```java
import java.util.*;

public class RowColumnarTransposition {
    static String key = "HACK";
    static String plainText = "";
    static Map<Character, Integer> keyMap = new HashMap<>();

    static void setPermuationOrder() {
        for(int i = 0; i < key.length(); i++) {
            keyMap.put(key.charAt(i), i);
        }
    }

    static String encrypt(String message) {
        if(message.isEmpty() || key.isEmpty()) {
            return "Please make sure you have a plain text and a key.";
        }

        StringBuilder cipher = new StringBuilder();
        int col = key.length(), row = (int) Math.ceil((double) message.length() / col);
        char[][] matrix = new char[row][col];

        for(int i = 0, k = 0; i < row; i++) {
            for(int j = 0; j < col;) {
                if(k < message.length()) {
                    char ch = message.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 colIndex = entry.getValue();
            for(int i = 0; i < row; i++) {
                if(Character.isLetter(matrix[i][colIndex]) || matrix[i][colIndex] == ' ' ||
                    matrix[i][colIndex] == '_') {
                    cipher.append(matrix[i][colIndex]);
                }
            }
        }

        return cipher.toString();
    }

    static String decryptMessage(String cipher) {
        if(cipher.isEmpty() || key.isEmpty()) {
            return "Please make sure you have a cipher text and a key.";
        }

        int col = key.length(), row = (int) Math.ceil((double) cipher.length() / col);
        char[][] cipherMat = new char[row][col];
        int k = 0;
        for(int j = 0; j < col; j++) {
            for(int i = 0; i < row; i++) {
                cipherMat[i][j] = cipher.charAt(k);
                k++;
            }
        }

        int index = 0;
        for(Map.Entry<Character, Integer> entry: keyMap.entrySet())
            entry.setValue(index++);
        char[][] decCipher = new char[row][col];
        for(int l = 0; l < key.length(); l++) {
            int colIndex = keyMap.get(key.charAt(l));
            for(int i = 0; i < row; i++) {
                decCipher[i][l] = cipherMat[i][colIndex];
            }
        }

        StringBuilder msg = new StringBuilder();
        for(int i = 0; i < row; i++) {
            for(int j = 0; j < col; j++) {
                if(decCipher[i][j] ≠ '_') msg.append(decCipher[i][j]);
            }
        }

        return msg.toString();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String cipher = "";
        while(true) {
            System.out.println("Choose an option:");
            System.out.println("1) Insert plain text");
            System.out.println("2) Generate key");
            System.out.println("3) Encrypt");
            System.out.println("4) Decrypt");
            System.out.println("5) Exit");
            int option = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch(option) {
                case 1:
                    System.out.println("Enter plain text:");
                    plainText = scanner.nextLine();
                    break;
                case 2:
                    System.out.println("Enter key:");
                    key = scanner.nextLine();
                    setPermuationOrder();
                    break;
                case 3:
                    System.out.println("Encrypting...");
                    cipher = encrypt(plainText);
                    System.out.println("Cipher text: " + cipher);
                    break;
                case 4:
                    System.out.println("Decrypting...");
                    String decryptedMessage = decryptMessage(cipher);
                    System.out.println("Plain Text: " + decryptedMessage);
                    break;
                case 5:
                    System.out.println("Exiting...");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid option. Please choose a valid option.");
            }
        }
    }
};
```

```
/*
-------------------------------------------------------------------------------------------------
OUTPUT
------------------------------------------------------------------------------------------------
Choose an option:
1) Insert plain text
2) Generate key
3) Encrypt
4) Decrypt
5) Exit

1

Enter plain text:
Geeks for Geeks

Choose an option:
1) Insert plain text
2) Generate key
3) Encrypt
4) Decrypt
5) Exit

2

Enter key:
HACK

Choose an option:
1) Insert plain text
2) Generate key
3) Encrypt
4) Decrypt
5) Exit

3

Encrypting...
Cipher text: e  kefGsGsrekoe_
Choose an option:
1) Insert plain text
2) Generate key
3) Encrypt
4) Decrypt
5) Exit

4

Decrypting...
Plain Text: Geeks for Geeks

Choose an option:
1) Insert plain text
2) Generate key
3) Encrypt
4) Decrypt
5) Exit

5

Exiting...


 */
```