```java
import components.naturalnumber.NaturalNumber;

/**
 * Program to evaluate XMLTree expressions of {@code int}.
 *
 * @author Shyam Sai Bethina
 *
 */
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"]
     * [exp to not have 0 as the second child of a division node]
     *  [if there is one(no division by 0)]
     * [All operational results should be positive]
     * </pre>
     * @ensures evaluate = [the value of the expression]
     */
    private static NaturalNumber evaluate(XMLTree exp) {
        NaturalNumber result = new NaturalNumber2();

        if (!exp.label().equals("number")) {
            //Each node only has two child nodes
            NaturalNumber firstNum = new
NaturalNumber2(evaluate(exp.child(0)));
            NaturalNumber secondNum = new NaturalNumber2(
                    evaluate(exp.child(1)));

            //does the operation depending on what type of label
```

```
   the node is
50             if (exp.label().equals("plus")) {
51                 firstNum.add(secondNum);
52             } else if (exp.label().equals("minus")) {
53                 if (firstNum.compareTo(secondNum) < 0) {
54                     /*
55                      * Since NN cannot be negative, outputs a
   message if
56                      * secondNum is more than firstNum
57                      */
58                     Reporter.fatalErrorToConsole(
59                             "Subtraction cannot result in negative
   number");
60                 }
61                 firstNum.subtract(secondNum);
62             } else if (exp.label().equals("times")) {
63                 firstNum.multiply(secondNum);
64             } else {
65                 //If secondNum is zero, outputs message saying so
66                 if (secondNum.isZero()) {
67                     Reporter.fatalErrorToConsole(
68                             "Illegal Operation - Division by zero
   ");
69                 }
70                 firstNum.divide(secondNum);
71             }
72
73             /*
74              * copies value of firstNum to result so that the
   final value
75              * persists out of the conditional statements
76              */
77             result.copyFrom(firstNum);
78
79         } else {
80             //if the label does equal number, then returns the
   value of the number instead
81             result = new NaturalNumber2(
82
   Integer.parseInt(exp.attributeValue("value")));
83         }
84         return result;
85
86     }
```

```
 87
 88      /**
 89       * Main method.
 90       *
 91       * @param args
 92       *            the command line arguments
 93       */
 94      public static void main(String[] args) {
 95          SimpleReader in = new SimpleReader1L();
 96          SimpleWriter out = new SimpleWriter1L();
 97
 98          out.print("Enter the name of an expression XML file: ");
 99          String file = in.nextLine();
100          while (!file.equals("")) {
101              XMLTree exp = new XMLTree1(file);
102              out.println(evaluate(exp.child(0)));
103              out.print("Enter the name of an expression XML file:
    ");
104              file = in.nextLine();
105          }
106
107          in.close();
108          out.close();
109      }
110
111  }
112
```