```java
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 Put your name here
 *
 */
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";

        // initializing first, which becomes the result of the
subexpression
        NaturalNumber first = new NaturalNumber2();
```

```
42
43          // if the root of exp is an operation, recursive call must
    take place
44          if (!exp.hasAttribute("value")) {
45              String op = exp.label();
46
47              // recurisve call to evalute both children
48              first = evaluate(exp.child(0));
49              NaturalNumber second = evaluate(exp.child(1));
50
51              // using NN methods according to operation name
52              if (op.equals("plus")) {
53                  first.add(second);
54              } else if (op.equals("minus")) {
55                  // reports error if expression would violate
    subtract's precondition
56                  if (first.compareTo(second) < 0) {
57                      Reporter.fatalErrorToConsole(
58                              "Subtraction may not result in a
    negative");
59                  }
60                  first.subtract(second);
61              } else if (op.equals("times")) {
62                  first.multiply(second);
63              } else {
64                  // reports error if expression would violate
    divides's precondition
65                  if (second.isZero()) {
66                      Reporter.fatalErrorToConsole("Cannot divide by
    0");
67                  }
68                  first.divide(second);
69              }
70
71          } else {
72              // subExpression becomes the number and simply returns
    itself as an NN
73              first = new
    NaturalNumber2(exp.attributeValue("value"));
74          }
75
76          return first;
77      }
78
```

```
 79     /**
 80      * Main method.
 81      *
 82      * @param args
 83      *            the command line arguments
 84      */
 85     public static void main(String[] args) {
 86         SimpleReader in = new SimpleReader1L();
 87         SimpleWriter out = new SimpleWriter1L();
 88
 89         out.print("Enter the name of an expression XML file: ");
 90         String file = in.nextLine();
 91         while (!file.equals("")) {
 92             XMLTree exp = new XMLTree1(file);
 93             out.println(evaluate(exp.child(0)));
 94             out.print("Enter the name of an expression XML file:
    ");
 95             file = in.nextLine();
 96         }
 97
 98         in.close();
 99         out.close();
100     }
101
102 }
103
```