```java
import components.naturalnumber.NaturalNumber;

/**
 * Program to evaluate XMLTree expressions of {@code NaturalNumber}.
 *
 * @author Put your Vaishnavi Kasabwala
 *
 */
public final class XMLTreeNNExpressionEvaluator {

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

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

        NaturalNumber solution = new NaturalNumber2(0);

        // digit
        if (exp.label().equals("number")) {
            NaturalNumber temp = new NaturalNumber2(
                    exp.attributeValue("value"));
            solution.copyFrom(temp);
        }
        // +
        else if (exp.label().equals("plus")) {
            solution = evaluate(exp.child(0));
            solution.add(evaluate(exp.child(1)));
        }
        // -
        else if (exp.label().equals("minus")) {
            solution = evaluate(exp.child(0));
            NaturalNumber second = evaluate(exp.child(1));
            if (second.compareTo(solution) > 0) {
                Reporter.fatalErrorToConsole("Negative natural number");
            }
            solution.subtract(second);
        }
        // *
        else if (exp.label().equals("times")) {
            solution = evaluate(exp.child(0));
            solution.multiply(evaluate(exp.child(1)));

        // "/"
        else if (exp.label().equals("divide")) {
            solution = evaluate(exp.child(0));
            NaturalNumber denomenator = evaluate(exp.child(1));
            if (denomenator.canConvertToInt() && denomenator.toInt() == 0) {
                Reporter.fatalErrorToConsole("Error: Dividing by zero");
            }
            solution.divide(denomenator);
        }

        return solution;
    }

    /**
     * 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();
    }

}
```