**Coding:**

import java.util.\*;

public class ColumnarCipher {

static final String encryptionKey = "BEST";

static Map<Character, Integer> keyMap = new HashMap<>();

static void setPermutationOrder() {

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

keyMap.put(encryptionKey.charAt(i), i);

}

}

static String encrypt(String plaintext) {

int rows, columns;

StringBuilder ciphertext = new StringBuilder();

columns = encryptionKey.length();

rows = (int) Math.ceil((double) plaintext.length() / columns);

char[][] matrix = new char[rows][columns];

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

for (int j = 0; j < columns; ) {

if (k < plaintext.length()) {

char ch = plaintext.charAt(k);

if (Character.isLetter(ch) || ch == ' ') {

matrix[i][j] = ch;

j++;

}

k++;

} else {

matrix[i][j] = '\_';

j++;

}

}

}

for (Map.Entry<Character, Integer> entry : keyMap.entrySet()) {

int columnIndex = entry.getValue();

for (int i = 0; i < rows; i++) {

if (Character.isLetter(matrix[i][columnIndex]) || matrix[i][columnIndex] == ' ' || matrix[i][columnIndex] == '\_') {

ciphertext.append(matrix[i][columnIndex]);

}

}

}

return ciphertext.toString();

}

static String decrypt(String ciphertext) {

int columns = encryptionKey.length();

int rows = (int) Math.ceil((double) ciphertext.length() / columns);

char[][] cipherMat = new char[rows][columns];

int k = 0;

for (int j = 0; j < columns; j++) {

for (int i = 0; i < rows; i++) {

if (k < ciphertext.length()) {

cipherMat[i][j] = ciphertext.charAt(k);

k++;

} else {

cipherMat[i][j] = '\_';

}

}

}

int index = 0;

for (Map.Entry<Character, Integer> entry : keyMap.entrySet()) {

entry.setValue(index++);

}

char[][] decCipher = new char[rows][columns];

for (int l = 0; l < encryptionKey.length(); l++) {

int columnIndex = keyMap.get(encryptionKey.charAt(l));

for (int i = 0; i < rows; i++) {

decCipher[i][l] = cipherMat[i][columnIndex];

}

}

StringBuilder msg = new StringBuilder();

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++) {

if (decCipher[i][j] != '\_') {

msg.append(decCipher[i][j]);

}

}

}

return msg.toString();

}

public static void main(String[] args) {

String plaintext = "This is a secret message.";

setPermutationOrder();

String ciphertext = encrypt(plaintext);

System.out.println("The Encrypted Message: " + ciphertext);

System.out.println("The Decrypted Message: " + decrypt(ciphertext));

}

}

**Output:**

