```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 Robert Frenken
 *
 */
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"]
     * [There isn't a subtraction operation where the resultant is less than zero]
     * [There isn't a division operation where the divider is zero]
     * </pre>
     *
     * @ensures evaluate = [the value of the expression]
     */
    private static NaturalNumber evaluate(XMLTree exp) {
        NaturalNumber num = new NaturalNumber2(0);
        // base case
        if (exp.label().equals("number")) {
            String val = exp.attributeValue("value");
            num.setFromString(val);

        } else {
            NaturalNumber first = num.newInstance();
            NaturalNumber second = num.newInstance();
            if (exp.numberOfChildren() > 1) {
                first.copyFrom(evaluate(exp.child(0)));
                second.copyFrom(evaluate(exp.child(1)));

                // determine operation
                if (exp.label().equals("plus")) {
                    first.add(second);
                    num.transferFrom(first);
```

```
58                } else if (exp.label().equals("minus")) {
59                    if (first.compareTo(second) < 0) {
60                        Reporter.fatalErrorToConsole(
61                                "Subtraction evaluation cannot be less than zero");
62                    }
63                    first.subtract(second);
64                    num.transferFrom(first);
65                } else if (exp.label().equals("times")) {
66                    first.multiply(second);
67                    num.transferFrom(first);
68                } else {
69                    if (second.isZero()) {
70                        Reporter.fatalErrorToConsole(
71                                "Denominator cannot be zero");
72                    }
73                    first.divide(second);
74                    num.transferFrom(first);
75                }
76            } else {
77                first.copyFrom(evaluate(exp.child(0)));
78                num.transferFrom(first);
79            }
80
81        }
82
83        return num;
84
85    }
86
87    /**
88     * Main method.
89     *
90     * @param args
91     *            the command line arguments
92     */
93    public static void main(String[] args) {
94        SimpleReader in = new SimpleReader1L();
95        SimpleWriter out = new SimpleWriter1L();
96
97        out.print("Enter the name of an expression XML file: ");
98        String file = in.nextLine();
99        while (!file.equals("")) {
100           XMLTree exp = new XMLTree1(file);
101           out.println(evaluate(exp.child(0)));
102           out.print("Enter the name of an expression XML file: ");
103           file = in.nextLine();
104       }
105
106       in.close();
107       out.close();
108   }
109
110}
```