```java
import java.util.Comparator;

/**
 * Program to sort lines from an input file in lexicographic order by using
 * selection sort on {@code Queue<String>}.
 *
 * @author Paolo Bucci
 */
public final class QueueSortMain {

    /**
     * Compare {@code String}s in lexicographic order.
     */
    private static class StringLT implements Comparator<String> {
        @Override
        public int compare(String o1, String o2) {
            return o1.compareTo(o2);
        }
    }

    /**
     * No-argument constructor--private to prevent instantiation.
     */
    private QueueSortMain() {
        // no code needed here
    }

    /**
     * @mathdefinitions <pre>
     * STRING_OF_LINES(s: string of character): string of string of character
     *  satisfies
     *   if s = empty_string
     *    then STRING_OF_LINES(s) = empty_string
     *    else if "\n" is not substring of s
     *     then STRING_OF_LINES(s) = <s>
     *     else there exists a, b: string of character
     *           (s = a * "\n" * b  and
     *            "\n" is not substring of a  and
     *            STRING_OF_LINES(s) = <a> * STRING_OF_LINES(b))
     * </pre>
     */

    /**
     * Gets one line at a time from {@code in} until end of input, and puts them
     * into the queue {@code q}.
     *
     * @param in
     *            the source of the lines to be input
     * @param q
     *            the queue of lines that are read
     * @updates in
     * @replaces q
     * @requires in.is_open
     * @ensures <pre>
     * in.is_open  and
     *  in.ext_name = #in.ext_name  and
     *  in.content = ""  and
```

```
64       *  lines = entries(STRING_OF_LINES(#in.content))
65       * </pre>
66       */
67      private static void getLinesFromInput(SimpleReader in, Queue<String> q) {
68          assert in != null : "Violation of: in is not null";
69          assert q != null : "Violation of: q is not null";
70          assert in.isOpen() : "Violation of: in.is_open";
71
72          q.clear();
73          while (!in.atEOS()) {
74              String str = in.nextLine();
75              q.enqueue(str);
76          }
77      }
78
79      /**
80       * Puts lines from {@code q}, one line at a time, onto {@code out}.
81       *
82       * @param out
83       *            the stream to receive output
84       * @param q
85       *            the queue of lines that are output
86       * @updates out
87       * @requires out.is_open
88       * @ensures <pre>
89       * out.is_open  and
90       *  out.ext_name = #out.ext_name  and
91       *  out.content = #out.content *
92       *   [the entries of q, in some order, with appropriate mark-up]
93       * </pre>
94       */
95      private static void putLinesToOutput(SimpleWriter out, Queue<String> q) {
96          assert out != null : "Violation of: out is not null";
97          assert q != null : "Violation of: q is not null";
98          assert out.isOpen() : "Violation of: out.is_open";
99
100         out.println();
101         out.println("--- Start of Queue<String> output (" + q.length()
102                 + " lines) ---");
103         for (String str : q) {
104             out.println(str);
105         }
106         out.println("--- End of Queue<String> output ---");
107     }
108
109     /**
110      * Main method.
111      *
112      * @param args
113      *            the command line arguments; unused here
114      */
115     public static void main(String[] args) {
116         SimpleReader in = new SimpleReader1L();
117         SimpleWriter out = new SimpleWriter1L();
118
119         /*
120          * Get input file name and open input stream
```

```
121          */
122         out.print("Enter an input file name: ");
123         String fileName = in.nextLine();
124         SimpleReader file = new SimpleReader1L(fileName);
125
126         /*
127          * Get lines from input and output them, unsorted
128          */
129         Queue<String> q = new Queue1LSort1();
130         getLinesFromInput(file, q);
131         putLinesToOutput(out, q);
132
133         /*
134          * Sort lines into non-decreasing lexicographic order
135          */
136         Comparator<String> cs = new StringLT();
137         q.sort(cs);
138
139         /*
140          * Output lines in sorted order
141          */
142         putLinesToOutput(out, q);
143
144         in.close();
145         out.close();
146     }
147
148 }
```