```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 Feras Akileh
 *
 */
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";

        // initializes the variable for one
        NaturalNumber one = new NaturalNumber2(1);

        // initializes the string for the exp.label()
        String expLabel = exp.label();

        // checks for the multiplication in the tree
        if (expLabel.equals("times")) {
            NaturalNumber childA = evaluate(exp.child(0));
            childA.multiply(evaluate(exp.child(1)));
            return childA;
        }

        // checks for the addition in the tree
        if (expLabel.equals("plus")) {
            NaturalNumber childA = evaluate(exp.child(0));
            childA.add(evaluate(exp.child(1)));
            return childA;
        }

```

```
60          // checks for the division in the tree
61          if (expLabel.equals("divide")) {
62              NaturalNumber childA = evaluate(exp.child(0));
63              NaturalNumber childB = evaluate(exp.child(1));
64              if (childB.canConvertToInt() && childB.toInt() == 0) {
65                  Reporter.fatalErrorToConsole("Sorry! You cannot divide by 0!");
66              }
67              childA.divide(childB);
68              return childA;
69          }
70
71          // checks for the subtraction in the tree
72          if (expLabel.equals("minus")) {
73              NaturalNumber childA = evaluate(exp.child(0));
74              NaturalNumber childB = evaluate(exp.child(1));
75              if (childB.compareTo(childA) > 0) {
76                  Reporter.fatalErrorToConsole("Sorry! You cannot divide by 0!");
77              }
78              childA.subtract(childB);
79              return childA;
80          }
81
82          // checks for the numbers in the tree
83          if (expLabel.equals("number")) {
84              NaturalNumber m = new NaturalNumber2(exp.attributeValue("value"));
85              return m;
86          } else {
87              return one;
88          }
89
90      }
91
92      /**
93       * Main method.
94       *
95       * @param args
96       *            the command line arguments
97       */
98      public static void main(String[] args) {
99          SimpleReader in = new SimpleReader1L();
100         SimpleWriter out = new SimpleWriter1L();
101
102         out.print("Enter the name of an expression XML file: ");
103         String file = in.nextLine();
104         while (!file.equals("")) {
105             XMLTree exp = new XMLTree1(file);
106             out.println(evaluate(exp.child(0)));
107             out.print("Enter the name of an expression XML file: ");
108             file = in.nextLine();
109         }
110
111         in.close();
112         out.close();
113     }
114
115 }
```