```java
import components.naturalnumber.NaturalNumber;
import components.naturalnumber.NaturalNumber2;

/**
 * Extension of {@code NaturalNumber2} with secondary operations implemented as
 * instance methods: add, subtract, and power.
 *
 * @author Gage Farmer
 *
 */
public final class NaturalNumberInstanceOps extends NaturalNumber2 {

    /**
     * No-argument constructor.
     */
    public NaturalNumberInstanceOps() {
    }

    /**
     * Constructor from {@code int}.
     *
     * @param i
     *            {@code int} to initialize from
     */
    public NaturalNumberInstanceOps(int i) {
        super(i);
    }

    /**
     * Constructor from {@code String}.
     *
     * @param s
     *            {@code String} to initialize from
     */
    public NaturalNumberInstanceOps(String s) {
        super(s);
    }

    /**
     * Constructor from {@code NaturalNumber}.
     *
     * @param n
     *            {@code NaturalNumber} to initialize from
     */
    public NaturalNumberInstanceOps(NaturalNumber n) {
        super(n);
    }

    @Override
    public void add(NaturalNumber n) {
        assert n != null : "Violation of: n is not null";
        /**
         * @decreases n
         */
        int thisLowDigit = this.divideBy10();
        int nLowDigit = n.divideBy10();
        if (!n.isZero()) {
            this.add(n);
        }
```

```
 60         thisLowDigit += nLowDigit;
 61         if (thisLowDigit >= RADIX) {
 62             thisLowDigit -= RADIX;
 63             this.increment();
 64         }
 65         this.multiplyBy10(thisLowDigit);
 66         n.multiplyBy10(nLowDigit);
 67     }
 68
 69     @Override
 70     public void subtract(NaturalNumber n) {
 71         assert n != null : "Violation of: n is not null";
 72         assert this.compareTo(n) >= 0 : "Violation of: this >= n";
 73
 74         int thisLowDigit = this.divideBy10();
 75         int nLowDigit = n.divideBy10();
 76         if (!n.isZero()) {
 77             this.subtract(n);
 78         }
 79         thisLowDigit -= nLowDigit;
 80         if (thisLowDigit < 0) {
 81             thisLowDigit += 10;
 82             this.decrement();
 83         }
 84         this.multiplyBy10(thisLowDigit);
 85         n.multiplyBy10(nLowDigit);
 86
 87     }
 88
 89     /**
 90      * Recursive function that calculates this to 'p'th power.
 91      *
 92      * @param p
 93      *            power, number of recursive cycles to do
 94      *
 95      * @param firstNum
 96      *            original number to do calculations with
 97      *
 98      * @ensures firstNum to 'p'th power
 99      */
100     public void power(int p, NaturalNumber firstNum) {
101         assert p >= 0 : "Violation of: p >= 0";
102
103         if (firstNum.equals(0)) {
104             firstNum.copyFrom(this);
105         }
106
107         if (p > 2) {
108             this.multiply(firstNum);
109             // "Assignment of parameter p is not allowed" <- don't care, didn't ask.
110             p -= 1;
111             this.power(p);
112         }
113
114     }
115
116 }
```