Homework Assignment #3

Data Structures Fall 2008
Eugene Weinstein
Expression Trees

- Trees representing algebraic expressions
  - Operators at inner nodes
    - Specify an operation to be done on their children
  - Operands at leaves
  - Q: Are expression trees always binary?
  - A: For our purposes, yes. In general, depends on operation set (e.g., min, max could be non-binary)
Homework Details

- We have the code to convert from infix to postfix
- This assignment
  - Implement algorithm to convert from postfix to expression tree
  - Print inorder, preorder, postorder traversals
- Details
  - Need to read expressions from text window
    - i.e., no JOptionPane
  - Can use Scanner or InputStreamReader
    - (use examples available in Java API pages)
More Details

- Continue until user exits program with Ctrl-C
- \texttt{while(true)}
  - \texttt{Read infix expression}
  - \texttt{Convert to postfix}
  - \texttt{Convert to expression tree}
  - \texttt{Print traversals}
- Expression tree building algorithm uses stack of trees
  - \texttt{Do not push Strings, Integers, etc. The only objects getting pushed should be of type Node}
  - \texttt{Remember type safety!}
  - \texttt{Should complete without exceptions if input postfix expression is well-formed}
- Algorithm is in Weiss, pages 109-112
Postfix to Tree Pseudocode

- while (more input)
  - token ← next input symbol
  - if isOperand(token)
    - n ← new leaf node from token
    - stack.push(n)
  - else if isOperator(token)
    - n1 ← s.pop
    - n2 ← s.pop
    - n3 ← new node from token with left child n2, right child n1
    - stack.push(n3)
- return stack.pop
Traversals Review

- Tree traversal: visit every node of the tree and do something at each node
- Implementation: recursive calls to children (subtrees)
- Inorder traversal:
  - `inOrder(Node n)`
    - if (n != null)
      - 1. `inOrder(n.left)`
      - 2. visit n (println, etc.)
      - 3. `inOrder(n.right)`
- To get different traversals, rearrange line order
  - Inorder: 1, 2, 3
  - Preorder: 2, 1, 3
  - Postorder: 1, 3, 2
Inorder, preorder and postorder traversals of expression tree yield infix, postfix, and prefix versions of the expression, respectively.

E.g., for the above tree, we get:
- Preorder: *2+35
- Postorder: 235+*
- Inorder: 2*3+5 ---- What's the problem?
Parentheses

• The correct expression is 2*(3+5)
• Need to generate parentheses with infix expression
• It's okay to overparenthesize, e.g., (2*(3+5))
• Hint: print parentheses around the expression corresponding to each subtree...
  o A bit more detail is in the book
• Remember: still no parentheses in postfix or prefix expressions!