```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.xmltree.XMLTree;
import components.xmltree.XMLTree1;

/**
 * Program to evaluate XMLTree expressions of Natural Numbers.
 *
 * @author Vishal Kumar
 *
 */
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 value = new NaturalNumber2(0);
        NaturalNumber zero = new NaturalNumber2(0);

        // statement to get past the first root node
        if (exp.label().equals("expression")) {
            value.add(evaluate(exp.child(0)));
        }
        // base case
        else if (exp.hasAttribute("value")) {
            NaturalNumber num = new NaturalNumber2();
            num.setFromInt(Integer.parseInt(exp.attributeValue("value")));
            value.add(num);
        }

        // recursive if else block to evaluate math operations
        else if (exp.label().equals("plus")) {
            NaturalNumber sum = value.newInstance();
            sum.add(evaluate(exp.child(0)));
            sum.add(evaluate(exp.child(1)));
            value.add(sum);
```

```
58          } else if (exp.label().equals("minus")) {
59              NaturalNumber difference = value.newInstance();
60              NaturalNumber child0 = evaluate(exp.child(0));
61              NaturalNumber child1 = evaluate(exp.child(1));
62              // verify that NN subtract method precondition is fulfilled
63              if (child0.compareTo(child1) >= 0) {
64                  difference.add(child0);
65                  difference.subtract(child1);
66                  value.add(difference);
67              } else {
68                  components.utilities.Reporter
69                          .fatalErrorToConsole("Error: child0 < child1");
70              }
71          } else if (exp.label().equals("times")) {
72              NaturalNumber product = value.newInstance();
73              product.add(evaluate(exp.child(0)));
74              product.multiply(evaluate(exp.child(1)));
75              value.add(product);
76          } else {
77              NaturalNumber quotient = value.newInstance();
78              NaturalNumber child0 = evaluate(exp.child(0));
79              NaturalNumber child1 = evaluate(exp.child(1));
80              // verify that NN divide method precondition is fulfilled
81              if (child1.compareTo(zero) > 0) {
82                  quotient.add(child0);
83                  quotient.divide(child1);
84                  value.add(quotient);
85              } else {
86                  components.utilities.Reporter
87                          .fatalErrorToConsole("Error: divide by zero");
88              }
89          }
90
91          // return final value
92          return value;
93      }
94
95      /**
96       * Main method.
97       *
98       * @param args
99       *            the command line arguments
100      */
101     public static void main(String[] args) {
102         SimpleReader in = new SimpleReader1L();
103         SimpleWriter out = new SimpleWriter1L();
104
105         out.print("Enter the name of an expression XML file: ");
106         String file = in.nextLine();
107         while (!file.equals("")) {
108             XMLTree exp = new XMLTree1(file);
109             out.println(evaluate(exp.child(0)));
110             out.print("Enter the name of an expression XML file: ");
111             file = in.nextLine();
112         }
113
114         in.close();
```

```
115         out.close();
116     }
117
118 }
```