```java
import components.queue.Queue;
import components.queue.Queue2;
import components.simplereader.SimpleReader;

/**
 * {@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 {

    public static void main(String[] args) {

    }

    /**
     * Returns the first "word" (maximal length string of
characters not in
     * {@code SEPARATORS}) or "separator string" (maximal length
string of
     * characters in {@code SEPARATORS}) in the given {@code text}
starting at
     * the given {@code position}.
     *
     * @param text
     *            the {@code String} from which to get the word or
separator
     *            string
     * @param position
     *            the starting index
     * @return the first word or separator string found in {@code
text} starting
     *         at index {@code position}
     * @requires 0 <= position < |text|
     * @ensures <pre>
     * nextWordOrSeparator =
     *   text[position, position + |nextWordOrSeparator|)  and
     * if entries(text[position, position + 1)) intersection
entries(SEPARATORS) = {}
     * then
     *   entries(nextWordOrSeparator) intersection
```

```
   entries(SEPARATORS) = {}  and
39      *   (position + |nextWordOrSeparator| = |text|  or
40      *    entries(text[position, position + |nextWordOrSeparator| +
   1))
41      *      intersection entries(SEPARATORS) /= {})
42      * else
43      *   entries(nextWordOrSeparator) is subset of
   entries(SEPARATORS)  and
44      *   (position + |nextWordOrSeparator| = |text|  or
45      *    entries(text[position, position + |nextWordOrSeparator| +
   1))
46      *      is not subset of entries(SEPARATORS))
47      * </pre>
48      */
49     private static String nextWordOrSeparator(String text, int
   position) {
50         String result;
51         int secondIndex = position;
52         if (SEPARATORS.contains(text.charAt(position))) {
53             while (secondIndex < text.length()
54                     &&
   SEPARATORS.contains(text.charAt(secondIndex))) {
55                 secondIndex++;
56             }
57         } else {
58             while (secondIndex < text.length()
59                     && !
   SEPARATORS.contains(text.charAt(secondIndex))) {
60                 secondIndex++;
61             }
62         }
63         result = text.substring(position, secondIndex);
64
65         return result;
66     }
67
68     /**
69      * Tokenizes the entire input getting rid of all whitespace
   separators and
70      * returning the non-separator tokens in a {@code
   Queue<String>}.
71      *
72      * @param in
73      *            the input stream
```

```
74      * @return the queue of tokens
75      * @requires in.is_open
76      * @ensures <pre>
77      * tokens =
78      *   [the non-whitespace tokens in #in.content] *
  <END_OF_INPUT>  and
79      * in.content = <>
80      * </pre>
81      */
82     public static Queue<String> tokens(SimpleReader in) {
83         Queue<String> answer = new Queue2<String>();
84         while (!in.nextLine().equals(END_OF_INPUT)) {
85             String nextWord = nextWordOrSeparator(in.nextLine(),
  0);
86             if (!nextWord.contains(SEPARATORS)) {
87                 answer.enqueue(nextWord);
88             }
89         }
90         return answer;
91     }
92 }
```