

```
1 import components.naturalnumber.NaturalNumber;
10
11 /**
12  * Program to evaluate XMLTree expressions of {@code NaturalNumber}.
13  *
14  * @author Put your Vaishnavi Kasabwala
15  *
16  */
17 public final class XMLTreeNNEvaluationEvaluator {
18
19     /**
20      * Private constructor so this utility class cannot be instantiated.
21      */
22     private XMLTreeNNEvaluationEvaluator() {
23     }
24
25     /**
26      * Evaluate the given expression.
27      *
28      * @param exp
29      *      the {@code XMLTree} representing the expression
30      * @return the value of the expression
31      * @requires <pre>
32      * [exp is a subtree of a well-formed XML arithmetic expression] and
33      * [the label of the root of exp is not "expression"]
34      * </pre>
35      * @ensures evaluate = [the value of the expression]
36      */
37     private static NaturalNumber evaluate(XMLTree exp) {
38         assert exp != null : "Violation of: exp is not null";
39
40         NaturalNumber solution = new NaturalNumber2(0);
41
42         // digit
43         if (exp.label().equals("number")) {
44             NaturalNumber temp = new NaturalNumber2(
45                 exp.attributeValue("value"));
46             solution.copyFrom(temp);
47         }
48         // +
49         if (exp.label().equals("plus")) {
50             solution = evaluate(exp.child(0));
51             solution.add(evaluate(exp.child(1)));
52         }
53         // -
54         if (exp.label().equals("minus")) {
55             solution = evaluate(exp.child(0));
56             NaturalNumber second = evaluate(exp.child(1));
57             if (second.compareTo(solution) > 0) {
58                 Reporter.fatalErrorToConsole("Negative natural number");
59             }
60             solution.subtract(second);
61         }
62         // *
63         if (exp.label().equals("times")) {
64             solution = evaluate(exp.child(0));
65             solution.multiply(evaluate(exp.child(1)));
66         }
67     }
68 }
```

```
66     }
67     // "/"
68     if (exp.label().equals("divide")) {
69         solution = evaluate(exp.child(0));
70         NaturalNumber denominator = evaluate(exp.child(1));
71         if (denominator.canConvertToInt() && denominator.toInt() == 0) {
72             Reporter.fatalErrorToConsole("Error: Dividing by zero");
73         }
74         solution.divide(denominator);
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 }
```