1. grammar FILE\_NAME;
2. import GRAMAR\_NAME:
3. @header{

package PACKAGE\_NAME;

}

1. Lexer / Parser : The most basic rule is just a rule name followed by a single alternative terminated with a semicolon:
2. Lexer
   * https://theantlrguy.atlassian.net/wiki/display/ANTLR4/Lexer+Rules
   * Start with uppercase letter
   * lexer grammar FILE\_NAME;
   * **Fragment**
     + Can also define rules that are not tokens but rather aid in the recognition of tokens. These fragment rules do not result in tokens visible to the parser:
   * **Commands**
     + Skip: A 'skip' command tells the lexer to get another token and throw out the current text.
     + Channel(x)
3. Parser
   * https://theantlrguy.atlassian.net/wiki/display/ANTLR4/Parser+Rules
   * Start with lowercase letters
   * parser grammar FILE\_NAME;
   * **Alternative Labels**
     + Can get more precise parse-tree listener events by labeling the outermost alternatives of a rule using the # operator. All alternatives within a rule must be labeled, or none of them.
   * **Rule Context Objects**
     + ANTLR generates methods to access the rule context objects (parse tree nodes) associated with each rule reference. For rules with a single rule reference, ANTLR generates a method with no arguments.

inc : e '++' ;

public static class IncContext extends ParserRuleContext {

public EContext e() {}

// return context object associated with e

}

* **Rule Element Labels**
* can label rule elements using the = operator to add fields to the rule context objects:

stat: 'return' value=e ';' # Return

| 'break' ';' # Break

;

public static class ReturnContext extends StatContext {

public EContext value;

}