```java
import components.naturalnumber.NaturalNumber;

/**
 * Program with implementation of {@code NaturalNumber} secondary operation
 * {@code root} implemented as static method.
 *
 * @author Feras Akileh
 *
 */
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";

        // initializes the lowEnough variable of the algorithm
        NaturalNumber lowEnough = new NaturalNumber2(0);

        // initializes the tooHigh variable of the algorithm
        NaturalNumber tooHigh = new NaturalNumber2();
        tooHigh.copyFrom(n);
        tooHigh.increment();

        // initializes two variables for numbers 1 and 2
        NaturalNumber one = new NaturalNumber2(1);
        NaturalNumber two = new NaturalNumber2(2);

        // initializes the guess variable
        NaturalNumber guess = new NaturalNumber2();
        guess.copyFrom(tooHigh);
        guess.add(lowEnough);
        guess.divide(two);

        // initializes the variable for difference
        NaturalNumber diff = new NaturalNumber2();
        diff.copyFrom(tooHigh);
        diff.subtract(lowEnough);

        while (diff.compareTo(one) > 0) {
            guess.copyFrom(tooHigh);
            guess.add(lowEnough);
            guess.divide(two);
```

```
63
64              NaturalNumber guessB = new NaturalNumber2();
65              guessB.copyFrom(guess);
66              guessB.power(r);
67
68              if (guessB.compareTo(n) <= 0) {
69                  lowEnough.copyFrom(guess);
70              } else {
71                  tooHigh.copyFrom(guess);
72              }
73              diff.copyFrom(tooHigh);
74              diff.subtract(lowEnough);
75          }
76
77          n.copyFrom(lowEnough);
78
79      }
80
81      /**
82       * Main method.
83       *
84       * @param args
85       *            the command line arguments
86       */
87      public static void main(String[] args) {
88          SimpleWriter out = new SimpleWriter1L();
89
90          final String[] numbers = { "0", "1", "13", "1024", "189943527", "0",
91                  "1", "13", "4096", "189943527", "0", "1", "13", "1024",
92                  "189943527", "82", "82", "82", "82", "82", "9", "27", "81",
93                  "243", "143489073", "2147483647", "2147483648",
94                  "9223372036854775807", "9223372036854775808",
95                  "6189700196426901374495621111",
96                  "16225927682921336339157801028812 7",
97                  "17014118346046923173168730371588410 5727" };
98          final int[] roots = { 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 15, 15, 15, 15, 15,
99                  2, 3, 4, 5, 15, 2, 3, 4, 5, 15, 2, 2, 3, 3, 4, 5, 6 };
100         final String[] results = { "0", "1", "3", "32", "13782", "0", "1", "2",
101                 "16", "574", "0", "1", "1", "1", "3", "9", "4", "3", "2", "1",
102                 "3", "3", "3", "3", "3", "46340", "46340", "2097151", "2097152",
103                 "4987896", "2767208", "2353973" };
104
105         for (int i = 0; i < numbers.length; i++) {
106             NaturalNumber n = new NaturalNumber2(numbers[i]);
107             NaturalNumber r = new NaturalNumber2(results[i]);
108             root(n, roots[i]);
109             if (n.equals(r)) {
110                 out.println("Test " + (i + 1) + " passed: root(" + numbers[i]
111                         + ", " + roots[i] + ") = " + results[i]);
112             } else {
113                 out.println("*** Test " + (i + 1) + " failed: root("
114                         + numbers[i] + ", " + roots[i] + ") expected <"
115                         + results[i] + "> but was <" + n + ">");
116             }
117         }
118
119         out.close();
120     }
121
```

```
122 }
```