```
 1 package foo;
 2
 3 /**
 4  * Überschrift:   Permutations
 5  * Beschreibung:  Example of an application that cannot be solved by
 6  *                iteration, recursion is necessary because you don't
 7  *                know how many nested loops you need
 8  * Copyright:     Copyright (c) 2003-04-14
 9  * Organisation:  FHTW
10  * @author        Debora Weber-Wulff
11  * @version       1.16
12  */
13
14 public class Perms {
15
16   static int count;
17
18   // Return the string s without the character at position i
19   public static String remove (String s, int i)
20   {
21    // check if string is long enough
22    int len = s.length();
23    if (i >= len || len == 1) return "";
24
25    if (i == 0) {
26       // i is the first character
27       return s.substring (1);
28    }
29    else {
30       if (i == len-1){
31          // i is the last character
32          // Note that s.substring (i,j) is the string that
33          // begins in position i and extends to j-1. Isn't Java fun!
34
35          return s.substring (0, len-1);
36       }
37       else {
38          // is is in the middle somewhere
39          return s.substring (0,i) + s.substring(i+1,len);
40       }
41    }
42   }
43
```

```
44    // This is the recursive method for constructing the permutations.
45    // We construct all permutations for rest and prepend prefix to them
46    public static void writePerms (String prefix,
47                                   String rest)
48    {
49      int len = rest.length();
50
51      // If rest is empty, we have a permutation, so print it!
52      if (len == 0) {
53         count++;
54         System.out.println (prefix);
55      }
56      else {
57         for (int i=0; i < len; i++) {
58             writePerms (prefix + rest.charAt(i),
59                         remove (rest, i));
60         }
61      }
62    }
63
64    public static void main (String[] args)
65    {
66      System.out.println ("Permutations of " + args[0]);
67      count = 0;
68      writePerms ("", args[0]);
69      System.out.println ("We found " + count + " permutations.");
70    }
71  }
```