```java
import components.naturalnumber.NaturalNumber;

/**
 * Program with implementation of {@code NaturalNumber} secondary operation
 * {@code root} implemented as static method.
 *
 * @author Yiming Cheng
 *
 */
public final class NaturalNumberRoot {

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

    /**
     * Updates {@code n} to the {@code r}-th root of its incoming value.
     *
     * @param n
     *            the number whose root to compute
     * @param r
     *            root
     * @updates n
     * @requires r >= 2
     * @ensures n ^ (r) <= #n < (n + 1) ^ (r)
     */
    public static void root(NaturalNumber n, int r) {
        assert n != null : "Violation of: n is  not null";
        assert r >= 2 : "Violation of: r >= 2";
        NaturalNumber lowEnough = new NaturalNumber2(0);
        /*
         * make the lowest number in the loop as 0
         */
        final NaturalNumber one = new NaturalNumber2(1);
        final NaturalNumber two = new NaturalNumber2(2);
        NaturalNumber tooHigh = new NaturalNumber2(n);
        tooHigh.increment();
        /*
         * distribute the n to tooHigh
         */
        NaturalNumber result = new NaturalNumber2(tooHigh);
        result.subtract(lowEnough);
        while (result.compareTo(one) != 0) {
            /*
             * comparing the difference is 1
             */
            NaturalNumber sum = new NaturalNumber2(lowEnough);
            sum.add(tooHigh);
            sum.divide(two);
            NaturalNumber square = new NaturalNumber2(sum);
            square.power(r);
            /*
             * using the square to compare with the n to find
             */
            if (n.compareTo(square) < 0) {
                tooHigh.transferFrom(sum);
            } else {
                lowEnough.transferFrom(sum);
            }
            result.copyFrom(tooHigh);
```

```
66              /*
67               * initialize the number for the result
68               */
69              result.subtract(lowEnough);
70
71          }
72          n.copyFrom(lowEnough);
73      }
74
75      /**
76       * Main method.
77       *
78       * @param args
79       *            the command line arguments
80       */
81      public static void main(String[] args) {
82          //finding whether the answer is correct or not
83          SimpleWriter out = new SimpleWriter1L();
84
85          final String[] numbers = { "0", "1", "13", "1024", "189943527", "0",
86                  "1", "13", "4096", "189943527", "0", "1", "13", "1024",
87                  "189943527", "82", "82", "82", "82", "82", "9", "27", "81",
88                  "243", "143489073", "2147483647", "2147483648",
89                  "9223372036854775807", "9223372036854775808",
90                  "6189700196426901374495621111",
91                  "16225927682921336339157801028 8127",
92                  "17014118346046923173168730371588 4105727" };
93          final int[] roots = { 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 15, 15, 15, 15, 15,
94                  2, 3, 4, 5, 15, 2, 3, 4, 5, 15, 2, 2, 3, 3, 4, 5, 6 };
95          final String[] results = { "0", "1", "3", "32", "13782", "0", "1", "2",
96                  "16", "574", "0", "1", "1", "1", "3", "9", "4", "3", "2", "1",
97                  "3", "3", "3", "3", "3", "46340", "46340", "2097151", "2097152",
98                  "4987896", "2767208", "2353973" };
99
100         for (int i = 0; i < numbers.length; i++) {
101             NaturalNumber n = new NaturalNumber2(numbers[i]);
102             NaturalNumber r = new NaturalNumber2(results[i]);
103             root(n, roots[i]);
104             if (n.equals(r)) {
105                 out.println("Test " + (i + 1) + " passed: root(" + numbers[i]
106                         + ", " + roots[i] + ") = " + results[i]);
107             } else {
108                 out.println("*** Test " + (i + 1) + " failed: root("
109                         + numbers[i] + ", " + roots[i] + ") expected <"
110                         + results[i] + "> but was <" + n + ">");
111             }
112         }
113
114         out.close();
115     }
116
117 }
118
```