```java
import java.util.Comparator;

import components.queue.Queue;
import components.queue.Queue1L;

/**
 *
 * @author Vaishnavi Kasabwala
 *
 */
public final class HW18 {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private HW18() {
    }

    /**
     * Removes and returns the minimum value from {@code q} according to the
     * ordering provided by the {@code compare} method from {@code order}.
     *
     * @param q
     *            the queue
     * @param order
     *            ordering by which to compare entries
     * @return the minimum value from {@code q}
     * @updates q
     * @requires <pre>
     * q /= empty_string  and
     *  [the relation computed by order.compare is a total preorder]
     * </pre>
     * @ensures <pre>
     * perms(q * <removeMin>, #q)  and
     *  for all x: string of character
     *      where (x is in entries (q))
     *    ([relation computed by order.compare method](removeMin, x))
     * </pre>
     */
    private static String removeMin(Queue<String> q, Comparator<String> order) {
        String min = null;

        for (String s : q) {
            if (min == null || order.compare(s, min) < 0) {
                min = s;
            }
        }

        Queue<String> temp = new Queue1L<>();

        for (String s : q) {
            if (s != min) {
                temp.enqueue(s);
            }
        }
        q.transferFrom(temp);

```

```
58          return min;
59      }
60
61      /**
62       * Sorts {@code q} according to the ordering provided by the {@code compare}
63       * method from {@code order}.
64       *
65       * @param q
66       *            the queue
67       * @param order
68       *            ordering by which to sort
69       * @updates q
70       * @requires [the relation computed by order.compare is a total preorder]
71       * @ensures q = [#q ordered by the relation computed by order.compare]
72       */
73      public static void sort(Queue<String> q, Comparator<String> order) {
74          Queue<String> temp = new Queue1L<>();
75
76          int length = q.length();
77          while (length > 0) {
78              temp.enqueue(removeMin(q, order));
79              length--;
80          }
81
82          q.transferFrom(temp);
83      }
84
85  }
86
```