```java
import components.naturalnumber.NaturalNumber;

/**
 * Program to evaluate XMLTree expressions of {@code int}.
 *
 * @author Gage Farmer
 *
 *         idk why this was scheduled due right after spring break....
 *
 */
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 total = new NaturalNumber2();
        NaturalNumber temp = new NaturalNumber2();
        NaturalNumber ZERO = new NaturalNumber2();

        if (exp.numberOfChildren() > 0) {
            for (int i = 0; i < exp.numberOfChildren(); i++) {
                if (exp.label().equals("plus")) {
                    temp = new NaturalNumber2(evaluate(exp.child(i)));
                    total.add(temp);
                } else if (exp.label().equals("minus")) {
                    if (total.compareTo(ZERO) == 0) {
                        temp = new NaturalNumber2(evaluate(exp.child(i)));
                        total.add(temp);
                    } else {
                        temp = new NaturalNumber2(evaluate(exp.child(i)));
                        total.subtract(temp);
                    }
                } else if (exp.label().equals("times")) {
                    if (total.compareTo(ZERO) == 0) {
                        temp = new NaturalNumber2(evaluate(exp.child(i)));
                        total.add(temp);
                    } else {
                        temp = new NaturalNumber2(evaluate(exp.child(i)));
                        total.multiply(temp);
                    }
                } else if (exp.label().equals("divide")) {
                    if (total.compareTo(ZERO) == 0) {
```

```
 67                          temp = new NaturalNumber2(evaluate(exp.child(i)));
 68                          total.add(temp);
 69                      } else {
 70                          assert evaluate(exp.child(i)).compareTo(
 71                                  ZERO) != 0 : "Violation of: Divide by 0";
 72                          temp = new NaturalNumber2(evaluate(exp.child(i)));
 73                          total.divide(temp);
 74                      }
 75                  }
 76
 77              }
 78          } else {
 79              total = new NaturalNumber2(
 80                      Integer.parseInt(exp.attributeValue("value")));
 81          }
 82
 83          /*
 84           * This line added just to make the program compilable. Should be
 85           * replaced with appropriate return statement.
 86           */
 87          return total;
 88      }
 89
 90      /**
 91       * Main method.
 92       *
 93       * @param args
 94       *            the command line arguments
 95       */
 96      public static void main(String[] args) {
 97          SimpleReader in = new SimpleReader1L();
 98          SimpleWriter out = new SimpleWriter1L();
 99
100          out.print("Enter the name of an expression XML file: ");
101          String file = in.nextLine();
102          while (!file.equals("")) {
103              XMLTree exp = new XMLTree1(file);
104              out.println(evaluate(exp.child(0)));
105              out.print("Enter the name of an expression XML file: ");
106              file = in.nextLine();
107          }
108
109          in.close();
110          out.close();
111      }
112
113  }
```