در این پروژه قصد داریم تا با استفاده از لکسر JFLEX و پارسر CUP و متصل کردن این دو به یکدیگر یک تحلیل گر نحوی برای زبانی شبیه به جاوا ایجاد کنیم.

برای اینکار ابتدا نیاز است تا JFLEX را نصب کنیم.

سپس برای طراحی لکسر باید فایلی داشته باشیم معمولا با پسوند flex. یا jflex. که در آن قوانین مختلف را بر اساس رگیولار اکسپرشن هایی که توسط ورژن JFLEX ای که داریم از آن استفاده می کنیم ساپورت می شود پیاده سازی می کنیم. شمای کلی این فایل به صورت زیر خواهد بود.

```
LineTerminator = \r|\n|\r\n
WhiteSpace     = {LineTerminator} | [ \t\f]

NUM = [0-9]+
IDENT = [A-Za-z_][A-Za-z_0-9]*
STRING = \"([^\\\"]|\\.)*\"
```

همچنین در آن بخشی برای مشخص کردن خروجی بر اساس قوانین مختلف خواهیم داشت.

یکی از تفاوت های فاز اول پروژه با فاز دوم آن این است که در این فاز نیاز است تا خروجی هر کدام از قانون ها را به سمبلی که پس از تعریف در CUP در اختیار ما قرار می گیرد متصل کنیم بنابراین قوانین ما به صورت زیر خواهد بود.

```
"("                    { return symbol(LPAREN); }
")"                    { return symbol(RPAREN); }
"{"                    { return symbol(LBRACE); }
"}"                    { return symbol(RBRACE); }
"["                    { return symbol(LBRACK); }
"]"                    { return symbol(RBRACK); }
";"                    { return symbol(SEMICOLON); }
","                    { return symbol(COMMA); }
"."                    { return symbol(DOT); }
```

سپس نیاز است تا فایل گرامر خود را در CUP بسازیم برای این کار فایلی با پسوند cup. ایجاد خواهیم کرد که محتوای آن دو بخش است.

یک بخش مانند تصویر زیر به تعریف ترمینال ها، نان تریمنال ها و دیگر دستورات مورد نیاز اختصاص داده شده است.

```
terminal BOOLEAN; // primitive_type
terminal BYTE, SHORT, INT, LONG, CHAR; // integral_type
terminal FLOAT, DOUBLE; // floating_point_type
terminal LBRACK, RBRACK; // array_type
terminal DOT; // qualified_name
terminal SEMICOLON, MULT, COMMA, LBRACE, RBRACE, EQ, LPAREN, RPAREN, COLON;
terminal PACKAGE; // package_declaration
terminal IMPORT; // import_declaration
terminal PUBLIC, PROTECTED, PRIVATE; // modifier
terminal STATIC; // modifier
terminal ABSTRACT, FINAL, NATIVE, SYNCHRONIZED, TRANSIENT, VOLATILE;
terminal CLASS; // class_declaration
terminal EXTENDS; // super
terminal IMPLEMENTS; // interfaces
terminal VOID; // method_header
terminal THROWS; // throws
terminal THIS, SUPER; // explicit_constructor_invocation
terminal INTERFACE; // interface_declaration
```

حال کافی است با استفاده ازین این تعریفات ترکیبات یا اکسپرشن هایی تعریف کنیم که از آن ها برای تحلیل نحوه استفاده خواهد شد.

```
while_statement ::=
        WHILE LPAREN expression RPAREN statement
    ;
```

مانند استیتمنت وایل که در بالا تعریف شده است و ترکیبی که سازنده حلقه وایل است را نمایش می دهد.

سپس نیاز به یک فایل Main داریم که باید فایل ورودی را گرفته و با استفاده از لکسر و پارسر خروجی دهد.

```java
public class Main {

  public static void main(String argv[]) {

    for (int i = 0; i < argv.length; i++) {
      try {
        System.out.println("Parsing [" + argv[i] + "]")
        Scanner s = new Scanner(new FileReader(argv[i]))
        Parser p = new Parser(s);
        p.parse();

        System.out.println( x: "No errors.");
      } catch (Exception e) {
        e.printStackTrace(System.err);
        System.exit( status: 1);
      }
    }
  }
}
```

این فایل به این صورت تعریف می شود. حال کافی است تا این کلاس ها را اجرا کنیم این کار به کمک اسنیپت کد زیر انجام می شود.

```
jflex/bin/jflex Scanner.flex;
java -jar java-cup-11b.jar -parser "Parser" parser.cup
java -cp .:java-cup-11b.jar java_cup.Main  < parser.cup;
javac -cp .:java-cup-11b.jar Scanner.java Parser.java JavaParser.java;
javac -cp .:java-cup-11b.jar Main.java;
java -cp .:java-cup-11b-runtime.jar Main input.txt;
```

که پس از اجرای این کد ها با پیام زیر روبرو خواهیم شد و کد ما تحلیل نحوی شده است.

```
poorya@Pooryas-MacBook-Air src % ./run.sh
Note: Main.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Parsing [input.txt]
No errors.
```