```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)));
```

```
66          }
67          // "/"
68          else if (exp.label().equals("divide")) {
69              solution = evaluate(exp.child(0));
70              NaturalNumber denomenator = evaluate(exp.child(1));
71              if (denomenator.canConvertToInt() && denomenator.toInt() == 0) {
72                  Reporter.fatalErrorToConsole("Error: Dividing by zero");
73              }
74              solution.divide(denomenator);
75          }
76
77          return solution;
78      }
79
80      /**
81       * Main method.
82       *
83       * @param args
84       *            the command line arguments
85       */
86      public static void main(String[] args) {
87          SimpleReader in = new SimpleReader1L();
88          SimpleWriter out = new SimpleWriter1L();
89
90          out.print("Enter the name of an expression XML file: ");
91          String file = in.nextLine();
92          while (!file.equals("")) {
93              XMLTree exp = new XMLTree1(file);
94              out.println(evaluate(exp.child(0)));
95              out.print("Enter the name of an expression XML file: ");
96              file = in.nextLine();
97          }
98
99          in.close();
100         out.close();
101     }
102
103 }
```