```
 1 import components.simplereader.SimpleReader;
 2 import components.simplereader.SimpleReader1L;
 3 import components.simplewriter.SimpleWriter;
 4 import components.simplewriter.SimpleWriter1L;
 5 import components.xmltree.XMLTree;
 6 import components.xmltree.XMLTree1;
 7
 8 /**
 9  * Program to evaluate XMLTree expressions of {@code int}.
10  *
11  * @author Ansh Pachauri
12  *
13  */
14 public final class XMLTreeIntExpressionEvaluator {
15
16     /**
17      * Private constructor so this utility class cannot be
   instantiated.
18      */
19     private XMLTreeIntExpressionEvaluator() {
20     }
21
22     /**
23      * Evaluate the given expression.
24      *
25      * @param exp
26      *            the {@code XMLTree} representing the
   expression
27      * @return the value of the expression
28      * @requires <pre>
29      * [exp is a subtree of a well-formed XML arithmetic
   expression]  and
30      *  [the label of the root of exp is not "expression"]
31      * </pre>
32      * @ensures evaluate = [the value of the expression]
33      */
34     private static int evaluate(XMLTree exp) {
35         assert exp != null : "Violation of: exp is not null";
36
```

```
37          // TODO - fill in body
38          int noReOccur = 0;
39          // when the node is a number
40          if (exp.label().equals("number")) {
41              noReOccur =
  Integer.parseInt(exp.attributeValue("value"));
42              // when the node is not a number
43          } else {
44              String action = exp.label();
45
46              XMLTree one = exp.child(0);
47              XMLTree two = exp.child(1);
48
49              noReOccur = evaluate(one);
50              // calculating the expression
51              if (action.equals("plus")) {
52                  noReOccur += evaluate(two);
53              } else if (action.equals("divide")) {
54                  noReOccur /= evaluate(two);
55              } else if (action.equals("multiply")) {
56                  noReOccur *= evaluate(two);
57              } else {
58                  noReOccur -= evaluate(two);
59              }
60          }
61          return noReOccur;
62      }
63
64      /**
65       * Main method.
66       *
67       * @param args
68       *             the command line arguments
69       */
70      public static void main(String[] args) {
71          SimpleReader in = new SimpleReader1L();
72          SimpleWriter out = new SimpleWriter1L();
73
74          out.print("Enter the name of an expression XML file: ");
```

```
75          String file = in.nextLine();
76          while (!file.equals("")) {
77              XMLTree exp = new XMLTree1(file);
78              out.println(evaluate(exp.child(0)));
79              out.print("Enter the name of an expression XML file:
  ");
80              file = in.nextLine();
81          }
82
83          in.close();
84          out.close();
85      }
86
87  }
88
```