

```
1 import components.naturalnumber.NaturalNumber;
2 import components.naturalnumber.NaturalNumber2;
3 import components.simplereader.SimpleReader;
4 import components.simplereader.SimpleReader1L;
5 import components.simplewriter.SimpleWriter;
6 import components.simplewriter.SimpleWriter1L;
7
8 /**
9  * Put a short phrase describing the program here.
10 *
11 * @author Sam Espanioly
12 *
13 */
14 public final class HW12 {
15
16     /**
17      * Private constructor so this utility class cannot be
18      * instantiated.
19      */
20     private HW12() {
21
22     }
23
24     /**Implement the static method declared as follows:
25     /**
26      * Returns the number of digits of {@code n}.
27      *
28      * @param n
29      *         {@code NaturalNumber} whose digits to
30      *         count
31      * @return the number of digits of {@code n}
32      * @ensures numberOfDigits = [number of digits of n]
33      */
34     private static int numberOfDigits(NaturalNumber n) {
35         int digit = 0;
36     }
37 }
```

```
33         int rem = n.divideBy10();
34         digit++;
35         if (!n.isZero()) {
36             // adds it after every recursion
37             digit = digit + numberOfDigits(n);
38
39         }
40         n.multiplyBy10(rem);
41         return digit;
42     }
43
44     //Implement the static method declared as follows:
45     /**
46      * Returns the sum of the digits of {@code n}.
47      *
48      * @param n
49      *         {@code NaturalNumber} whose digits to
50      *         add
51      * @return the sum of the digits of {@code n}
52      * @ensures sumOfDigits = [sum of the digits of n]
53      */
54     private static int sumOfDigits(NaturalNumber n) {
55         int remainder = n.divideBy10();
56
57         int sum = remainder;
58         if (!n.isZero()) {
59             //adds it after it returns it after calling it
60             sum = sum + sumOfDigits(n);
61         }
62         n.multiplyBy10(remainder);
63         return sum;
64     }
65
66     //In addition to the kernel methods, for this question
```

(only!) you are allowed to use the NaturalNumber method add. Implement the static method declared as follows:

```
66     /**
67      * Returns the sum of the digits of {@code n}.
68      *
69      * @param n
70      *      {@code NaturalNumber} whose digits to
add
71      * @return the sum of the digits of {@code n}
72      * @ensures sumOfDigits = [sum of the digits of n]
73      */
74     private static NaturalNumber
sumOfDigits2(NaturalNumber n) {
75         int temp = n.divideBy10();
76
77         NaturalNumber sum = new NaturalNumber2(temp);
78         if (!n.isZero()) {
79             sum.add(sumOfDigits2(n));
80             //sum = sum + sumOfDigits(n);
81         }
82         n.multiplyBy10(temp);
83         return sum;
84     }
85
86     //Implement the static method declared as follows:
87     /**
88      * Divides {@code n} by 2.
89      *
90      * @param n
91      *      {@code NaturalNumber} to be divided
92      * @updates n
93      * @ensures 2 * n <= #n < 2 * (n + 1)
94      */
95     private static void divideBy2(NaturalNumber n) {
```

```

96         int rem = n.divideBy10();
97         if (!n.isZero()) {
98             // get next remainder
99             int nextRem = n.divideBy10();
100            //check if its even
101            if (nextRem % 2 == 0) {
102                rem = rem / 2;
103            } else { //else its odd
104                // add 5 in case next decimal is odd
105                rem = rem / 2 + 5;
106            }
107            // added this so it divides the last digit
without trouble
108            if (n.isZero()) {
109                nextRem /= 2;
110            }
111            n.multiplyBy10(nextRem);
112            divideBy2(n);
113
114        }
115
116        n.multiplyBy10(rem);
117    }
118
119    //Implement the static method declared as follows:
120    /**
121     * Checks whether a {@code String} is a palindrome.
122     *
123     * @param s
124     *         {@code String} to be checked
125     * @return true if {@code s} is a palindrome, false
otherwise
126     * @ensures isPalindrome = (s = rev(s))
127     */
```

```
128     private static boolean isPalindrome(String s) {
129         boolean check = true;
130         int i = 0; //first char
131         int len = s.length();
132         int u = len - 1; // last char
133         // len / 2 + 1 to make it more efficient
134         while (i < ((len / 2) + 1)) {
135             //to avoid spaces
136             if (s.charAt(i) == ' ') {
137                 i++;
138             }
139             //to avoid spaces
140             if (s.charAt(u) == ' ') {
141                 u--;
142             }
143             // compare them
144             if (s.charAt(i) != s.charAt(u)) {
145                 check = false;
146                 i = len; // added this to make the loop
quit fast== more efficient
147             }
148             u--;
149             i++;
150         }
151         return check;
152     }
153
154     /**
155      * Main method.
156      *
157      * @param args
158      *         the command line arguments
159      */
160     public static void main(String[] args) {
```

```
161     SimpleReader in = new SimpleReader1L();
162     SimpleWriter out = new SimpleWriter1L();
163     //testing methods
164     out.println("Enter n: ");
165     NaturalNumber n = new
        NaturalNumber2(in.nextLine());
166     out.println(numberOfDigits(n));
167     out.println(n); // check for restore n
168     out.println(sumOfDigits(n));
169     out.println(n); // check for restore n
170     out.println(sumOfDigits2(n));
171     out.println(n); // check for restore n
172     divideBy2(n);
173     out.println("divide by 2: " + n);
174     //testing methods
175     String s = "racecar";
176     out.println(isPalindrome(s));
177     s = "r ace c ar";
178     out.println(isPalindrome(s));
179     s = "not a racecar";
180     out.println(isPalindrome(s));
181     in.close();
182     out.close();
183 }
184 }
185
```