```
import java.util.*;

public class ADFGVXEncryptor {

 static final char[] HEADERS = {'A', 'D', 'F', 'G', 'V', 'X'};

 static final String ALPHANUM = "abcdefghijklmnopqrstuvwxyz0123456789";    // إزالة التكرار من المفتاح الأول

  public static String removeDuplicates(String key) {

  StringBuilder result = new StringBuilder();

for (char c : key.toCharArray()) {

  if (result.indexOf(String.valueOf(c)) == -1)

        result.append(c);       }

return result.toString();    }    // توليد مصفوفة التشفير 6×6    public static char[][] buildMatrix(String key) {

  key = removeDuplicates(key.toLowerCase().replaceAll("[^a-z0-9]", ""));

 StringBuilder fullKey = new StringBuilder(key);

 for (char c : ALPHANUM.toCharArray()) {

     if (fullKey.indexOf(String.valueOf(c)) == -1)

fullKey.append(c);       }

 char[][] matrix = new char[6][6];

 int index = 0;

  for (int i = 0; i < 6; i++)

       for (int j = 0; j < 6; j++)

  matrix[i][j] = fullKey.charAt(index++);

 return matrix;    }    // البحث عن موقع الحرف في المصفوفة

   public static int[] findPosition(char[][] matrix, char c) {

 for (int i = 0; i < 6; i++)

 for (int j = 0; j < 6; j++)

   if (matrix[i][j] == c)

   return new int[]{i, j};

     return new int[]{-1, -1};    }    // تحويل النص إلى أزواج من ADFGVX

  public static String encode(String plainText, char[][] matrix) {

    plainText = plainText.toLowerCase().replaceAll("[^a-z0-9]", "");
```

```
  StringBuilder cipher = new StringBuilder();
 for (char c : plainText.toCharArray()) {
  int[] pos = findPosition(matrix, c);
 cipher.append(HEADERS[pos[0]]).append(HEADERS[pos[1]]);       }
    return cipher.toString();    }    // (عموديًا) توزيع الأحرف في جدول حسب المفتاح الثاني)
public static char[][] distribute(String cipherText, String key2) {
   int rows = key2.length();
      int cols = (int) Math.ceil((double) cipherText.length() / rows);
char[][] grid = new char[rows][cols
int index = 0;       for (int j = 0; j < cols; j++)
  for (int i = 0; i < rows; i++)
 grid[i][j] = (index < cipherText.length()) ? cipherText.charAt(index++) : 'X';
    return grid;    }    // ترتيب الأسطر حسب الترتيب الأبجدي للمفتاح الثاني
 public static String reorderByRows(char[][] grid, String key2) {
 key2 = key2.toLowerCase();
  int[] indexing = new int[26];
    Arrays.fill(indexing, -1);
 for (int i = 0; i < key2.length(); i++)
indexing[key2.charAt(i) - 'a'] = i;
 int cols = grid[0].length;
 StringBuilder result = new StringBuilder();
 for (int i = 0; i < 26; i++) {
 if (indexing[i] != -1) {
      int row = indexing[i];
      for (int j = 0; j < cols; j++)
 result.append(grid[row][j]);          }       }
  return result.toString();    }
 public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
   System.out.print("أدخل النص الأصلي: ");
```

```
   String plainText = sc.nextLine();
  System.out.print("أدخل المفتاح الأول: ");
 String key1 = sc.nextLine();
   System.out.print("أدخل المفتاح الثاني: ");
   String key2 = sc.nextLine();
   char[][] matrix = buildMatrix(key1);String cipherStep1 = encode(plainText,
matrix);
  char[][] grid = distribute(cipherStep1, key2);
   String finalCipher = reorderByRows(grid, key2);
 System.out.println("النص المشفّر النهائي هو: " + finalCipher);}}
```