Abstract Syntax Trees

- The parser’s output is an abstract syntax tree (AST) representing the grammatical structure of the parsed input.
- ASTs represent only semantically meaningful aspects of input program, unlike concrete syntax trees which record the complete textual form of the input.
  - There’s no need to record keywords or punctuation like (), ;, else.
  - The rest of compiler only cares about the abstract structure.
MiniJava AST Node Classes

Each node in an AST is an instance of an AST class
- e.g. If, Assign, Plus, VarDecl, etc.

Each AST class declares its own instance variables holding its AST subtrees
- If has Exp, and 2 Statement
- Assign has Identifier and Exp
- Plus has 2 Exp
- VarDecl has Type and Identifier
AST Class Hierarchy

- AST classes are organized into an inheritance hierarchy based on commonalities of meaning and structure.

- Each "abstract non-terminal" that has multiple alternative concrete forms will have an abstract class that’s the superclass of the various alternative forms.
  - Statement is abstract superclass of If, Assign, etc.
  - Exp is abstract superclass of Plus, IdentifierExp, etc.
  - Type is abstract superclass of IntegerType, IdentifierType, etc.
Productions

- All of the form:

  \[ \text{LHS} ::= \text{RHS}_1 \{ : \text{Java code 1} : \} \]
  \[ \ | \text{RHS}_2 \{ : \text{Java code 2} : \} \]
  \[ \ | \ldots \]
  \[ \ | \text{RHS}_n \{ : \text{Java code } n : \}; \]

- Can label symbols in RHS with :\text{var} suffix to refer to its result value in Java code

  - \text{varleft} is set to line in input where \text{var} symbol was
Productions (cont.)

- **Example**

  \[
  \text{Exp ::= Exp:arg1 PLUS Exp:arg2} \\
  \quad \{ : \text{RESULT} = \text{new AddExp}(\text{arg1}, \text{arg2}, \text{arg1left}); : \}
  \]

  \[
  \mid \text{INT LITERAL:value} \{ : \text{RESULT} = \text{new IntegerLiteral( value, valuele}\text{ft}); : \}
  \]

  \[
  \mid \text{Exp:rcvr PERIOD Identifier:message OPEN_PAREN ExpList:args CLOSE_PAREN} \\
  \quad \{ : \text{RESULT} = \text{new Call( rcvr, message, args, rcvrle}\text{ft}); : \}
  \]

  \[
  \mid \ldots ;
  \]