**Парсер:**

1. Общий вид файла с описанием парсера:  
     
   **lexer** LexerClassName;  
   **start** startRule;  
    **header** {  
   package your.favorite.package;  
   import java.util.Collections;  
   };  
     
   startRule [int a, int b, int c] **returns** [String s, boolean result]  
    -> rule1 TOKEN1   
    | rule2 rule3 {$s = “” + a + b + c; $result = true;}  
    ;  
   ...
2. Аналогично ANTLR, имена правил начинаются с маленькой буквы, токенов – с большой.
3. Есть наследуемые и синтезируемые атрибуты:   
   rule1 [int a, int b, int c] **returns** [String s, int d] -> …;
4. Используя любые атрибуты, необходимо помечать их знаком $:  
   rule1 [int x] -> rule2[$x + 2];
5. Знаки доллара корректно обрабатываются, если они находятся внутри строкового или символьного литерала:  
   rule1 [int x] -> rule2[“Value of $x = “ + $x + “, some dollars: “ + ‘$’ + ‘$’ + ‘$’];
6. Запятые также корректно обрабатываются при передаче параметров:  
   rule1 -> rule2[1, 2, “1, 2, 3”, ‘,’, 3];
7. При передаче параметров поддерживаются вложенные квадратные скобки с правильным балансом, а так же произвольные скобки в строковых и символьных литералах:  
   rule1 -> rule2[“a[0] = “ + a[0] + “, ]]][[“];
8. Чтобы обратиться к тексту токена, необходимо просто добавить $ к имени токена:  
   rule -> NUMBER {System.out.println(“Parsed number: “ + $NUMBER);};
9. В случае одинаковых правил или токенов в определении правила, переменные, которые им соответствуют, называются следующим образом: ruleName, ruleName1, ruleName2, … :  
   rule returns [String s] -> ruleA TOKEN ruleA TOKEN ruleA {$s = $ruleA.s + $TOKEN + $ruleA1.s + $TOKEN1 +$ruleA2.s; }
10. Поддерживаются однострочные и многострочные комментарии  
    // parse list of elems  
    rule1 **returns** [List<String> list]   
     -> /\* initialization \*/ {$list = new ArrayList<>();}   
     | /\* add \*/ elem rule1 {$list = $rule1.list; $list.add($elem.a);}  
     ;
11. Содержимое блока header помещается в начало сгенерированного файла.

**Лексер:**

1. Пример лексера:  
     
   **package** your.favorite.package;  
     
   NUMBER -> ‘[0-9]+';  
   ID -> ‘[a-zA-Z\_][a-zA-Z0-9\_]\*];  
   WS -> ‘[ \t\r\n]+’ :: skip;
2. Общий вид файла с описанием лексера:  
     
   TOKEN\_NAME -> ‘regex for java.util.regex.\*’;  
   MOD\_TOKEN -> ‘regex’ :: modificator1, modificator2, ..., modificatorn;
3. Правда, поддерживается всего один модификатор skip, который пропускает этот токен и пытается взять следующий:  
   WS -> ‘[ \t\r\n]+’ :: skip;  
   COMMENT -> ‘//[^\n]\*\n’ :: skip, skip, skip; // lol
4. Ну и да, в лексере тоже поддерживаются /\*…\*/ и //… комментарии.
5. Если в файле есть строка “package a.b.c;”, то аналогичная строка вставится в начало сгенерированного файла.