```java
import java.util.Comparator;
import java.util.Iterator;

/**
 * Layered implementations of secondary methods for {@code WaitingLine}.
 *
 * <p>
 * Assuming execution-time performance of O(1) for method {@code iterator} and
 * its return value's method {@code next}, execution-time performance of
 * {@code front} as implemented in this class is O(1). Execution-time
 * performance of {@code replaceFront} and {@code flip} as implemented in this
 * class is O(|{@code this}|). Execution-time performance of {@code append} as
 * implemented in this class is O(|{@code q}|). Execution-time performance of
 * {@code sort} as implemented in this class is O(|{@code this}| log
 * |{@code this}|) expected, O(|{@code this}|^2) worst case. Execution-time
 * performance of {@code rotate} as implemented in this class is
 * O({@code distance} mod |{@code this}|).
 *
 * @param <T>
 *            type of {@code WaitingLine} entries
 */
public abstract class WaitingLineSecondary<T> implements WaitingLine<T> {

    /*
     * Private members --------------------------------------------------------
     */

    /*
     * 2221/2231 assignment code deleted.
     */

    /*
     * Public members ---------------------------------------------------------
     */

    /*
     * Common methods (from Object) -------------------------------------------
     */

    @Override
    public final boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (!(obj instanceof WaitingLine<?>)) {
            return false;
        }
        WaitingLine<?> q = (WaitingLine<?>) obj;
        if (this.length() != q.length()) {
            return false;
        }
        Iterator<T> it1 = this.iterator();
        Iterator<?> it2 = q.iterator();
        while (it1.hasNext()) {
            T x1 = it1.next();
            Object x2 = it2.next();
            if (!x1.equals(x2)) {
                return false;
            }
        }
        return true;
    }

    // CHECKSTYLE: ALLOW THIS METHOD TO BE OVERRIDDEN
    @Override
    public int hashCode() {
        final int samples = 2;
        final int a = 37;
        final int b = 17;
        int result = 0;
        /*
         * This code makes hashCode run in O(1) time. It works because of the
         * iterator order string specification, which guarantees that the (at
         * most) samples entries returned by the it.next() calls are the same
         * when the two WaitingLines are equal.
         */
        int n = 0;
        Iterator<T> it = this.iterator();
        while (n < samples && it.hasNext()) {
            n++;
            T x = it.next();
            result = a * result + b * x.hashCode();
        }
        return result;
    }

    // CHECKSTYLE: ALLOW THIS METHOD TO BE OVERRIDDEN
    @Override
    public String toString() {
        StringBuilder result = new StringBuilder("<");
        Iterator<T> it = this.iterator();
        while (it.hasNext()) {
            result.append(it.next());
            if (it.hasNext()) {
                result.append(",");
            }
        }
        result.append(">");
        return result.toString();
    }

    /*
     * Other non-kernel methods -----------------------------------------------
     */

    // CHECKSTYLE: ALLOW THIS METHOD TO BE OVERRIDDEN
    @Override
    public T front() {
        assert this.length() > 0 : "Violation of: this /= <>";
        // return statement line to avoid error
        return null;

        /*
         * 2221/2231 assignment code deleted.
         */
    }

    // CHECKSTYLE: ALLOW THIS METHOD TO BE OVERRIDDEN
    @Override
    public T replaceFront(T x) {
        assert this.length() > 0 : "Violation of: this /= <>";
        // return statement line to avoid error
        return x;

        /*
         * 2221/2231 assignment code deleted.
         */
    }

    // CHECKSTYLE: ALLOW THIS METHOD TO BE OVERRIDDEN
    @Override
    public void append(WaitingLine<T> q) {
        assert q != null : "Violation of: q is not null";
        assert q != this : "Violation of: q is not this";

        /*
         * 2221/2231 assignment code deleted.
         */
    }

    // CHECKSTYLE: ALLOW THIS METHOD TO BE OVERRIDDEN
    @Override
    public void flip() {

        /*
         * 2221/2231 assignment code deleted.
         */
    }

    // CHECKSTYLE: ALLOW THIS METHOD TO BE OVERRIDDEN
    @Override
    public void sort(Comparator<T> order) {
        assert order != null : "Violation of: order is not null";

        /*
         * 2221/2231 assignment code deleted.
         */
    }

    // CHECKSTYLE: ALLOW THIS METHOD TO BE OVERRIDDEN
    @Override
    public void rotate(int distance) {

        /*
         * 2221/2231 assignment code deleted.
         */
    }

```

```java
}

```