```java
import components.naturalnumber.NaturalNumber;
import components.naturalnumber.NaturalNumber2;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;
import components.utilities.Reporter;
import components.xmltree.XMLTree;
import components.xmltree.XMLTree1;

/**
 * Program to evaluate XMLTree expressions of {@code int}.
 *
 * @author Put your name here
 *
 */
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) {
        assert exp != null : "Violation of: exp is not null";

        NaturalNumber num1 = new NaturalNumber2();
        NaturalNumber num2 = new NaturalNumber2();
        NaturalNumber result = new NaturalNumber2();
        NaturalNumber temp = new NaturalNumber2();
        String label = exp.label();

        if (exp.numberOfChildren() > 0) {
            num1.copyFrom(evaluate(exp.child(0)));
            if (exp.numberOfChildren() > 1) {
                num2.copyFrom(evaluate(exp.child(1)));
            }
            if (label.equals("plus")) {
                temp.copyFrom(num1);
                temp.add(num2);
                result.copyFrom(temp);
            } else if (label.equals("minus")) {
                temp.copyFrom(num1);
                // Making sure result is not negative
                if (num2.compareTo(num1) > 0) {
                    Reporter.fatalErrorToConsole(
                            "NaturalNumber cannot be negative.");
                } else {
                    temp.subtract(num2);
                    result.copyFrom(temp);
                }
            } else if (label.equals("times")) {
                temp.copyFrom(num1);
                temp.multiply(num2);
                result.copyFrom(temp);
            } else if (label.equals("divide")) {
                // Making sure we do not divide by 0
                if (num2.toInt() == 0) {
                    Reporter.fatalErrorToConsole("Cannot divide by zero.");
                } else {
                    temp.copyFrom(num1);
                    temp.divide(num2);
                    result.copyFrom(temp);
                }
            }
        } else {
            result.setFromString(exp.attributeValue("value"));
        }

        return result;
    }

    /**
     * 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();
    }

}
```