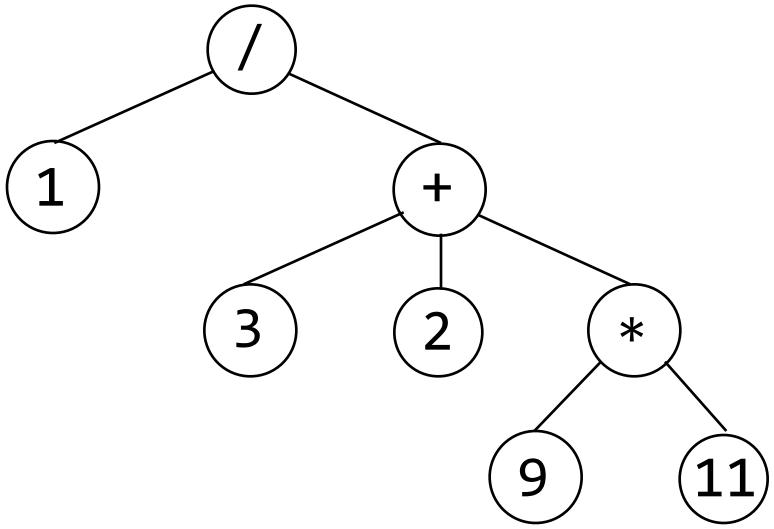
Arithmetic Expression Trees: Visitor Homework

Prelude

Arithmetic expressions can be represented as a tree. For example, consider the tree...



which is just *one* of many possible representations of the *same* arithmetic expression:

1 / (3 + 2 + (9 * 11)) ⇐ *infix form*

(/ 1 (+ 3 2 (* 9 11))) ⇐ *lisp s-expression*

0.0096153 ⇐ *evaluated*

⇐ *textual tree*

```
[/]
 +---[1]
 +---[+]
      +---[3]
      +---[2]
      +---[*]
           +---[9]
           +---[11]
```

(continued on the next page)

Homework

Part I (aka, the first part):

Implement the classes necessary to represent arithmetic expressions as trees

Part II (aka, the fun part):

Implement four (*4*) kinds of **Visitor**s over arithmetic expression trees
that after traversing an expression tree, when queried

- returns a String representing the expression in *infix* form
- returns a String representing the expression as a *lisp s-expression*
- return a double representing the tree's *value*
- returns a String representing the expression in *text tree* form

Aside: the previous *text tree* example was "simple"

Here's another expression and the corresponding *text tree*

11 + ((1 / 2) * ((3 + -5) - 45)) + -23

```
[+]
 +---[11]
 +---[*]
 |    +---[/]
 |    |    +---[1]
 |    |    +---[2]
 |    +---[-]
 |         +---[+]
 |         |    +---[3]
 |         |    +---[-5]
 |         +---[45]
 +---[-23]
```

From the 2 examples, you are to ***deduce the formatting rules*** *for "text trees"*

Part III (aka, the proof-is-in-the-pudding part):

make multiple test trees and prove your **Visitor**s are correct