```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 David Park
 *
 */
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";

        // TODO - fill in body

        NaturalNumber result = new NaturalNumber2(0);
        NaturalNumber second = new NaturalNumber2(0);

        // Check the operation by xmlTree node
        if (exp.label().equals("times")) {
            // If the operation is multiplication, recursively evaluate with both child
            result = evaluate(exp.child(0));
            second = evaluate(exp.child(1));
            result.multiply(second);
        } else if (exp.label().equals("divide")) {
            // If the operation is division, recursively evaluate with both child
            result = evaluate(exp.child(0));
            second = evaluate(exp.child(1));
            // Check for division by zero, which is not allowed
            if (evaluate(exp.child(1)).isZero()) {
                Reporter.fatalErrorToConsole(
                        "A number divided by zero is undefinded");
            }
            result.divide(second);
        } else if (exp.label().equals("plus")) {
            // If the operation is addition, recursively evaluate both child
            // and then add the results together
            result = evaluate(exp.child(0));
            second = evaluate(exp.child(1));
            result.add(second);
        } else if (exp.label().equals("minus")) {
            // If the operation is subtraction, recursively evaluate both child
            // and then subtract the second result from the first
            result = evaluate(exp.child(0));
            second = evaluate(exp.child(1));
            if (result.compareTo(second) < 0) {
                Reporter.fatalErrorToConsole(
                        "A natural number cannot be negative. ");
            }
            result.subtract(second);
        } else if (exp.label().equals("number")) {
            result = new NaturalNumber2(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();
    }
}
```