```java
import components.simplereader.SimpleReader;

/**
 * Program to evaluate XMLTree expressions of {@code int}.
 *
 * @author Feras Akileh
 *
 */
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";

        // initializes the string for the exp.label()
        String expLabel = exp.label();

        // checks for the multiplication in the tree
        if (expLabel.equals("times")) {
            return evaluate(exp.child(0)) * evaluate(exp.child(1));
        }
        // checks for the addition in the tree
        if (expLabel.equals("plus")) {
            return evaluate(exp.child(0)) + evaluate(exp.child(1));
        }
        // checks for the division in the tree
        if (expLabel.equals("divide")) {
            // initializes a variable that is the dividend so that an error
            // can be reported if the dividend is ever 0
            int divCheck = evaluate(exp.child(1));
            if (divCheck == 0) {
                Reporter.fatalErrorToConsole("Sorry! You cannot divide by 0!");
            }
            return evaluate(exp.child(0)) / evaluate(exp.child(1));
        }
        // checks for the subtraction in the tree
        if (expLabel.equals("minus")) {
            return evaluate(exp.child(0)) - evaluate(exp.child(1));
        }

        // checks for the numbers in the tree
        if (expLabel.equals("number")) {
```

```
66              String value = exp.attributeValue("value");
67              return Integer.parseInt(value);
68          } else {
69              return 1;
70          }
71
72      }
73
74      /**
75       * Main method.
76       *
77       * @param args
78       *            the command line arguments
79       */
80      public static void main(String[] args) {
81          SimpleReader in = new SimpleReader1L();
82          SimpleWriter out = new SimpleWriter1L();
83
84          out.print("Enter the name of an expression XML file: ");
85          String file = in.nextLine();
86          while (!file.equals("")) {
87              XMLTree exp = new XMLTree1(file);
88              out.println(evaluate(exp.child(0)));
89              out.print("Enter the name of an expression XML file: ");
90              file = in.nextLine();
91          }
92
93          in.close();
94          out.close();
95      }
96
97  }
```