```java
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;
import components.utilities.Reporter;
import components.xmltree.XMLTree;
import components.xmltree.XMLTree1;

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

        // TODO - fill in body

        int result = 0;

        if (exp.label().equals("times")) {
            // If the operation is multiplication, recursively solve left n right
            result = evaluate(exp.child(0)) * evaluate(exp.child(1));
        } else if (exp.label().equals("divide")) {
            if (evaluate(exp.child(1)) == 0) {
                Reporter.fatalErrorToConsole(
                        "A number divided by zero is undefined.");
            } else {
                result = evaluate(exp.child(0)) / evaluate(exp.child(1));
            }
        } else if (exp.label().equals("plus")) {
            // If the operation is addition, recursively
            // evaluate the left and right child
            // of this node and add their results.
            result = evaluate(exp.child(0)) + evaluate(exp.child(1));
        } else if (exp.label().equals("minus")) {
            // If the operation is subtraction, recursive solve left n right
            result = evaluate(exp.child(0)) - evaluate(exp.child(1));
        } else if (exp.label().equals("number")) {
            result = Integer.parseInt(exp.attributeValue("value"));
        }
        return result;
    }

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments
     */
    public static void main(String[] args) {
        SimpleReader in = new SimpleReader1L();
        SimpleWriter out = new SimpleWriter1L();

        out.print("Enter the name of an expression XML file: ");
        String file = in.nextLine();
        while (!file.equals("")) {
            XMLTree exp = new XMLTree1(file);
            out.println(evaluate(exp.child(0)));
            out.print("Enter the name of an expression XML file: ");
            file = in.nextLine();
        }

        in.close();
        out.close();
    }
}

```