**PLAYFAIR EXPERIMENT**

import java.awt.Point; //The Point class represents a location in a two-dimensional (x, y) coordinate space

import java.util.Scanner; //The Scanner class is a class in java.util which allows the user to read values of various types

public class PlayfairCipher {

private static char[][] charTable; //type char two-dimensional matrix created with name charTable

private static Point[] positions; //type point matrix created with matrix name as positions

public static void main(String[] args) { //MAIN METHOD

int i,j;

Scanner sc = new Scanner(System.in); //INPUT

//prompt method called

System.out.println("");

String key = prompt("Enter an encryption key (min length 6): ", sc, 6); // ENCRYPTION KEY ENTERED

System.out.println("");

String txt = prompt("Enter the message: ", sc, 1); //ENTER PLAIN-TEXT

System.out.println("");

String jti = prompt("Replace J with I? y/n: ", sc, 1); // REPLACE J WITH I IF J IS NOT PRESENT IN KEY ELSE OMIT Q IF I AND J PRESENT IN KEYWORD

System.out.println("");

boolean changeJtoI = jti.equalsIgnoreCase("y"); // COMPARES INPUT GIVEN IN STRING jti TO STRING IN PARAMETER 2 IGNORING CASE

createTable(key, changeJtoI);// createTable method called

String enc = encode(prepareText(txt, changeJtoI));//encode and prepareText method called

for(i=0;i<5;i++)//Printing the Table

{

for(j=0;j<5;j++)

{

System.out.print(charTable[i][j]);

System.out.print(" ");

}

System.out.println(" ");

}

String dec = decode(enc);// Decoding

System.out.printf("%nEncoded message: %n%s%n", enc); //Printing Encoded Text in digrams

System.out.printf("%nDecoded message: %n%s%n", dec); //Printing Decoded Text in digrams

dec = dec.replaceAll("\\s+","");

dec = dec.replaceAll("X","");

System.out.printf("%nDecoded message: %n%s%n", dec); //Printing Decoded Text removing Spaces and 'X'

}

//prompt method with 3 parameters

private static String prompt(String promptText, Scanner sc, int minLen) {

String s;

do {

System.out.print(promptText); //printing out the plain-text

s = sc.nextLine().trim(); // removing lagging and leading white spaces

} while (s.length() < minLen);

return s;

}

//prepareText method which takes input as plain-text and boolean value of changeJtoI

private static String prepareText(String s, boolean changeJtoI) {

s = s.toUpperCase().replaceAll("[^A-Z]", ""); //converting plain-text to upper-case and remove non-characters

return changeJtoI ? s.replace("J", "I") : s.replace("Q", ""); // If boolean value of JtoI is true replace J in the plain-text with I else omit Q

}

//createTable method with input as KEY and boolean value of changeJtoI

private static void createTable(String key, boolean changeJtoI) {

charTable = new char[5][5]; // 5\*5 matrix

positions = new Point[26]; //26-length array for position of each letter

String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ", changeJtoI);

int len = s.length();

for (int i = 0, k = 0; i < len; i++) {

char c = s.charAt(i);

if (positions[c - 'A'] == null) { //position of 'A' is not defined for the first time hence null

charTable[k / 5][k % 5] = c;

positions[c - 'A'] = new Point(k % 5, k / 5);

k++;

}

}

}

//encode method called with input as plain-text to encrypt the text

private static String encode(String s) {

String encrypt;

String text1 ="";

StringBuilder sb = new StringBuilder(s);

for (int i = 0; i < sb.length(); i += 2) {

if (i == sb.length() - 1)

sb.append(sb.length() % 2 == 1 ? 'X' : "");

else if (sb.charAt(i) == sb.charAt(i + 1))

sb.insert(i + 1, 'X');

}

encrypt = codec(sb, 1);

int length1 = encrypt.length();

for(int i=0;i < length1-1;i=i+2)

{

text1+= encrypt.substring(i,i+2); //printing Encrypted text as digrams

text1+=" ";

}

return text1;

}

private static String decode(String s) {

String decrypt;

String text2="";

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

decrypt = codec(new StringBuilder(s), 4); //printing Decrypted text as digrams

int length2 = decrypt.length();

for(int i=0;i < length2-1;i=i+2)

{

text2+= decrypt.substring(i,i+2);

text2+=" ";

}

return text2;

}

//codec method with input as altered text and another parameter as direction

private static String codec(StringBuilder text, int direction) {

int len = text.length();

for (int i = 0; i < len; i += 2) {

char a = text.charAt(i);

char b = text.charAt(i + 1);

int row1 = positions[a - 'A'].y;

int row2 = positions[b - 'A'].y;

int col1 = positions[a - 'A'].x;

int col2 = positions[b - 'A'].x;

if (row1 == row2) { //RULE 1: If present in same column then replace it with the corresponding next letter in same column

col1 = (col1 + direction) % 5;

col2 = (col2 + direction) % 5;

} else if (col1 == col2) { //RULE 2: If present in same row then replace it with the corresponding next letter in same row

row1 = (row1 + direction) % 5;

row2 = (row2 + direction) % 5;

} else {//RULE 3: Else draw a square and replace it with the corresponding opposite letter in the same row

int tmp = col1;

col1 = col2;

col2 = tmp;

}

text.setCharAt(i, charTable[row1][col1]);

text.setCharAt(i + 1, charTable[row2][col2]);

}

return text.toString();

}

}

