

```
1 import components.naturalnumber.NaturalNumber;
2
3 /**
4  * Program to test arrays, references, and arrays of references.
5  *
6  * @author Gage Farmer
7  */
8 public final class ArraysAndReferences {
9
10     /**
11      * Private constructor so this utility class cannot be instantiated.
12      */
13     private ArraysAndReferences() {
14     }
15
16     /**
17      * Computes the product of the {@code NaturalNumber}s in the given array.
18      *
19      * @param nnArray
20      *        the array
21      * @return the product of the numbers in the given array
22      * @requires nnArray.length > 0
23      * @ensures <pre>
24      *   productOfArrayElements =
25      *     [nnArray[0] * nnArray[1] * ... * nnArray[nnArray.length-1]]
26      * </pre>
27      */
28     private static NaturalNumber productOfArrayElements(
29         NaturalNumber[] nnArray) {
30         assert nnArray != null : "Violation of: nnArray is not null";
31         assert nnArray.length > 0 : "Violation of: nnArray.length > 0";
32
33         NaturalNumber total = new NaturalNumber2(1);
34
35         for (int i = 0; i < nnArray.length; i++) {
36             total.multiply(nnArray[i]);
37         }
38
39         return total;
40     }
41
42     /**
43      * Replaces each element of {@code nnArray} with the partial product of all
44      * the elements in the incoming array, up to and including the current
45      * element.
46      *
47      * @param nnArray
48      *        the array
49      * @updates nnArray
50      * @requires nnArray.length > 0
51      * @ensures <pre>
52      *   for all i: integer where (0 <= i < nnArray.length)
53      *     (nnArray[i] = [#nnArray[0] * #nnArray[1] * ... * #nnArray[i]])
54      * </pre>
55      */
56     private static void computePartialProducts(NaturalNumber[] nnArray) {
57         assert nnArray != null : "Violation of: nnArray is not null";
58         assert nnArray.length > 0 : "Violation of: nnArray.length > 0";
```

```
63
64     NaturalNumber[] copy = nnArray;
65
66     for (int i = 1; i < nnArray.length; i++) {
67         nnArray[i].multiply(copy[i - 1]);
68     }
69
70 }
71
72 /**
73  * Creates and returns a new array of {@code NaturalNumber}s, of the same
74  * size of the given array, containing the partial products of the elements
75  * of the given array.
76  *
77  * @param nnArray
78  *       the array
79  * @return the array of partial products of the elements of the given array
80  * @requires nnArray.length > 0
81  * @ensures <pre>
82  * partialProducts.length = nnArray.length and
83  * for all i: integer where (0 <= i < partialProducts.length)
84  *   (partialProducts[i] = [nnArray[0] * nnArray[1] * ... * nnArray[i]])
85  * </pre>
86  */
87 private static NaturalNumber[] partialProducts(NaturalNumber[] nnArray) {
88     assert nnArray != null : "Violation of: nnArray is not null";
89     assert nnArray.length > 0 : "Violation of: nnArray.length > 0";
90
91     // TODO - fill in body
92
93     /*
94      * This line added just to make the program compilable. Should be
95      * replaced with appropriate return statement.
96      */
97     return null;
98 }
99
100 /**
101  * Main method.
102  *
103  * @param args
104  *       the command line arguments
105  */
106 public static void main(String[] args) {
107     SimpleWriter out = new SimpleWriter1L();
108
109     /*
110      * Initialize an array of NaturalNumbers with values 1 through 5.
111      */
112     NaturalNumber[] array = new NaturalNumber[5];
113     NaturalNumber count = new NaturalNumber2(1);
114     for (int i = 0; i < array.length; i++) {
115         array[i] = new NaturalNumber2(count);
116         count.increment();
117     }
118     /*
119      * Compute and output the product of the numbers in the array (should be
120      * 42!, i.e., the factorial of 42).
121      */
122 }
```

```
122     NaturalNumber product = productOfArrayElements(array);
123     out.println(product);
124
125     computePartialProducts(array);
126     for (int i = 0; i < array.length; i++) {
127         out.print(array[i].toString() + " ");
128     }
129
130     out.close();
131 }
132
133 }
```