```java
import components.naturalnumber.NaturalNumber;

/**
 * Program to evaluate XMLTree expressions of {@code int}.
 *
 * @author Jonathan Pater
 *
 */
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"] and
     *  [the expression will not attempt to divide by 0 and the expression will
     *  not result in a negative number]
     *
     * </pre>
     * @ensures evaluate = [the value of the expression]
     * @ensures the value of the expression returned is >= 0
     * @ensures the expression does not divide by zero.
     */
    private static NaturalNumber evaluate(XMLTree exp) {
        assert exp != null : "Violation of: exp is not null";

        // TODO - fill in body
        NaturalNumber evaluate;
        NaturalNumber zero1 = new NaturalNumber2(0);
        String op = exp.label();
        if (exp.numberOfChildren() == 0) {
            evaluate = new
NaturalNumber2(exp.attributeValue("value"));
        } else {
            XMLTree child1 = exp.child(0);
            XMLTree child2 = exp.child(1);
            if (op.equals("plus")) {
                evaluate = evaluate(child1);
                NaturalNumber temp2 = evaluate(child2);
                evaluate.add(temp2);
            } else if (op.equals("times")) {
                evaluate = evaluate(child1);
                NaturalNumber temp2 = evaluate(child2);
                evaluate.multiply(temp2);
            } else if (op.equals("divide")) {
                evaluate = evaluate(child1);
                NaturalNumber temp2 = evaluate(child2);
                int zero = temp2.compareTo(zero1);
                if (zero == 0) {
                    Reporter.fatalErrorToConsole(
                            "Violation: Attempting to Divide by Zero");
                }
                evaluate.divide(temp2);
            } else {
                evaluate = evaluate(child1);
                NaturalNumber temp2 = evaluate(child2);
                int negative = temp2.compareTo(evaluate);
                if (negative > 0) { //the case where the difference is negative
                    Reporter.fatalErrorToConsole(
                            "Violation: Expression Evaluates to a Negative"
                                    + " Number");
                }
                evaluate.subtract(temp2);
            }
        }
        return evaluate;
    }

    /**
     * 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();
    }

}
```