + plaintext);

        System.out.println("key        :" + key);

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

        System.out.println("Cipher Text  :" + ciphertext);

        System.out.println("Decrypted text:" + decrtptedtext);

    }

}

CaesarCipher

Code:

import java.util.Scanner;

public class CaesarCipher {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Input Data to encrypt");

        String str = sc.nextLine();

        System.out.println("Input the key length");

        int key = sc.nextInt();

        String encrypted = encrypt(str, key);

        System.out.println("Encryped Test is :" + encrypted);

        String decrypted = decrypt(encrypted, key);

        System.out.println("Decrypted:" + decrypted);

    }

    public static String encrypt(String str, int key) {

        String ct = "";

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

            int c = str.charAt(i);

            if (Character.isUpperCase(c)) {

                c = c + (key % 26);

                if (c > 'Z')

                    c = c - 26;

            } else if (Character.isLowerCase(c)) {

                c = c + (key % 26);

                if (c > 'z')

                    c = c - 26;

            }

            ct += (char) c;

        }

        return ct;

    }

    public static String decrypt(String str, int key) {

        String pt = "";

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

            int c = str.charAt(i);

            if (Character.isUpperCase(c)) {

                c = c - (key % 26);

                if (c < 'A')

                    c = c + 26;

            } else if (Character.isLowerCase(c)) {

                c = c - (key % 26);

                if (c < 'a')

                    c = c + 26;

            }

            pt += (char) c;

        }

        return pt;

    }

}

Output:

