```java
import java.util.Comparator;

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

/**
 * {@code Queue} represented as a {@code Sequence} of entries, with
 * implementations of primary methods.
 *
 * @param <T>
 *            type of {@code Queue} entries
 * @correspondence this = $this.entries
 */
public class HelloWorld {

    /**
     * Inserts the given {@code T} in the {@code Queue<T>} sorted according to
     * the given {@code Comparator<T>} and maintains the {@code Queue<T>}
     * sorted.
     *
     * @param <T>
     *            type of {@code Queue} entries
     * @param q
     *            the {@code Queue} to insert into
     * @param x
     *            the {@code T} to insert
     * @param order
     *            the {@code Comparator} defining the order for {@code T}
     * @updates q
     * @requires <pre>
     * IS_TOTAL_PREORDER([relation computed by order.compare method])  and
     * IS_SORTED(q, [relation computed by order.compare method])
     * </pre>
     * @ensures <pre>
     * perms(q, #q * <x>)  and
     * IS_SORTED(q, [relation computed by order.compare method])
     * </pre>
     */
    private static <T> void insertInOrder(Queue<T> q, T x,
            Comparator<T> order) {
```

```java
        int i = 0;
        Queue<T> result = q.newInstance();
        while (order.compare(x, q.front()) < 0 && i < q.length()) {
            result.enqueue(q.dequeue());
            i++;
        }

        result.enqueue(x);
        result.append(q);
        q.transferFrom(result);
    }

    /**
     * Sorts {@code this} according to the ordering provided by the
     * {@code compare} method from {@code order}.
     *
     * @param order
     *            ordering by which to sort
     * @updates this
     * @requires IS_TOTAL_PREORDER([relation computed by
 order.compare method])
     * @ensures <pre>
     * perms(this, #this)  and
     * IS_SORTED(this, [relation computed by order.compare method])
     * </pre>
     */
    public void sort(Comparator<T> order) {
        Queue<T> temp = new Queue1L<T>();
        while (this.length > 0) {
            insertInOrder(temp, this.deque(), order);
        }

        this.transferFrom(temp);

    }

}
```