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

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

    /**
     * Default constructor--private to prevent instantiation.
     */
    private HelloWorld() {
        // 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)
     */
    private static void generateElements(String str, Set<Character> strSet) {
        if (strSet == null || str == null || strSet.size() == 0) {
            return;
        }
        //new set
        Set<Character> chars = new Set1L<>();
```

```
37          //seperate string into characters
38          for (char c : str.toCharArray()) {
39              // add characters to the set
40              chars.add(c);
41          } //transfer set
42          strSet.transferFrom(chars);
43      }
44
45      /**
46       * Returns the first "word" (maximal length string of
  characters not in
47       * {@code separators}) or "separator string" (maximal length
  string of
48       * characters in {@code separators}) in the given {@code text}
  starting at
49       * the given {@code position}.
50       *
51       * @param text
52       *            the {@code String} from which to get the word or
  separator
53       *            string
54       * @param position
55       *            the starting index
56       * @param separators
57       *            the {@code Set} of separator characters
58       * @return the first word or separator string found in {@code
  text} starting
59       *         at index {@code position}
60       * @requires 0 <= position < |text|
61       * @ensures <pre>
62       * nextWordOrSeparator =
63       *   text[position, position + |nextWordOrSeparator|)  and
64       * if entries(text[position, position + 1)) intersection
  separators = {}
65       * then
66       *   entries(nextWordOrSeparator) intersection separators = {}
  and
67       *   (position + |nextWordOrSeparator| = |text|  or
68       *    entries(text[position, position + |nextWordOrSeparator| +
```

```
1))
69      *      intersection separators /= {})
70      * else
71      *   entries(nextWordOrSeparator) is wordset of separators  and
72      *   (position + |nextWordOrSeparator| = |text|  or
73      *    entries(text[position, position + |nextWordOrSeparator| +
1))
74      *      is not subset of separators)
75      * </pre>
76      */
77     private static String nextWordOrSeparator(String text, int
position,
78             Set<Character> separators) {
79         assert position >= 0 && position < text
80                 .length() : "Violation of: position is within the
bounds of text";
81         String word = ""; // start with empty string
82         int len = text.length();
83         for (char c : text.substring(position, len).toCharArray())
{
84             // if it does not include c then add c to the word
85             if (!separators.contains(c)) {
86                 word += c;
87             } else {
88                 // this will cause it to break the loop
89                 len = 0;
90             }
91         }
92         // return the word
93         return word;
94     }
95
96     /**
97      * Main method.
98      *
99      * @param args
100     *            the command line arguments; unused here
101     */
102    public static void main(String[] args) {
```

```
103         SimpleWriter out = new SimpleWriter1L();
104         out.println("Hello World!");
105         out.close();
106     }
107
108 }
109
```