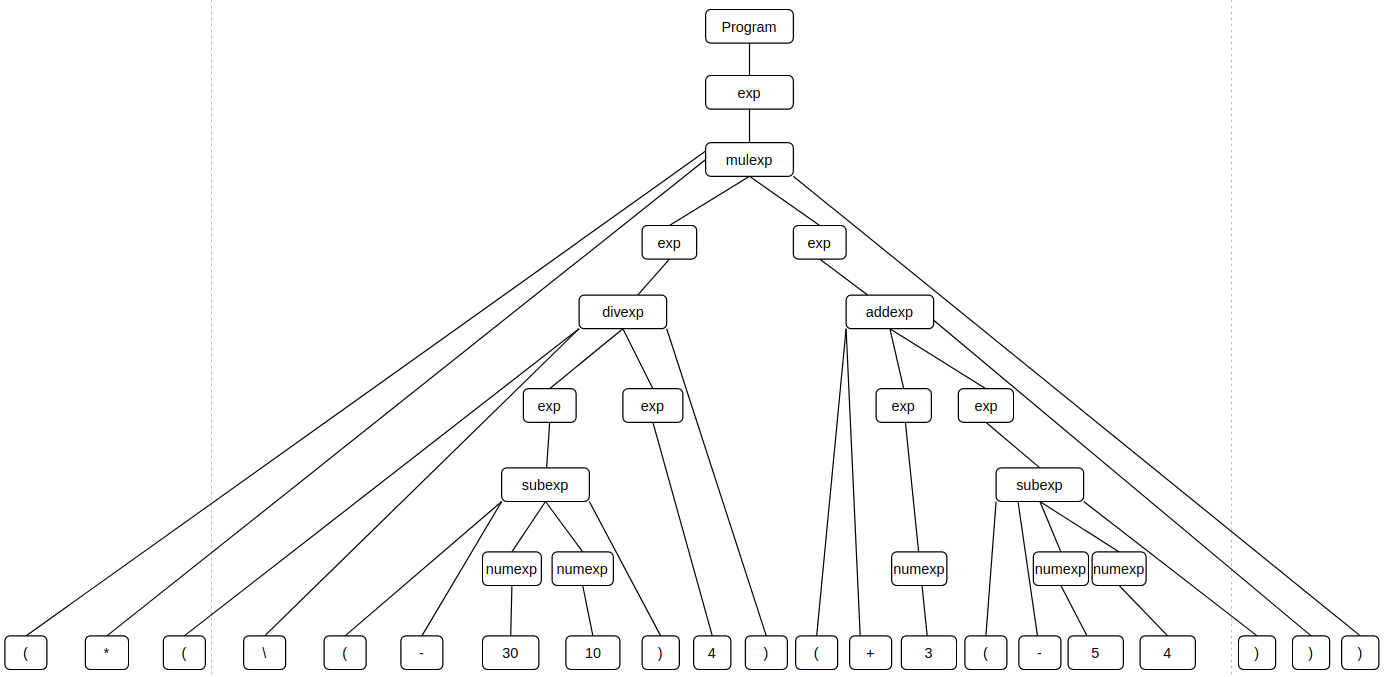
1.

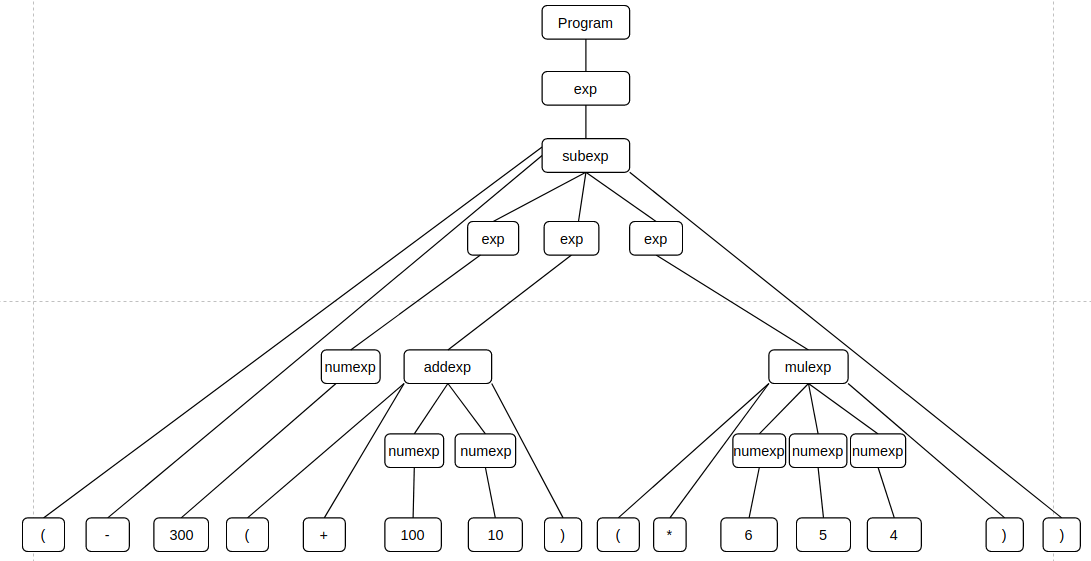
(+ 2 (\* 2 (+ 100 70)))

(+ (\* 2 2) (\* 2 169))

2.



3.



3.

a. The AST class represents expressions in the abstract syntax tree manipulated by this interpreter. The Evaluator class is used to compute a value of the arithmetic expression from an AST.

b. Program ”(+ 1 2 3)” can manually be constructed as follows.

|  |
| --- |
| Exp exp1 =newNumExp(1);  Exp exp2 =newNumExp(2);  Exp exp3 =newNumExp(3);  List<Exp> expList =new ArrayList<Exp>();  expList.add(exp1);  expList.add(exp2);  expList.add(exp3);  AddExp addExp =new AddExp(expList);  Program prog =new Program(addExp);  Evaluator eva =new Evaluator();  porg.accept(eva); |

|  |
| --- |
| @Override  public Value visit(AddExp e) {  List<Exp> operands = e. all ();  double result = 0;  for(Exp exp: operands) {  NumVal intermediate = (NumVal) exp.accept(this);  result += intermediate.v();  }  return new NumVal(result);  }  @Override  public Value visit(NumExp e) {  return new NumVal(e.v());  }  @Override  public Value visit(Program p) {  return (Value) p.e().accept(this);  } |

The call to method accept *porg.accept(eva)* causes method visit (Program p) in class Evaluator to run. Then invokes method accept on the object AddExp, which in turn causes method visit (AddExp e) in class Evaluator to run. This method iterates over the component expression exp1, exp2, and exp3 and invokes method accept on each object. That in turn causes method visit (NumExp e) in class Evaluator to run three times returning result strings ”1”, ”2”, and ”3” respectively. Consequently, return value of the method visit (AddExp e) is the string ”(+ 1 2 3)”.

c.

The class must provide the type of the value it will generate oo implement the visitor pattern.

For the AST class, each sub-class implemented a method accept that takes an object of type Visitor as a parameter and invokes method visit on that object. An interface of Visitor should be declared in AST class. This interface provides a method visit for each sub-AST node. Sub-AST traversal functionalities can be implemented by extending the Visitor interface.

For the Evaluator, the value is the class Value. The implementation must override visit methods for each type of node (or subclass) of the AST node types. For each sub-expression in each expression, the visitor must call the method accept, so these sub-expressions are evaluated.