

Transposition Cipher

```
import java.io.*;
import java.util.*;

// Class
// For transposition cipher
public class Tr {

    // Member variables of this class
    public static String selectedKey;
    public static char sortedKey[];
    public static int sortedKeyPos[];

    // Constructor 1 of this class
    // Default constructor defining the default key
    public Tr() {
        selectedKey = "megabuck";
        sortedKeyPos = new int[selectedKey.length()];
        sortedKey = selectedKey.toCharArray();
    }

    // Constructor 2 of this class
    // Parameterized constructor defining the custom key
    public Tr(String mykey) {
        selectedKey = mykey;
        sortedKeyPos = new int[selectedKey.length()];
        sortedKey = selectedKey.toCharArray();
    }

    // Method 1 - doProcessOnKey()
    // To reorder data do the sorting on selected key
    public static void doProcessOnKey() {
        // Find position of each character in selected key
        // and arranging it in alphabetical order
        int min, i, j;
        char orginalKey[] = selectedKey.toCharArray();
        char temp;

        // Step 1: Sorting the array of selected key
        // using nested for loops
        for (i = 0; i < selectedKey.length(); i++) {
            min = i;
            for (j = i; j < selectedKey.length(); j++) {
                if (sortedKey[min] > sortedKey[j]) {
                    min = j;
                }
            }

            if (min != i) {
                temp = sortedKey[i];
                sortedKey[i] = sortedKey[min];
                sortedKey[min] = temp;
            }
        }

        // Step 2: Filling the position of array
        // according to alphabetical order
```

```

        // using nested for loops
        for (i = 0; i < selectedKey.length(); i++) {
            for (j = 0; j < selectedKey.length(); j++) {
                if (originalKey[i] == sortedKey[j])
                    sortedKeyPos[i] = j;
            }
        }
    }

// Method 2 - doEncryption()
// To encrypt the targeted string
public static String doEncryption(String plainText) {
    int min, i, j;
    char originalKey[] = selectedKey.toCharArray();
    char temp;
    doProcessOnKey();

    // Step 3: Generating the encrypted message by
    // doing encryption using Transposition Cipher
    int row = plainText.length() / selectedKey.length();
    int extrabit
        = plainText.length() % selectedKey.length();
    int exrow = (extrabit == 0) ? 0 : 1;
    int rowtemp = -1, coltemp = -1;
    int totallen = (row + exrow) * selectedKey.length();
    char pmat[][] = new char[(row + exrow)]
        [(selectedKey.length())];
    char encry[] = new char[totallen];

    int tempcnt = -1;
    row = 0;

    for (i = 0; i < totallen; i++) {
        coltemp++;
        if (i < plainText.length()) {
            if (coltemp == (selectedKey.length())) {
                row++;
                coltemp = 0;
            }
            pmat[row][coltemp] = plainText.charAt(i);
        } else {

            // Padding can be added between two
            // consecutive alphabets or a group of
            // alphabets of the resultant cipher text
            pmat[row][coltemp] = '-';
        }
    }

    int len = -1, k;

    for (i = 0; i < selectedKey.length(); i++) {
        for (k = 0; k < selectedKey.length(); k++) {
            if (i == sortedKeyPos[k]) {
                break;
            }
        }
        for (j = 0; j <= row; j++) {
            len++;

```

```

        encry[len] = pmat[j][k];
    }
}

String p1 = new String(encry);
return (new String(p1));
}

// Method 3 - doEncryption()
// To decrypt the targeted string
public static String doDecryption(String s) {
    int min, i, j, k;
    char key[] = selectedKey.toCharArray();
    char encry[] = s.toCharArray();
    char temp;

    doProcessOnKey();

    // Step 4: Generating a plain message
    int row = s.length();
    selectedKey.length();
    char pmat[][] = new char[row][(selectedKey.length())];
    int tempcnt = -1;

    for (i = 0; i < selectedKey.length(); i++) {
        for (k = 0; k < selectedKey.length(); k++) {
            if (i == sortedKeyPos[k]) {
                break;
            }
        }

        for (j = 0; j < row; j++) {
            tempcnt++;
            pmat[j][k] = encry[tempcnt];
        }
    }

    // Step 5: Storing matrix character in
    // to a single string
    char p1[] = new char[row * selectedKey.length()];

    k = 0;
    for (i = 0; i < row; i++) {
        for (j = 0; j < selectedKey.length(); j++) {
            if (pmat[i][j] != '*') {
                p1[k++] = pmat[i][j];
            }
        }
    }

    p1[k++] = '\\0';
    return (new String(p1));
}

@SuppressWarnings("static-access")

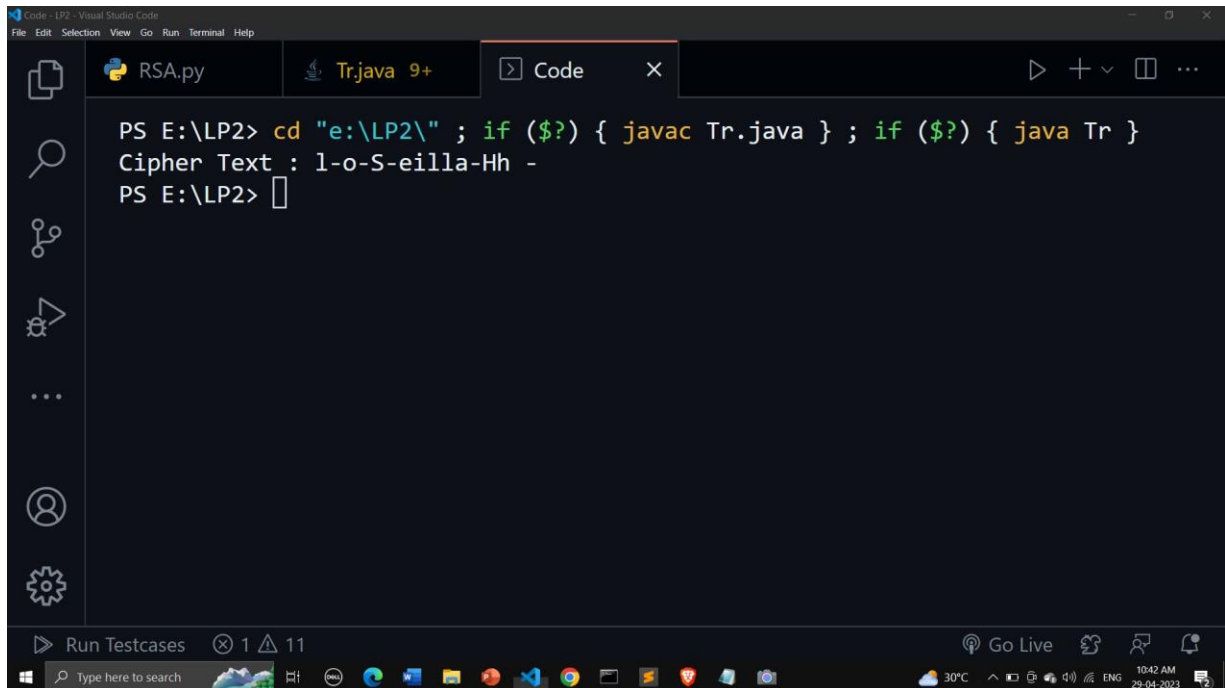
// Method 4 - main()
// Main driver method
public static void main(String[] args) {

```

```
// Creating object of class in main method
Tr tc = new Tr();

// Printing the ciphre text
// Custom input - Hello Geek
System.out.println("Cipher Text : " +
    tc.doEncryption("Hello Sahil"));
}
```

Output:



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the following commands and output:

```
PS E:\LP2> cd "e:\LP2\" ; if ($?) { javac Tr.java } ; if ($?) { java Tr }
Cipher Text : l-o-S-eilla-Hh -
PS E:\LP2>
```

The terminal output shows the command prompt navigating to the directory "e:\LP2\" and running the Java compilation and execution commands. The output of the program is "Cipher Text : l-o-S-eilla-Hh -".