Esteban Sanchez

Cmps 342

1. Arithlang Compute to 24

1) (*(+(+24)(-816))1)=24

2) (*(+(*210)(÷520)1)

20 4

2. left most Derivation: (* (+ 2(- 5 8))(* 2 1))

Program
→ exp
→ multexp
→ (* exp(exp)+)
→ (* addexp(exp)+
→ (* (+ exp(exp)+) (exp+)
→ (* (+ exp(subexp)+) (exp+)
⇒ (* (+ exp(- exp(exp)+) (exp+)
→ (* (+ exp(- exp(exp)+)) (multexp+)
→ (* (+ exp(- exp(exp+)) (* exp(exp)+)
→ (* (+ numexp(- exp(exp+)) (* exp(exp)+)
→ (* (+digit (- exp(exp+)) (* exp(exp)+)
→* (* (+ 2 (- numexp(numexp)) (* exp(exp)+)
→* (*(+2(- digit(digit)) (* exp(exp)+)
→* (*(+2(- 58)) (* exp(exp)+)
→* (*(+2(- 58)) (* numexp(numexp))
→* (*(+2(- 58)) (* 2 1))

2 Right (÷34(*.42)(+34 67))

→ Program

→ exp

→ divexp

→ (/ exp exp exp)

→ (/ exp exp addexp)

→ (/ exp exp (+ exp (exp)+)

→ (/ exp multexp (+ exp (exp)+)

→ (/ exp (* exp (exp)+)(+ exp (exp)+)

→ (/ exp (* exp (exp)+)(+ exp (numexp)+))

→ (/ exp (* exp (exp)+)(+ exp (number))

→ (/ exp (* exp (exp)+)(+ exp digitnonzero (digit))

→ (/ exp (* exp (exp)+)(+ exp ' 67))

→ (/ exp (* exp 'exp)(+ number 67))

→ (/ exp (* exp exp)(* digitnonzero (digit)+ 67))

→ (/ exp (* exp exp)(+ 34 67))

→ (/ exp (* numexp numexp)(+34 67))

→ (/ exp (* number number)(+34 67))

→ (/ exp (* digit digit)(+34 67))

→ (/ exp (* 4 2)(+34 67))

→ (/ numexp (* 4 2)(+34 67))

→ (/ number (* 4 2)(+34 67))

→ (/ digit nonzero (* 4 2)(+34 67))

→ (/ 34 (* 4 2)(+34 67))

3. The Purpose of the two classes and how the visitor class works.

Evaluator:

This is taking the Arithlang We learned in class. They Are being implemented in Java. The logic on how it works Being Done Respectfully in Javier

Then Formatter

The Formatter is taking the logic from the Evaluator, running it through AST, which Then it is able to take the information from The evaluator and prints its Correct Formed

Visitor is essentially Polymorphism keeping the code neat and its able to keep adding and form of exp without hassel