```java
import components.naturalnumber.NaturalNumber;
import components.naturalnumber.NaturalNumber2;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * Program to test arrays, references, and arrays of references.
 *
 * @author Put your name here
 *
 */
public final class ArraysAndReferences {

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

    /**
     * Computes the product of the {@code NaturalNumber}s in the given array.
     *
     * @param nnArray
     *            the array
     * @return the product of the numbers in the given array
     * @requires nnArray.length > 0
     * @ensures <pre>
     * productOfArrayElements =
     *    [nnArray[0] * nnArray[1] * ... * nnArray[nnArray.length-1]]
     * </pre>
     */
    private static NaturalNumber productOfArrayElements(
            NaturalNumber[] nnArray) {
        assert nnArray != null : "Violation of: nnArray is not null";
        assert nnArray.length > 0 : "Violation of: nnArray.length > 0";

        NaturalNumber product = new NaturalNumber2(1);
        for (int i = 0; i < nnArray.length; i++) {
            product.multiply(nnArray[i]);
        }

        /*
         * This line added just to make the program compilable. Should be
         * replaced with appropriate return statement.
         */
        return product;
    }

    /**
     * Replaces each element of {@code nnArray} with the partial product of all
     * the elements in the incoming array, up to and including the current
     * element.
     *
     * @param nnArray
     *            the array
     * @updates nnArray
     * @requires nnArray.length > 0
```

```
58      * @ensures <pre>
59      * for all i: integer where (0 <= i < nnArray.length)
60      *   (nnArray[i] = [#nnArray[0] * #nnArray[1] * ... * #nnArray[i]])
61      * </pre>
62      */
63     private static void computePartialProducts(NaturalNumber[] nnArray) {
64         assert nnArray != null : "Violation of: nnArray is not null";
65         assert nnArray.length > 0 : "Violation of: nnArray.length > 0";
66
67         for (int i = 1; i < nnArray.length; i++) {
68             nnArray[i].multiply(nnArray[i - 1]);
69         }
70
71     }
72
73     /**
74      * Creates and returns a new array of {@code NaturalNumber}s, of the same
75      * size of the given array, containing the partial products of the elements
76      * of the given array.
77      *
78      * @param nnArray
79      *            the array
80      * @return the array of partial products of the elements of the given array
81      * @requires nnArray.length > 0
82      * @ensures <pre>
83      * partialProducts.length = nnArray.length  and
84      *  for all i: integer where (0 <= i < partialProducts.length)
85      *    (partialProducts[i] = [nnArray[0] * nnArray[1] * ... * nnArray[i]])
86      * </pre>
87      */
88     private static NaturalNumber[] partialProducts(NaturalNumber[] nnArray) {
89         assert nnArray != null : "Violation of: nnArray is not null";
90         assert nnArray.length > 0 : "Violation of: nnArray.length > 0";
91
92         // TODO - fill in body
93
94         /*
95          * This line added just to make the program compilable. Should be
96          * replaced with appropriate return statement.
97          */
98         return null;
99     }
100
101    /**
102     * Main method.
103     *
104     * @param args
105     *            the command line arguments
106     */
107    public static void main(String[] args) {
108        SimpleWriter out = new SimpleWriter1L();
109
110        /*
111         * Initialize an array of NaturalNumbers with values 1 through 42.
112         */
113        NaturalNumber[] array = new NaturalNumber[5];
114        NaturalNumber count = new NaturalNumber2(1);
```

```
115         for (int i = 0; i < array.length; i++) {
116             NaturalNumber temp = new NaturalNumber2(count);
117             array[i] = temp;
118             count.increment();
119         }
120         /*
121          * Compute and output the product of the numbers in the array (should be
122          * 42!, i.e., the factorial of 42).
123          */
124         NaturalNumber product = productOfArrayElements(array);
125         out.println(product);
126 
127         computePartialProducts(array);
128         for (int i = 0; i < array.length; i++) {
129             out.print(array[i] + ", ");
130         }
131 
132         out.close();
133     }
134 
135 }
136 
```