# Manual Técnico del Analizador Léxico-Sintáctico en Java (parserpy)

El analizador léxico-sintáctico (parserpy) desarrollado en Java es una poderosa herramienta de análisis de gramáticas y estructuras de lenguaje natural. Este manual técnico proporciona una comprensión profunda del funcionamiento del software, su implementación y sus posibles aplicaciones, desarrollado específicamente en un entorno Windows utilizando JDK 15 y el entorno de desarrollo integrado (IDE) NetBeans y es la fase número dos del software parserpy.

## Gramática

La gramática utilizada en el analizador léxico-sintáctico se define formalmente de la siguiente manera:

G = (N, T, P, S)

Donde:

N: Conjunto de símbolos no terminales.

T: Conjunto de símbolos terminales.

P: Conjunto de producciones.

S: Símbolo inicial.

Símbolos no terminales (N):

programa: El programa principal.

declaración: Regla para declaraciones de variables y funciones.

asignación: Regla para asignaciones de variables.

expresión: Regla general para expresiones matemáticas y lógicas.

condicional: Regla para las estructuras de control condicionales como if y else.

ciclo: Regla para bucles como while y for.

función: Regla para definiciones y llamadas de funciones.

termino: Componente básico de una expresión, como un operando o una constante.

operador: Operadores matemáticos o lógicos, como +, -, *, /, <, >, ==, !=, <=, >=.

identificador: Nombres de variables o funciones definidas por el usuario.

constante: Valores constantes como enteros, flotantes, cadenas, etc.

Símbolos terminales (T):

- Palabras reservadas como "if", "else", "for", "while", "def", etc.

- Operadores como "+", "-", "*", "/", "=", etc.

- ...

Producciones (P):

- programa -> sentencia programa | ε

- sentencia -> declaracion | asignacion | expresion | if | for | while | funcion | llamada_funcion

- ...

Símbolo inicial (S):

- programa

Reglas de la gramática:

- Se utilizan los símbolos no terminales y terminales para definir las producciones.

- Las reglas indican cómo se pueden derivar las cadenas de símbolos terminales.

- Cada producción está representada de la forma A -> α, donde A es un símbolo no terminal y α es una cadena de símbolos terminales y no terminales.

La gramática proporcionada es esencial para comprender la estructura y las reglas de construcción del lenguaje para el analizador léxico-sintáctico.

Algunas de las producciones fueron las siguientes:

programa -> sentencia programa | sentencia

sentencia -> declaracion | asignacion | ciclo_for | ciclo_while | llamada_funcion

declaracion -> "var" identificador

asignacion -> identificador "=" expresion

ciclo_for -> "for" identificador "in" rango "do" bloque "end"

ciclo_while  -> "while" condicion "do" bloque "end"

llamada_funcion -> identificador "(" argumentos ")"

argumentos   -> expresion "," argumentos | expresion

rango        -> expresion ".." expresion

expresion    -> identificador | constante | "(" expresion ")" | operador expresion

operador     -> "+" | "-" | "*" | "/"

identificador -> letra (letra | digito | "_")*

constante    -> digito+

letra        -> "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

digito       -> "0" | "1" | ... | "9"

programa     -> sentencia programa | ε

sentencia       -> declaracion | asignacion | expresion | if_statement | for_loop | while_loop | function_definition

declaracion  -> "var" identificador

asignacion   -> identificador "=" expresion

expresion    -> ...

if_statement -> "if" "(" expresion ")" ":" programa ["else" ":" programa]

for_loop     -> "for" identificador "in" expresion ":" programa

while_loop  -> "while" "(" expresion ")" ":" programa

function_definition -> "def" identificador "(" parametros ")" ":" programa

parametros   -> identificador "," parametros | identificador | ε

expresion -> expresion "+" expresion

| expresion "-" expresion

| expresion "*" expresion

| expresion "/" expresion

| "(" expresion ")"

| identificador

| numero

asignacion -> identificador "=" expresion

| identificador "+=" expresion

| identificador "-=" expresion

| identificador "*=" expresion

| identificador "/=" expresion

| identificador "%=" expresion

| identificador "**=" expresion

| identificador "//=" expresion

arreglo -> "[" expresion ("," expresion)* "]"

diccionario -> "{" parejas_diccionario "}"

parejas_diccionario -> identificador ":" expresion ("," identificador ":" expresion)*

## Requisitos del sistema

El software de análisis léxico-sintáctico en Java tiene los siguientes requisitos mínimos del sistema:

- Sistema operativo: Windows (7/8/10)
- Java Development Kit (JDK) 15 instalado
- NetBeans IDE instalado y configurado correctamente
- Espacio en disco suficiente para la instalación del software y los archivos de datos

## Funcionalidades clave

El software de análisis léxico-sintáctico en Java ofrece las siguientes funcionalidades clave:

- Análisis léxico preciso para identificar tokens y componentes léxicos en la entrada.
- Análisis sintáctico robusto para comprender la estructura gramatical de la entrada.
- Generación de informes detallados sobre posibles errores léxicos o sintácticos en la entrada proporcionada.
- Flexibilidad para personalizar reglas gramaticales según las necesidades específicas del usuario.

## Conclusiones

El analizador léxico-sintáctico basado en Java, desarrollado en un entorno Windows con JDK 15 y NetBeans, es una herramienta eficaz y versátil para el análisis de estructuras de lenguaje natural. Su capacidad para identificar errores y comprender la gramática en la entrada lo convierte en una solución poderosa para una variedad de aplicaciones, desde la programación de compiladores hasta el procesamiento de lenguaje natural.

Este modelo actualizado refleja el entorno de desarrollo específico y proporciona detalles más precisos sobre el software de análisis léxico-sintáctico desarrollado en Java. No dudes en ajustar este modelo según tus necesidades y los detalles específicos de tu propio software.