```java
import java.io.IOException;

public class KeyboardArranger {
    public static char[] keyboard =
        { 'q', 'a', 'z', 'w', 's', 'x', 'e', 'd', 'c', 'r', 'f', 'v', 't', 'g', 'b',
 'y', 'h', 'n', 'u', 'j', 'm', 'i', 'k', 'o', 'l', 'p' };
    public static char[] referenceKeyboard =
        { 'q', 'a', 'z', 'w', 's', 'x', 'e', 'd', 'c', 'r', 'f', 'v', 't', 'g', 'b',
 'y', 'h', 'n', 'u', 'j', 'm', 'i', 'k', 'o', 'l', 'p' };
    private static int swappedLetterL1 = 0;
    private static int swappedLetterL2 = 0;

    private static int swappedLetterR1 = 15;
    private static int swappedLetterR2 = 15;

    /**
     * Multiplies the corresponding indexes in the digraphs and frequencies arrays
     * and sums the products to return the cost.
     *
     * @return the cost of the keyboard. The cost is essentially a digraph time
     *         average that is weighted based on letter pair frequencies.
     * @throws IOException if files are not found and the digraphs and frequencies
     *                     arrays cannot be organized.
     */
    public static double returnCost() throws IOException {
        double cost = 0;
        double[][] digraphs = Digraphs.getDigraphs();
        double[][] frequencies = Frequencies.getFrequencies();
        for (int i = 0; i < 26; i++) {
            for (int j = 0; j < 26; j++) {
                cost += digraphs[i][j] * frequencies[i][j];
            }
        }
        return cost/100.0;
    }

    /**
     * Swaps one key (intex swappedLetter1 in the keyboard array) with another
     * (index swappedLetter2). If called multiple times, the indexes of
     * swappedLetter1 and swappedLetter2 will change, thus swapping different keys.
     *
     * @return the swapped keyboard.
     */
    public static void keyboardSwap() {

        if (Digraphs.sheetNumber == 5 || Digraphs.sheetNumber == 6) {
            int L1 = swappedLetterL1;
            int L2 = swappedLetterL2;
            char[] rearranged = keyboard.clone();
            char storage = keyboard[L1];
            rearranged[L1] = rearranged[L2];
            rearranged[L2] = storage;

            if (L1 == 13 && L2 == 14) {
                swappedLetterL1 = 0;
                swappedLetterL2 = 0;
            } else if (L2 == 14) {
                swappedLetterL1++;
                swappedLetterL2 = swappedLetterL1 + 1;
```

```
58              } else
59                  swappedLetterL2++;
60              referenceKeyboard = rearranged;
61          }
62          else {
63              int R1 = swappedLetterR1;
64              int R2 = swappedLetterR2;
65
66              char[] rearranged = keyboard.clone();
67              char storage = keyboard[R1];
68              rearranged[R1] = rearranged[R2];
69              rearranged[R2] = storage;
70
71              if (R1 == keyboard.length - 2 && R2 == keyboard.length - 1) {
72                  swappedLetterR1 = 15;
73                  swappedLetterR2 = 15;
74              } else if (R2 == keyboard.length - 1) {
75                  swappedLetterR1++;
76                  swappedLetterR2 = swappedLetterR1 + 1;
77              } else
78                  swappedLetterR2++;
79              referenceKeyboard = rearranged;
80          }
81      }
82  }
83
```