```java
import components.naturalnumber.NaturalNumber;

/**
 * Program with implementation of {@code NaturalNumber} secondary operation
 * {@code root} implemented as static method.
 *
 * @author Gage Farmer
 *
 */
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 ONE = new NaturalNumber2(1);
        NaturalNumber TWO = new NaturalNumber2(2);
        NaturalNumber lowEnough = new NaturalNumber2();
        NaturalNumber tooHigh = new NaturalNumber2();
        NaturalNumber power = new NaturalNumber2();
        tooHigh.copyFrom(n);
        NaturalNumber guess = new NaturalNumber2();
        guess.copyFrom(tooHigh);
        guess.add(lowEnough);
        NaturalNumber diff = new NaturalNumber2();
        diff.copyFrom(tooHigh);
        diff.subtract(lowEnough);

        while (diff.compareTo(ONE) == 1) {
            // set guess between highest and lowest vals
            guess.copyFrom(tooHigh);
            guess.add(lowEnough);
            guess.divide(TWO);
            // if guess is too high, set to lower val
            power.copyFrom(guess);
            power.power(r);
            if (power.compareTo(n) == 1) {
                tooHigh.copyFrom(guess);
                guess.copyFrom(lowEnough);
                // if guess is too low, set to higher val
            } else if (power.compareTo(n) == -1) {
                lowEnough.copyFrom(guess);
```

```
63                  guess.copyFrom(tooHigh);
64                  // if guess is right, leave loop
65              } else {
66                  break;
67              }
68              guess.copyFrom(tooHigh);
69              guess.add(lowEnough);
70              guess.divide(TWO);
71              diff.copyFrom(tooHigh);
72              diff.subtract(lowEnough);
73
74          }
75
76          n.copyFrom(guess);
77
78      }
79
80      /**
81       * Main method.
82       *
83       * @param args
84       *            the command line arguments
85       */
86      public static void main(String[] args) {
87          SimpleWriter out = new SimpleWriter1L();
88
89          final String[] numbers = { "0", "1", "13", "1024", "189943527", "0",
90                  "1", "13", "4096", "189943527", "0", "1", "13", "1024",
91                  "189943527", "82", "82", "82", "82", "82", "9", "27", "81",
92                  "243", "143489073", "2147483647", "2147483648",
93                  "92233720368547758O7", "9223372036854775808",
94                  "6189700196426901374495621l1",
95                  "162259276829213363391578010288127",
96                  "170141183460469231731687303715884105727" };
97          final int[] roots = { 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 15, 15, 15, 15, 15,
98                  2, 3, 4, 5, 15, 2, 3, 4, 5, 15, 2, 2, 3, 3, 4, 5, 6 };
99          final String[] results = { "0", "1", "3", "32", "13782", "0", "1", "2",
100                 "16", "574", "0", "1", "1", "1", "3", "9", "4", "3", "2", "1",
101                 "3", "3", "3", "3", "3", "46340", "46340", "2097151", "2097152",
102                 "4987896", "2767208", "2353973" };
103
104         for (int i = 0; i < numbers.length; i++) {
105             NaturalNumber n = new NaturalNumber2(numbers[i]);
106             NaturalNumber r = new NaturalNumber2(results[i]);
107             root(n, roots[i]);
108             if (n.equals(r)) {
109                 out.println("Test " + (i + 1) + " passed: root(" + numbers[i]
110                         + ", " + roots[i] + ") = " + results[i]);
111             } else {
112                 out.println("*** Test " + (i + 1) + " failed: root("
113                         + numbers[i] + ", " + roots[i] + ") expected <"
114                         + results[i] + "> but was <" + n + ">");
115             }
116         }
117
118         out.close();
119     }
120
121 }
```