

MANUAL TECNICO

SOFTWARE DE ANALIZADOR Y  
PROCESADOR DE ER

PRACTICA 1

EXPRESIONES REGULARES

UTILIZACION DE JFLEX Y CUP

## OBJETIVO

Brindar la información necesaria para poder realizar la instalación y configuración del aplicativo

Especificaciones:

- Representar la funcionalidad técnica de la estructura, diseño y definición del aplicativo
- Definir claramente la especificación de los requerimientos de hardware y software necesarios para la instalación de la aplicación.
- Describir las herramientas utilizadas para el diseño y desarrollo del prototipo.

## INTRODUCCION

Este manual describe los pasos necesarios para cualquier persona que tenga ciertas bases de sistemas pueda realizar la instalación del aplicativo creado para la usuarios. Es importante tener en cuenta que en el presente manual se hace mención a las especificaciones mínimas de hardware y software para la correcta instalación del aplicativo.

## REQUERIMIENTOS TÉCNICOS. REQUERIMIENTOS MÍNIMOS DE HARDWARE

1. Procesador : Core
2. Memoria RAM: Mínimo : 1 Gigabytes (GB)
3. Disco Duro : 500Gb.

## REQUERIMIENTOS MÍNIMOS DE SOFTWARE

- Privilegios de administrador
- Sistema Operativo: : Windows XP/Vista/8.1 y Windows 10

# HERRAMIENTAS A UTILIZAR PARA EL DESARROLLO JFLEX

## Introducción

JFlex es un generador de analizador léxico para Java 1 escrito en Java. También es una reescritura de la herramienta JLex (Berk 1996) que fue desarrollada por Elliot Berk en la Universidad de Princeton. Como Vern Paxson afirma para su herramienta C / C ++ flex (Paxson 1995) : no comparten ningún código.

Un generador de analizador léxico toma como entrada una especificación con un conjunto de expresiones regulares y acciones correspondientes. Genera un programa (un *lexer* ) que lee la entrada, compara la entrada con las expresiones regulares en el archivo de especificaciones y ejecuta la acción correspondiente si una expresión regular coincide. Los Lexers suelen ser el primer paso de front-end en compiladores, palabras clave de coincidencia, comentarios, operadores, etc., y generan una secuencia de token de entrada para analizadores. También se pueden utilizar para muchos otros fines.

para instalar jflex en Windows se sigue los pasos siguientes.

1. Descomprima el archivo que descargó en el directorio en el que desea JFlex. Si lo descomprimió para decir C:\, se debe generar la siguiente estructura de directorio:

2. Edite el archivo **bin\jflex.bat**(en el ejemplo es C:\jflex-1.7.0\bin\jflex.bat) de tal manera que 1. **JAVA\_HOME**contiene el directorio donde está instalado su Java JDK (por ejemplo C:\java) y

2. **JFLEX\_HOME**el directorio que contiene JFlex (en el ejemplo: C:\jflex-1.7.0)

3. Incluye el bin\directorio de JFlex en tu camino. (El que contiene el script de inicio, en el ejemplo:) C:\jflex-1.7.0\bin.

Para trabar con jflex consta de tres partes, divididas por %%

- Código usuario
- Opciones y declaraciones
- Reglas lexicas

# CUP

describe la operación básica y el uso del Constructor basado en Java (tm) de analizadores útiles (CUP, por sus siglas en inglés). CUP es un sistema para generar analizadores LALR a partir de especificaciones simples. Cumple la misma función que el programa YACC [1], que se usa ampliamente y, de hecho, ofrece la mayoría de las funciones de YACC. Sin embargo, CUP está escrito en Java, utiliza especificaciones que incluyen el código Java incorporado y produce analizadores que se implementan en Java.

El uso de CUP implica la creación de una especificación simple basada en la gramática para la cual se necesita un analizador, junto con la construcción de un escáner capaz de dividir los caracteres en símbolos significativos (como palabras clave, números y símbolos especiales).

Como ejemplo simple, considere un sistema para evaluar expresiones aritméticas simples sobre enteros. Este sistema leerá expresiones de entrada estándar (cada una terminará con un punto y coma), las evaluará e imprimirá el resultado en la salida estándar. Una gramática para la entrada a dicho sistema podría tener el siguiente aspecto:

```
expr_list :: = expr_list expr_part | expr_part
expr_part :: = expr ';'
expr :: = expr '+' expr | expr '-' expr | expr '*' expr
| expr '/' expr | expr '%' expr | '(' expr ')'
| '-' expr | número
```

Para especificar un analizador basado en esta gramática, nuestro primer paso es identificar y nombrar el conjunto de símbolos terminales que aparecerán en la entrada y el conjunto de símbolos no terminales.

En este caso, los no terminales son:

expr\_list, expr\_part y expr .

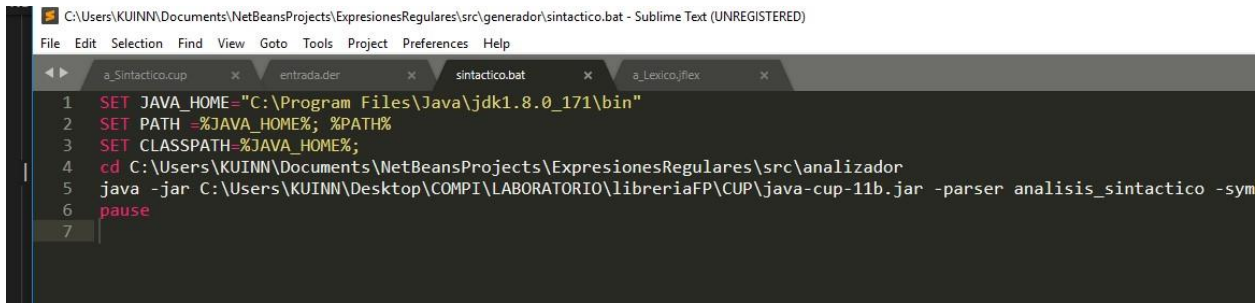
Para nombres de terminales podemos elegir:

SEMI, PLUS, MENOS, TIEMPOS, DIVIDIR, MOD, NÚMERO, LPAREN

Y RPAREN

El usuario experimentado notará un problema con la gramática anterior. Es ambiguo. Una gramática ambigua es una gramática que, dada una cierta entrada, puede reducir las partes de la entrada de dos maneras diferentes, como para dar dos respuestas diferentes. Tome la gramática anterior, por ejemplo. dada la siguiente entrada: 3 + 4 \* 6 La gramática puede evaluar el 3 + 4 y luego multiplicar siete por seis, o puede evaluar 4 \* 6 y luego agregar tres. Las versiones anteriores de CUP forzaban al usuario a escribir gramáticas no ambiguas, pero ahora hay una construcción que permite al usuario especificar las precedencias y asociaciones para los terminales. Esto significa que se puede utilizar la gramática ambigua anterior, después de especificar las precedencias y las asociatividades. Hay más explicación más adelante. Basándonos en estos nombres, podemos construir una pequeña especificación de CUP de la siguiente manera:

## INSTALACION DE CUP



```
C:\Users\KUIINN\Documents\NetBeansProjects\ExpresionesRegulares\src\generador\sintactico.bat - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
a_Sintactico.cup x entrada.der x sintactico.bat x a_Lexico.jflex x
1 SET JAVA_HOME="C:\Program Files\Java\jdk1.8.0_171\bin"
2 SET PATH=%JAVA_HOME%; %PATH%
3 SET CLASSPATH=%JAVA_HOME%;
4 cd C:\Users\KUIINN\Documents\NetBeansProjects\ExpresionesRegulares\src\analizador
5 java -jar C:\Users\KUIINN\Desktop\COMPI\LABORATORIO\libreriaFP\CUP\java-cup-11b.jar -parser analisis_sintactico -sym
6 pause
7
```

## NO TERMINALES EN CUP

```
/* Aqui estan los no terminales */
non terminal String INICIO;
non terminal String CUERPO;
non terminal String CONJUNTO;
non terminal String L_CONJUNTO;
non terminal String VALOR_CONJ;

non terminal ER;
non terminal String BLOQ_ER;
non terminal String L_ER;

non terminal String BLOQ_SEPARAR;
non terminal String SIM_SEPARACION;

non terminal String BLOQ_LEX;
non terminal String LEXEMA;
non terminal String L_LEXEMA;
non terminal PREO;

non terminal String INTERPRETE;
non terminal String MI_INTERPRETE;

/*orden de precedencia*/
precedence right punto;
precedence right barra;
precedence right por;
```

## TERMINALES EN CUP

```
5
6  /* terminales palabras reservadas */
7  terminal String conj;
8  terminal String flecha;
9  terminal String simSeparacion;
10
11
12  /*ER*/
13  terminal String letra;
14  terminal String digito;
15  terminal String entero;
16  terminal String decimal;
17  terminal String id;
18  terminal String cadena;
19  terminal String especiales;
20  terminal String conjunto;
21  terminal String str;
22
23
24  /*---SIMBOLOS---*/
25  terminal String admira;
26  terminal String comDob;
27  terminal String numeral;
28  terminal String dolar;
29  terminal String porsentaje;
30  terminal String ampersand;
31  terminal String comSim;
```

*Funcione para ejecutar jflex y cup en java*

```
public void analizar() throws Exception{
    String input = this.txtEntrada.getText();
    AnalizadorLexico lexico = new AnalizadorLexico(new BufferedReader(new Str
    analisis_sintactico sin = new analisis_sintactico(lexico);
    sin.parse();
    this.mostrarSalida(sin.salida);
    misER();
}
```

```
public void mostrarSalida(String salida){...14 lines }
```