```java
import java.util.Comparator;

/**
 * {@code QueueKernel} enhanced with secondary methods.
 *
 * @param <T>
 *            type of {@code Queue} entries
 * @mathdefinitions <pre>
 * IS_TOTAL_PREORDER (
 *   r: binary relation on T
 *  ) : boolean is
 *  for all x, y, z: T
 *   ((r(x, y) or r(y, x))  and
 *    (if (r(x, y) and r(y, z)) then r(x, z)))
 *
 * IS_SORTED (
 *   s: string of T,
 *   r: binary relation on T
 *  ) : boolean is
 *  for all x, y: T where (<x, y> is substring of s) (r(x, y))
 * </pre>
 */
public interface WaitingLine<T> extends WaitingLineKernel<T> {

    /**
     * Reports the front of {@code this}.
     *
     * @return the front entry of {@code this}
     * @aliases reference returned by {@code front}
     * @requires this /= <>
     * @ensures <front> is prefix of this
     */
    T front();

    /**
     * Replaces the front of {@code this} with {@code x}, and returns the old
     * front.
     *
     * @param x
     *            the new front entry
     * @return the old front entry
     * @aliases reference {@code x}
     * @updates this
     * @requires this /= <>
     * @ensures <pre>
     * <replaceFront> is prefix of #this  and
     * this = <x> * #this[1, |#this|)
     * </pre>
     */
    T replaceFront(T x);

    /**
     * Concatenates ("appends") {@code q} to the end of {@code this}.
     *
     * @param q
     *            the {@code Queue} to be appended to the end of {@code this}
     * @updates this
     * @clears q
     * @ensures this = #this * #q
```

```
60      */
61     void append(WaitingLine<T> q);
62
63     /**
64      * Reverses ("flips") {@code this}.
65      *
66      * @updates this
67      * @ensures this = rev(#this)
68      */
69     void flip();
70
71     /**
72      * Sorts {@code this} according to the ordering provided by the
73      * {@code compare} method from {@code order}.
74      *
75      * @param order
76      *            ordering by which to sort
77      * @updates this
78      * @requires IS_TOTAL_PREORDER([relation computed by order.compare method])
79      * @ensures <pre>
80      * perms(this, #this)  and
81      * IS_SORTED(this, [relation computed by order.compare method])
82      * </pre>
83      */
84     void sort(Comparator<T> order);
85
86     /**
87      * Rotates {@code this}.
88      *
89      * @param distance
90      *            distance by which to rotate
91      * @updates this
92      * @ensures <pre>
93      * if #this = <> then
94      *  this = #this
95      * else
96      *  this = #this[distance mod |#this|, |#this|) * #this[0, distance mod |#this|)
97      * </pre>
98      */
99     void rotate(int distance);
100
101 }
```