```java
import java.util.Comparator;

import components.queue.Queue;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * HW15 sorting
 *
 * @author Sam Espanioly
 */
public final class orderCompare {

    /**
     * Default constructor--private to prevent instantiation.
     */
    private orderCompare() {
        // no code needed here
    }

    /**
     * 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) {
        // I do not understand how is this different than sort
    }

    /**
     * Sorts {@code this} according to the ordering provided by the
     * {@code compare} method from {@code order}.
     *
     * @param <T>
     *
     * @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 <T> void sort(Comparator<T> order) {
        // I need help understanding this assignment
    }

//    Statement   Variable Values
//    SortingMachine<Integer> sm = new SortingMachine1L<>(new IntegerGE());
//        sm = (true, >=, {})
//    sm.add(0);
//        sm = (true, >=, {0})
//    sm.add(2);
//        sm = (true, >=, {0, 2})
//    sm.add(-1);
//        sm = (true, >=, {0, 2, -1})
//    sm.changeToExtractionMode();
//        sm = (false, >=, {-1, 0, 2})
//    int i = sm.removeFirst();
//        sm =sm = (false, >=, {0, 2})
//        i = -1
//    sm.clear();
//        sm = (true, >=, {})
//        i = -1

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments; unused here
     */
    public static void main(String[] args) {
        SimpleWriter out = new SimpleWriter1L();
        out.close();
    }

}
```