```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 Put your name here
 *
 */
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);
        }
        thisLowDigit += nLowDigit;
        if (thisLowDigit >= RADIX) {
            thisLowDigit -= RADIX;
            this.increment();
        }
        this.multiplyBy10(thisLowDigit);
        n.multiplyBy10(nLowDigit);
    }

    @Override
    public void subtract(NaturalNumber n) {
        assert n != null : "Violation of: n is not null";
        assert this.compareTo(n) >= 0 : "Violation of: this >= n";
        /**
         * @decreases n
         */
        int thisLowDigit = this.divideBy10();
        int nLowDigit = n.divideBy10();
        if (!n.isZero()) {
            this.subtract(n);
        }
        thisLowDigit -= nLowDigit;
        if (thisLowDigit < 0) {
            thisLowDigit += RADIX;
            this.decrement();
        }
        this.multiplyBy10(thisLowDigit);
        n.multiplyBy10(nLowDigit);
    }

    @Override
    public void power(int p) {
        assert p >= 0 : "Violation of: p >= 0";

        NaturalNumber x = new NaturalNumber2(this);
        NaturalNumber y = new NaturalNumber2(this);

        if (p % 2 == 0) {
            x.power(p / 2);
            x.power(2);
        } else if (p % 2 == 1) {
            x.power(p / 2);
            x.power(2);
            x.multiply(y);
        }
        this.copyFrom(x);
    }
}
```