```java
import components.simplereader.SimpleReader;

/**
 * Program to evaluate XMLTree expressions of {@code int}.
 *
 * @author Gage Farmer
 *
 */
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";
        int total = 0;

        if (exp.numberOfChildren() > 0) {
            for (int i = 0; i < exp.numberOfChildren(); i++) {
                if (exp.label().equals("plus")) {
                    total += evaluate(exp.child(i));
                } else if (exp.label().equals("minus")) {
                    if (total == 0) {
                        total += evaluate(exp.child(i));
                    } else {
                        total -= evaluate(exp.child(i));
                    }
                } else if (exp.label().equals("times")) {
                    if (total == 0) {
                        total += evaluate(exp.child(i));
                    } else {
                        total *= evaluate(exp.child(i));
                    }
                } else if (exp.label().equals("divide")) {
                    if (total == 0) {
                        total += evaluate(exp.child(i));
                    } else {
                        assert evaluate(exp
                                .child(i)) != 0 : "Violation of: Divide by 0";
                        total /= evaluate(exp.child(i));
                    }
                }

            }
```

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