# Introduction to Compiler Tools

# Compiler Tools

- lex/yacc (flex/bison)
  - generate bottom-up parsers which are often hard to understand, hard to debug and give weird error messages
- antlr4 (Java) and antlr3
  - produces a recursive descent top-down parser which is much easier to understand, you can actually debug it easily, you can only use subrules for a parse operation (e.g. just parse expressions instead of the full language)
- SableCC (Java)
- Parse::EBNF, Parse::Yapp and Marpa (Perl)
- and SimpleParse (Python)

# ANTLR

- ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files.
- The latest version of ANTLR is 4.5, released January 22, 2015. As of 4.5, ANTLR has a Java, C#, JavaScript, Python2, Python3 targets (and alternate C# target). C++ is next on deck [no completion estimate].

# Antlr Install

- https://theantlrguy.atlassian.net/wiki/display/ANTLR4/Getting+Started+with+ANTLR+v4
  - 1. Download antlr
  - 2. Add antlr-4.5-complete.jar to your CLASSPATH
  - 3. Create aliases (Optional)
  - 4. Test your install

# Eclipse Plugin

- ANTLR 4.x
- Advanced Syntax Highlighting ([even for target language](https://raw.github.com/jknack/antlr4ide/master/updates/screenshots/target-language-highlighting.png))
- Automatic Code Generation (on save)
- Manual Code Generation (through External Tools menu)
- Code Formatter (Ctrl+Shift+F)
- [Syntax Diagrams as HTML](http://jknack.github.io/antlr4ide/Java/Javav4.g4.html)
- Live Parse Tree evaluation
- Advanced Rule Navigation between files (F3 or Ctrl+Click over a rule)
- Quick fixes

# Eclipse Plugin Install

- Download
  - plug-ins for Intellij, NetBeans, and Eclipse
  - http://www.antlr.org/tools.html
- Install plugin for Elicpse
  - Following instructions in README.md

# AntlrWorks

- ANTLRWorks 2 provides editor hints to alert a developer to potential bugs or performance issues in a grammar which the grammar compiler does not currently detect. Since this type of static analysis is unique to each language, we work with both language creators and users to provide analysis features most likely to detect problematic code early in the development cycle. Current analysis features for grammars in ANTLRWorks include the following.a
  - Errors and warnings reported by the ANTLR 4 tool are displayed "live" in the editor
  - Identifying implicit token definitions in a parser (probable bug)
  - Factor label out of set (performance hint)
  - Group set elements (performance hint)

# Anltr Lexicon - Comments

- Comments
  - There are single-line, multiline, and Javadoc-style comments.

```
/** This grammar is an example illustrating the three kinds
 * of comments.
 */
grammar T;
/* a multi-line
  comment
*/
/** This rule matches a declarator for my language */
decl : ID ; // match a variable name
```

# Anltr Lexicon - Identifiers

- Identifiers
  - Token names always start with a capital letter and so do lexer rules as defined by Java’s Character.isUpperCase method.
  - Parser rule names always start with a lowercase letter (those that fail Character.isUpperCase).
  - The initial character can be followed by uppercase and lowercase letters, digits, and underscores. Here are some sample names

```
ID, LPAREN, RIGHT_CURLY // token names/rules
expr, simpleDeclarator, d2, header_file  // rule names
```

# Anltr Lexicon - Literals

- ANTLR does not distinguish between character and string literals as most languages do. All literal strings one or more characters in length are enclosed in single quotes such as ' ;' , ' if' , ' >=' , and ' \ ' ' (refers to the one-character string containing the single quote character).
- Literals never contain regular expressions.

# Anltr Lexicon - Action

- Actions are code blocks written in the target language. You can use actions in a number of places within a grammar, but the syntax is always the same: arbitrary text surrounded by curly braces. You don’ t need to escape a closing curly character if it’ s in a string or comment: {"}"} or {/*}*/;}. If the curlies are balanced, you also don’ t need to escape }: {{...}}. Otherwise, escape extra curlies with a backslash: {\{} or {\}}. The action text should conform to the target language as specified with thelanguage option.
- Embedded code can appear in: @header and @members named actions, parser and lexer rules, exception catching specifications, attribute sections for parser rules (return values, arguments, and locals), and some rule element options (currently predicates).
- The only interpretation ANTLR does inside actions relates to grammar attributes; see Token Attributes and Chapter 10, Attributes and Actions. Actions embedded within lexer rules are emitted without any interpretation or translation into generated lexers.

# Anltr Lexicon - Keywords

- import, fragment, lexer, parser, grammar, returns, locals, throws, catch, finally, mode, options, tokens
- Also, although it is not a keyword, do not use the word rule as a rule name. Further, do not use any keyword of the target language as a token, label, or rule name. For example, rule if would result in a generated function called if.