

```
1 import components.naturalnumber.NaturalNumber;
2 import components.naturalnumber.NaturalNumber2;
3 import components.simplewriter.SimpleWriter;
4 import components.simplewriter.SimpleWriter1L;
5
6 /**
7  * Simple HelloWorld program (clear of Checkstyle and FindBugs warnings).
8  *
9  * @author me
10 */
11 public final class HelloWorld {
12
13     /**
14      * Default constructor--private to prevent instantiation.
15      */
16     private HelloWorld() {
17         // no code needed here
18     }
19
20     /**
21      * Returns the number of digits of {@code n}.
22      *
23      * @param n
24      *         {@code NaturalNumber} whose digits to count
25      * @return the number of digits of {@code n}
26      * @ensures numberOfDigits = [number of digits of n]
27      */
28     private static int numberOfDigits(NaturalNumber n) {
29         String number = n.toString();
30         int i = number.length();
31         return i;
32     }
33
34     /**
35      * Returns the sum of the digits of {@code n}.
36      *
37      * @param n
38      *         {@code NaturalNumber} whose digits to add
39      * @return the sum of the digits of {@code n}
40      * @ensures sumOfDigits = [sum of the digits of n]
41      */
42     private static int sumOfDigits(NaturalNumber n) {
43         int sum = 0;
44         String number = n.toString();
45         int length = number.length();
46         int i = 0;
47         while (i < length-1) {
48             sum = sum + number.charAt(i);
49             ++i;
50         }
51         return sum;
52     }
53
54     /**
55      * Returns the sum of the digits of {@code n}.
56      *
57      * @param n
```

```

58      *      {@code NaturalNumber} whose digits to add
59      * @return the sum of the digits of {@code n}
60      * @ensures sumOfDigits = [sum of the digits of n]
61      */
62      private static NaturalNumber sumOfDigits2 NaturalNumber n {
63          NaturalNumber sum = new NaturalNumber2 sumOfDigits(n));
64          return sum;
65      }
66
67      /**
68       * Divides {@code n} by 2.
69       *
70       * @param n
71       *      {@code NaturalNumber} to be divided
72       * @updates n
73       * @ensures 2 * n <= #n < 2 * (n + 1)
74       */
75      private static void divideBy2 NaturalNumber n {
76          NaturalNumber t = new NaturalNumber2 2;
77          NaturalNumber r = n.divide(t);
78          int rr = r.toInt();
79          if (rr % 2 != 0) { // check if there's any more remainders
80              n.increment(); // add to n because we have to make n + 1
81          }
82      }
83
84      /**
85       * Checks whether a {@code String} is a palindrome.
86       *
87       * @param s
88       *      {@code String} to be checked
89       * @return true if {@code s} is a palindrome, false otherwise
90       * @ensures isPalindrome = (s = rev(s))
91       */
92      private static boolean isPalindrome String s {
93          String rev = "";
94          boolean kayak = false;
95          int len = s.length();
96          int i = 0;
97          while (i < len) {
98              // proceed only if it's not an empty character
99              if (s.charAt(i) != ' ') {
100                  rev = s.charAt(i) + rev;
101              }
102              ++i;
103          }
104          if (s.equals(rev)) {
105              kayak = true;
106          }
107          return kayak;
108      }
109
110      /**
111       * Main method.
112       *
113       * @param args
114       *      the command line arguments; unused here

```

```
115     */
116     public static void main(String[] args) {
117         SimpleWriter out = new SimpleWriter1L();
118         //out.println("Hello World!");
119
120         out.close();
121     }
122
123 }
124
```