```java
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 {@code int}.
 *
 * @author Jeng Zhuang
 *
 */
public final class XMLTreeIntExpressionEvaluator {

    /**
     * Private constructor so this utility class cannot be
instantiated.
     */
    private XMLTreeIntExpressionEvaluator() {
    }

    /**
     * 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 int evaluate(XMLTree exp) {
        assert exp != null : "Violation of: exp is not null";

```

```
37          // Variable to store the result of the expression
38          int result = 0;
39
40          // Base case: if the current node is a number, return
  its value
41          if (exp.label().equals("number")) {
42              String value = exp.attributeValue("value");
43              return Integer.parseInt(value);
44          } else {
45
46              // Recursive case: evaluate the left and right
  subtrees
47              int left = evaluate(exp.child(0)); // Evaluate the
  left child
48              int right = evaluate(exp.child(1)); // Evaluate the
  right child
49
50              // Determine the operator and compute the result
51              String operator = exp.label();
52              if (operator.equals("plus")) {
53                  result = left + right; // Addition
54
55              } else if (operator.equals("minus")) {
56                  result = left - right; // Subtraction
57
58              } else if (operator.equals("times")) {
59                  result = left * right; // Multiplication
60
61              } else if (operator.equals("divide")) {
62                  result = left / right; // Division
63
64              }
65              return result;
66
67          }
68      }
69
70      /**
71       * Main method.
```

```
72      *
73      * @param args
74      *             the command line arguments
75      */
76     public static void main(String[] args) {
77         SimpleReader in = new SimpleReader1L();
78         SimpleWriter out = new SimpleWriter1L();
79
80         out.print("Enter the name of an expression XML file: ");
81         String file = in.nextLine();
82         while (!file.equals("")) {
83             XMLTree exp = new XMLTree1(file);
84             out.println(evaluate(exp.child(0)));
85             out.print("Enter the name of an expression XML file:
   ");
86             file = in.nextLine();
87         }
88
89         in.close();
90         out.close();
91     }
92
93 }
94
```