Step-by-Step: Designing a Parser using ANTLR + Java in Eclipse

**Example 1: Arithmetic Expression Parser**

**Step 1: Setup Eclipse with ANTLR**

1. **Install ANTLR Plugin (Optional but helpful)**:
   * Go to Help > Eclipse Marketplace.
   * Search for: ANTLR v4 IDE.
   * Install it and restart Eclipse.
2. **Create a new Java Project**:
   * File > New > Java Project.
   * Name it something like ANTLRParserDemo.
3. **Add ANTLR Runtime Library**:

* Right-click your project → Build Path > Configure Build Path.
* Go to **Libraries** → Add External JARs.
* Download antlr-4.X-complete.jar from ANTLR site.
* Add it to the build path.

**Step 2: Write Your Grammar File (.g4)**

1. Right-click src folder → New > File → Name it Expr.g4 (or your grammar name).
2. Sample grammar (arithmetic expression parser):

grammar Expr;

// Parser rules

prog: stat+ ;

stat: expr NEWLINE # printExpr

| ID '=' expr NEWLINE # assign

| NEWLINE # blank

;

expr: expr op=('\*'|'/') expr # MulDiv

| expr op=('+'|'-') expr # AddSub

| INT # int

| ID # id

| '(' expr ')' # parens

;

// Lexer rules

ID : [a-zA-Z]+ ;

INT : [0-9]+ ;

NEWLINE : [\r\n]+ ;

WS : [ \t]+ -> skip ;

**Step 3: Generate Parser and Lexer**

You can do this in two ways:

**Option 1: Use ANTLR Plugin (Easier)**

* Right-click the .g4 file → Run As > Generate ANTLR Recognizer.
* This will create Java classes: ExprLexer.java, ExprParser.java, etc.

**Option 2: Use ANTLR Tool via Command Line**

* If plugin isn't available:

java -jar antlr-4.X-complete.jar -Dlanguage=Java Expr.g4

Run this from the terminal in the directory containing Expr.g4.

**Step 4: Write Java Code to Use the Parser**

Create a new Java class, e.g., Main.java:

import org.antlr.v4.runtime.\*;

import org.antlr.v4.runtime.tree.\*;

public class Main {

public static void main(String[] args) throws Exception {

// Input

String input = "a = 3 + 4\n";

// Create input stream from string

CharStream cs = CharStreams.fromString(input);

// Lexer

ExprLexer lexer = new ExprLexer(cs);

CommonTokenStream tokens = new CommonTokenStream(lexer);

// Parser

ExprParser parser = new ExprParser(tokens);

// Parse the input using the top rule (prog)

ParseTree tree = parser.prog();

// Print the parse tree

System.out.println(tree.toStringTree(parser));

}

}

**Step 5: Run the Parser**

Right-click Main.java → Run As > Java Application.

You should see the parse tree printed in the console.

**(prog (stat (expr (ID a) = (expr (expr (INT 3)) + (expr (INT 4)))) \n))**

**Example 2: Boolean Expression Parser**

**Step 1: Create Grammar File (BoolExpr.g4)**

Create a file called BoolExpr.g4 in your src folder.

**grammar BoolExpr;**

**prog: expr EOF ;**

**expr: expr 'OR' expr # OrExpr**

**| expr 'AND' expr # AndExpr**

**| 'NOT' expr # NotExpr**

**| '(' expr ')' # ParenExpr**

**| BOOL # BoolAtom**

**;**

**BOOL: 'true' | 'false' ;**

**WS: [ \t\r\n]+ -> skip ;**

**This grammar supports logical expressions using:**

* **AND, OR, NOT**
* **Boolean literals true, false**
* **Parentheses ()**

**Step 2: Generate Lexer + Parser Classes**

Option 1: Using Eclipse ANTLR Plugin:

* Right-click BoolExpr.g4 → Run As > Generate ANTLR Recognizer.

Option 2: Manually (if plugin not available):  
Run this from terminal:

java -jar antlr-4.X-complete.jar -Dlanguage=Java BoolExpr.g4

This will generate:

* BoolExprLexer.java
* BoolExprParser.java
* BoolExprBaseVisitor.java
* BoolExprBaseListener.java

**Step 3: Write Java Code to Use the Parser**

Create a new file: BoolMain.java

import org.antlr.v4.runtime.\*;

import org.antlr.v4.runtime.tree.\*;

public class BoolMain {

public static void main(String[] args) throws Exception {

// Input string

String input = "true AND false OR NOT false";

// Create input stream

CharStream cs = CharStreams.fromString(input);

// Create lexer and token stream

BoolExprLexer lexer = new BoolExprLexer(cs);

CommonTokenStream tokens = new CommonTokenStream(lexer);

// Create parser and parse starting from 'prog'

BoolExprParser parser = new BoolExprParser(tokens);

ParseTree tree = parser.prog();

// Print parse tree (text format)

System.out.println(tree.toStringTree(parser));

}

}

**Step 3: Run the Parser**

Right-click BoolMain.java → Run As > Java Application.

You should see the parse tree printed in the console.

For input: true AND false OR NOT false

You’ll get something like:

(prog (expr (expr (expr true) AND (expr false)) OR (expr NOT (expr false))) <EOF>)

**To Show Tree Graphically**

Add this to BoolMain.java if you want a GUI tree:

import org.antlr.v4.gui.Trees; // import GUI tree utility

// After parsing:

javax.swing.JFrame frame = new javax.swing.JFrame("Boolean Parse Tree");

frame.setDefaultCloseOperation(javax.swing.JFrame.EXIT\_ON\_CLOSE);

javax.swing.JScrollPane scrollPane = new javax.swing.JScrollPane(Trees.inspect(tree, parser));

frame.add(scrollPane);

frame.setSize(600, 400);

frame.setVisible(true);