```java
import components.naturalnumber.NaturalNumber;

/**
 * Program to evaluate XMLTree expressions of {@code NN}.
 *
 * @author Yiming Cheng
 *
 */
public final class XMLTreeNNExpressionEvaluator {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private XMLTreeNNExpressionEvaluator() {
    }

    /**
     * Evaluate the given expression.
     *
     * @param exp
     *            the {@code XMLTree} representing the expression
     * @return the value of the expression
     * @requires <pre>
     * [exp is a subtree of a well-formed XML arithmetic expression]  and
     *  [the label of the root of exp is not "expression"]
     * </pre>
     * @ensures evaluate = [the value of the expression]
     */
    private static NaturalNumber evaluate(XMLTree exp) {
        NaturalNumber evl = new NaturalNumber2();
        if (exp.numberOfChildren() > 0) {
            //find the xml's label which is times, and do the corresponding actions
            if (exp.label().equals("times")) {
                evl.copyFrom(evaluate(exp.child(0)));
                evl.multiply(evaluate(exp.child(1)));
                ////find the xml's label which is divide, and do the corresponding actions
            } else if (exp.label().equals("divide")) {
                if (!evaluate(exp.child(1)).isZero()) {
                    evl.copyFrom(evaluate(exp.child(0)));
                    evl.divide(evaluate(exp.child(1)));
                    //report the error when the divisor would be smaller than 0
                } else {
                    Reporter.fatalErrorToConsole(
                            "The divisor would be more than 0.");
                }
                //find the xml's label which is plus, and do the corresponding actions
            } else if (exp.label().equals("plus")) {
                evl.copyFrom(evaluate(exp.child(0)));
                evl.add(evaluate(exp.child(1)));
                ////find the xml's label which is minus, and do the corresponding actions
            } else if (exp.label().equals("minus")) {
                if (evaluate(exp.child(0))
                        .compareTo(evaluate(exp.child(1))) >= 0) {
                    evl.copyFrom(evaluate(exp.child(0)));
                    evl.subtract(evaluate(exp.child(1)));
                } else {
                    //report the error when the result would be smaller than 0
                    Reporter.fatalErrorToConsole(
                            "The second one would be smaller than the first one");
                }
            }
        } else {
            String word = exp.attributeValue("value");
            evl = new NaturalNumber2(word);
        }
        return evl;
    }

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments
     */
    public static void main(String[] args) {
        SimpleReader in = new SimpleReader1L();
        SimpleWriter out = new SimpleWriter1L();

        out.print("Enter the name of an expression XML file: ");
        String file = in.nextLine();
        while (!file.equals("")) {
            XMLTree exp = new XMLTree1(file);
            out.println(evaluate(exp.child(0)));
            out.print("Enter the name of an expression XML file: ");
            file = in.nextLine();
        }

        in.close();
        out.close();
    }

}

```