CS536

Java CUP

Last Time

- What do we want?
  - An AST
- When do we want it?
  - Now!

This Time

- A little review of ASTs
- The philosophy and use of a *Parser Generator*

Translating Lists

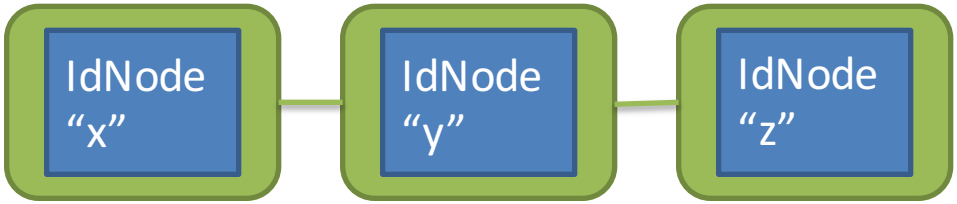
CFG

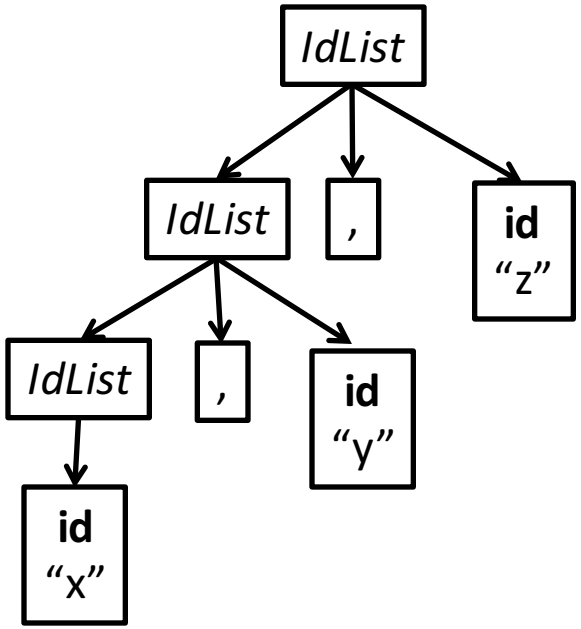
IdList -> **id**
| *IdList* **comma id**

Input

x, y, z

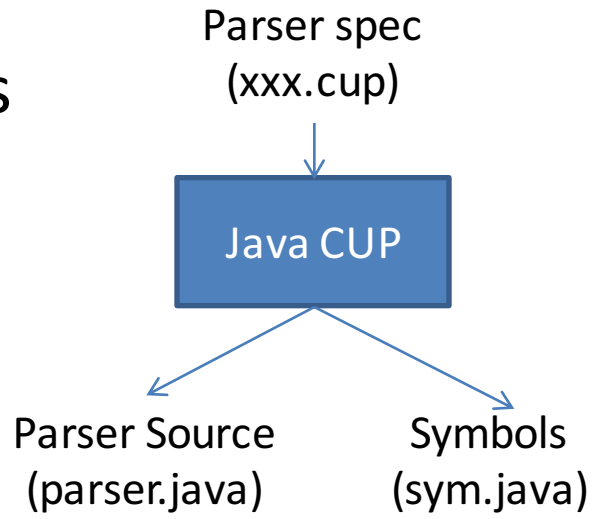
AST





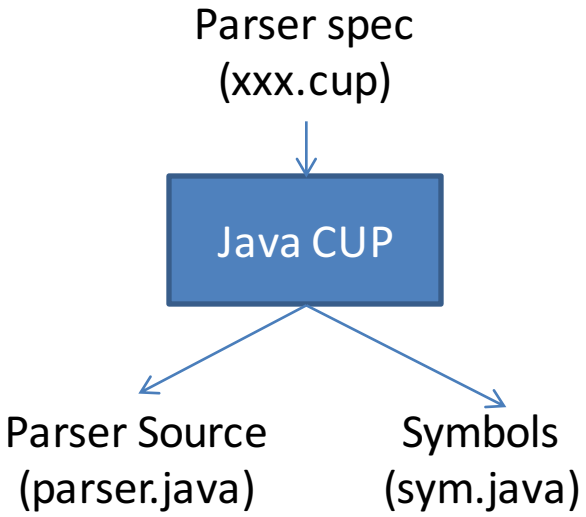
Parser Generators

- Tools that take an SDT spec and build an AST
  - YACC: Yet Another Compiler Compiler
  - Java CUP: Constructor of Useful Parsers
- Conceptually similar to JLex
  - Input: Language rules + actions
  - Output: java code



Java CUP

- Parser.java
  - Constructor takes arg of type Yylex
  - Contains parse method
    - return: Symbol whose value contains translation of root nonterm
  - Uses output of JLex
    - Depends on scanner and TokenVals
  - Uses defs of AST classes (ast.java)



Java CUP Input Spec

- Terminal & nonterminal declarations
- Optional precedence and associativity declarations
- Grammar with rules and actions

Grammar rules

```
Expr ::= intliteral
       |  id
       |  Expr plus Expr
       |  Expr times Expr
       |  lparens Expr rparens
```

Terminal and Nonterminals

```
terminal intliteral;
terminal id;
terminal plus;
terminal times;
terminal lparen;
terminal rparen;
non terminal Expr;
```

Precedence and Associativity

```
precedence left plus;
precedence left times;
prededence nonassoc less;
```

lowest precedence

Java CUP Example

- Assume ExpNode Subclasses
  - `PlusNode`, `TimesNode` have 2 children for operands
  - `IdNode` has a String field
  - `IntLitNode` has an int field
- Assume Token classes
  - `IntLitTokenVal` with field intVal for int literal token
  - `IdTokenVal` with field idVal for identifier token

Step 1: Add types to terminals

```
terminal IntLitTokenVal intliteral;
terminal IdTokenVal id;
terminal plus;
terminal times;
terminal lparen;
terminal rparen;

non terminal ExpNode expr;
```

Java CUP Example

```
Expr ::= intliteral
         {:

         :}
       | id
         {:

         :}
       | Expr plus Expr
         {:

         :}
       | Expr times Expr
         {:

         :}
       | lparen Expr rparen
         {:

         :}
       ;
```

Java CUP Example

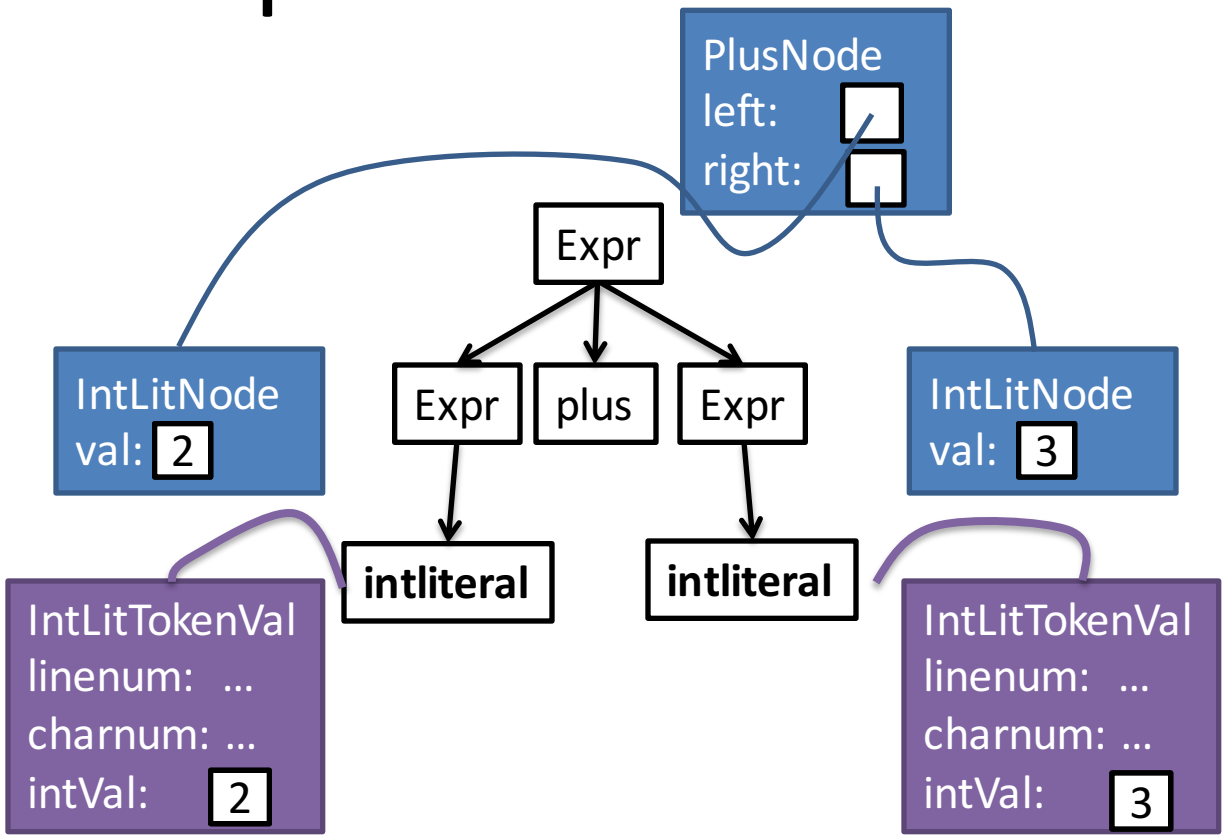
```
Expr ::= intliteral:i
         {:
            RESULT = new IntLitNode(i.intVal);
         :}
       |  id
         {:

         :}
       |  Expr plus Expr
         {:

         :}
       |  Expr times Expr
         {:

         :}
       |  lparen Expr rparen
         {:

         :}
       ;
```

Java CUP Example

```
Expr ::= intliteral:i
         {:
            RESULT = new IntLitNode(i.intVal);
         :}
       |  id:i
         {:
             RESULT = new IdNode(i.idVal);
         :}
       |  Expr:e1 plus Expr:e2
         {:
             RESULT = new PlusNode(e1,e2);
         :}
       |  Expr:e1 times Expr:e2
         {:
             RESULT = new TimesNode(e1,e2);
         :}
       |  lparen Expr:e rparen
         {:
             RESULT = e;
         :}
      ;
```

Java CUP Example

```
Input: 2+ 3
```



Purple = Terminal Token (Built by Scanner)
Blue = Symbol (Built by Parser)

Java CUP Demo