```
import components.naturalnumber.NaturalNumber;
import components.naturalnumber.NaturalNumber2;
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 Ansh Pachauri
 *
 */
public final class XMLTreeIntNNExpressionEvaluator {

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

    /**
     * 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]
     */
```

```java
    private static NaturalNumber evaluate(XMLTree exp) {
        assert exp != null : "Violation of: exp is not null";

        // TODO - fill in body
        NaturalNumber noReOccur = new NaturalNumber2();
        // when the node is a number
        if (exp.label().equals("number")) {
            noReOccur = new NaturalNumber2(exp.attributeValue("value"));
            // when the node is not a number
        } else {
            String action = exp.label();
            NaturalNumber zero = new NaturalNumber2(0);
            String msg = "The expression has violated a precondition";

            XMLTree one = exp.child(0);
            XMLTree two = exp.child(1);

            noReOccur.transferFrom(evaluate(one));
            // calculating the expression
            if (action.equals("plus")) {
                noReOccur.add(evaluate(two));
            } else if (action.equals("divide")) {
                if (evaluate(two).equals(zero)) {
                    // giving error for a precondition not satisfied
                    Reporter.fatalErrorToConsole(msg);
                } else {
                    noReOccur.divide(evaluate(two));
                }
            } else if (action.equals("multiply")) {
                noReOccur.multiply(evaluate(two));
            } else {
                noReOccur.subtract(evaluate(two));
            }
        }
        return noReOccur;
    }

    /**
     * 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();
    }

}

```