```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;

/**
 * Program to test static methods {@code generateElements} and
 * {@code nextWordOrSeparator}.
 *
 * @author Put your name here
 *
 */
public final class NextWordOrSeparatorTest {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private NextWordOrSeparatorTest() {
    }

    /**
     * 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)
     */
    private static void generateElements(String str, Set<Character> charSet) {
        assert str != null : "Violation of: str is not null";
        assert charSet != null : "Violation of: charSet is not null";

        for (int i = 0; i < str.length(); i++) {
            char a = str.charAt(i);
            if (!charSet.contains(a)) {
                charSet.add(a);
            }
        }
    }

    /**
     * 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
     * @param separators
```

```
58       *            the {@code Set} of separator characters
59       * @return the first word or separator string found in {@code text} starting
60       *         at index {@code position}
61       * @requires 0 <= position < |text|
62       * @ensures <pre>
63       * nextWordOrSeparator =
64       *   text[position, position + |nextWordOrSeparator|)  and
65       * if entries(text[position, position + 1)) intersection separators = {}
66       * then
67       *   entries(nextWordOrSeparator) intersection separators = {}  and
68       *   (position + |nextWordOrSeparator| = |text|  or
69       *    entries(text[position, position + |nextWordOrSeparator| + 1))
70       *      intersection separators /= {})
71       * else
72       *   entries(nextWordOrSeparator) is subset of separators  and
73       *   (position + |nextWordOrSeparator| = |text|  or
74       *    entries(text[position, position + |nextWordOrSeparator| + 1))
75       *      is not subset of separators)
76       * </pre>
77       */
78      private static String nextWordOrSeparator(String text, int position,
79              Set<Character> separators) {
80          assert text != null : "Violation of: text is not null";
81          assert separators != null : "Violation of: separators is not null";
82          assert 0 <= position : "Violation of: 0 <= position";
83          assert position < text.length() : "Violation of: position < |text|";
84
85          String temp = text.substring(position);
86          int end = temp.length();
87
88          if (!separators.contains(temp.charAt(0))) {
89              int i = 1;
90              while (i < temp.length() && !separators.contains(temp.charAt(i))) {
91                  i++;
92              }
93              end = i;
94          } else {
95              int i = 1;
96              while (i < temp.length() && separators.contains(temp.charAt(i))) {
97                  i++;
98              }
99              end = i;
100         }
101         return temp.substring(0, end);
102     }
103
104     /**
105      * Main method.
106      *
107      * @param args
108      *            the command line arguments
109      */
110     public static void main(String[] args) {
111         /*
112          * Define separator characters for test
113          */
114         final String separatorStr = " \t, ";
```

```
115         Set<Character> separatorSet = new Set1L<>();
116         generateElements(separatorStr, separatorSet);
117         /*
118          * Open input and output streams
119          */
120         SimpleReader in = new SimpleReader1L();
121         SimpleWriter out = new SimpleWriter1L();
122         /*
123          * Ask for test cases
124          */
125         out.println();
126         out.print("New test case (y/n)? ");
127         String response = in.nextLine();
128         while (response.equals("y")) {
129             /*
130              * Output heading
131              */
132             out.print("Test case: ");
133             String testStr = in.nextLine();
134             out.println();
135             out.println("----Next test case----");
136             out.println();
137             /*
138              * Process test case
139              */
140             int position = 0;
141             while (position < testStr.length()) {
142                 String token = nextWordOrSeparator(testStr, position,
143                         separatorSet);
144                 if (separatorSet.contains(token.charAt(0))) {
145                     out.print("  Separator: <");
146                 } else {
147                     out.print("  Word: <");
148                 }
149                 out.println(token + ">");
150                 position += token.length();
151             }
152             /*
153              * Ask user whether to continue
154              */
155             out.println();
156             out.print("New test case (y/n)? ");
157             response = in.nextLine();
158         }
159         /*
160          * Close input and output streams
161          */
162         in.close();
163         out.close();
164     }
165 }
```