```java
import components.set.Set;
import components.set.Set1L;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * Simple HelloWorld program (clear of Checkstyle and FindBugs warnings).
 *
 * @author Sam Espanioly
 */
public final class practice {

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

    /**
     * Generates the set of characters in the given {@code String} into the
     * given {@code Set}.
     *
     * @param str
     *            the given {@code String}
     * @param charSet
     *            the {@code Set} to be replaced
     * @replaces charSet
     * @ensures charSet = entries(str)
```

```
32      */
33     private static void generateElements(String str,
   Set<Character> charSet) {
34         if (charSet == null || str == null ||
   charSet.size() == 0) {
35         } else {
36             //new set
37             Set<Character> chars = new Set1L<>();
38             //seperate string into characters
39             for (char c : str.toCharArray()) {
40                 // add characters to the set
41                 if (!chars.contains(c)) {
42                     chars.add(c);
43                 }
44             }
45             //transfer set
46             charSet.transferFrom(chars);
47         }
48     }
49
50     /**
51      * Returns the first "word" (maximal length string of
   characters not in
52      * {@code separators}) or "separator string" (maximal
   length string of
53      * characters in {@code separators}) in the given
   {@code text} starting at
54      * the given {@code position}.
55      *
56      * @param text
57      *             the {@code String} from which to get the
   word or separator
58      *             string
59      * @param position
```

```
60      *            the starting index
61      * @param separators
62      *            the {@code Set} of separator characters
63      * @return the first word or separator string found in
  {@code text} starting
64      *         at index {@code position}
65      * @requires 0 <= position < |text|
66      * @ensures <pre>
67      * nextWordOrSeparator =
68      *   text[position, position + |nextWordOrSeparator|)
  and
69      * if entries(text[position, position + 1))
  intersection separators = {}
70      * then
71      *   entries(nextWordOrSeparator) intersection
  separators = {}  and
72      *   (position + |nextWordOrSeparator| = |text|  or
73      *    entries(text[position, position + |
  nextWordOrSeparator| + 1))
74      *      intersection separators /= {})
75      * else
76      *   entries(nextWordOrSeparator) is subset of
  separators  and
77      *   (position + |nextWordOrSeparator| = |text|  or
78      *    entries(text[position, position + |
  nextWordOrSeparator| + 1))
79      *      is not subset of separators)
80      * </pre>
81      */
82     private static String nextWordOrSeparator(String text,
  int position,
83             Set<Character> separators) {
84         assert position >= 0 && position < text
85                 .length() : "Violation of: position is
```

```
   within the bounds of text";
 86         String word = ""; // start with empty string
 87         int len = text.length();
 88         //another string to hold the separators
 89         String sep = "";
 90         for (char c : text.substring(position,
   len).toCharArray()) {
 91             sep += c;
 92             // if it does not include c then add c to the
   word
 93             if (!separators.contains(c)) {
 94                 word += c;
 95             } else {
 96                 //string of separators if charAt position
   is a separator
 97                 word = sep;
 98                 // this will cause it to break the loop or
   to crash but it will stop the for each loop
 99
100                 len = 0;
101             }
102         }
103         // return the word
104         return word;
105
106     }
107
108     /**
109      * Main method.
110      *
111      * @param args
112      *             the command line arguments; unused here
113      */
114     public static void main(String[] args) {
```

```
115         SimpleWriter out = new SimpleWriter1L();
116         SimpleReader in = new SimpleReader1L();
117
118         //ensuring that the code will work
119
120         //answers for the rest of the questions
121         out.close();
122         in.close();
123     }
124
125 }
126
```