CSE 2221 – Software 1: Software Components
Lecturer: Nyigel Spann

Project #6: Natural Number Roots

The Ohio State University
College of Engineering
Columbus, Ohio

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

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

        /*
         * Initialize lower bound for binary search (halving) algorithm.
         *
         * lower bound = 0
         */
        NaturalNumber lowerBound = new NaturalNumber2(0);

        /*
         * Initialize upper bound for binary search (halving) algorithm.
         *
         * upper bound = n + 1
         */
        NaturalNumber upperBound = new NaturalNumber2(n);
        upperBound.increment();

        /*
         * Initialize delta, the difference between the upper and the lower
         * bound for binary search (halving) algorithm.
         *
         * delta = upper bound - lower bound
         */
        NaturalNumber delta = new NaturalNumber2(upperBound);
        delta.subtract(lowerBound);

        /*
         * Initialize middle for binary search (halving) algorithm.
         *
         * middle = delta / 2 = (upper bound - lower bound) / 2
         */
        NaturalNumber middle = new NaturalNumber2(delta);
        final NaturalNumber two = new NaturalNumber2(2);
        middle.divide(two);

        final NaturalNumber one = new NaturalNumber2(1);
        while (delta.compareTo(one) > 0) {
           /*
            * Initialize power for binary search (halving) algorithm.
            *
            * power = middle ^ r
            */
           NaturalNumber power = new NaturalNumber2(middle);
           power.power(r);

           if (n.compareTo(power) < 0) {
              /*
               * Update upper bound.
               */
              upperBound.copyFrom(middle);
           } else {
              /*
               * Update lower bound.
               */
              lowerBound.copyFrom(middle);
```

```
        }

        /*
         * Update delta.
         */
        delta.copyFrom(upperBound);
        delta.subtract(lowerBound);
        /*
         * Update middle.
         */
        middle.copyFrom(lowerBound);
        middle.add(upperBound);
        middle.divide(two);
    }
    n.copyFrom(lowerBound);
}

/**
 * Main method.
 *
 * @param args
 *          the command line arguments
 */
public static void main(String[] args) {
    SimpleWriter out = new SimpleWriter1L();

    final String[] numbers = { "0", "1", "13", "1024", "189943527", "0",
            "1", "13", "4096", "189943527", "0", "1", "13", "1024",
            "189943527", "82", "82", "82", "82", "82", "9", "27", "81",
            "243", "143489073", "2147483647", "2147483648",
            "9223372036854775807", "9223372036854775808",
            "618970019642690137449562111",
            "162259276829213363391578010288127",
            "170141183460469231731687303715884105727" };
    final int[] roots = { 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 15, 15, 15, 15, 15,
            2, 3, 4, 5, 15, 2, 3, 4, 5, 15, 2, 2, 3, 3, 4, 5, 6 };
    final String[] results = { "0", "1", "3", "32", "13782", "0", "1", "2",
            "16", "574", "0", "1", "1", "1", "3", "9", "4", "3", "2", "1",
            "3", "3", "3", "3", "3", "46340", "46340", "2097151", "2097152",
            "4987896", "2767208", "2353973" };

    for (int i = 0; i < numbers.length; i++) {
        NaturalNumber n = new NaturalNumber2(numbers[i]);
        NaturalNumber r = new NaturalNumber2(results[i]);
```

```
            root(n, roots[i]);
            if (n.equals(r)) {
                out.println("Test " + (i + 1) + " passed: root(" + numbers[i]
                        + ", " + roots[i] + ") = " + results[i]);
            } else {
                out.println("*** Test " + (i + 1) + " failed: root("
                        + numbers[i] + ", " + roots[i] + ") expected <"
                        + results[i] + "> but was <" + n + ">");
            }
        }

        out.close();
    }

}
```