**Laboratorio**

**Analizador Léxico con JAVACC**

JavaCC (Java Compiler Compiler) es una herramienta de generación automática de analizadores gramaticales basada en Java. La herramienta es propiedad de Oracle, la compañía propietaria del lenguaje Java, por lo que se ha convertido en el metacompilador más usado por los programadores en Java.

El funcionamiento de la herramienta consiste en analizar un fichero de entrada, que contiene la descripción de una gramática, y generar un conjunto de ficheros de salida, escritos en Java, que contienen la especificación de un analizador léxico y de un analizador sintáctico para la gramática especificada.

Las características más importantes de esta herramienta son las siguientes:

* Es la herramienta más utilizada en Java. Los propietarios estiman en cientos de miles el número de descargas de la herramienta y los foros de discusión congregan a miles de usuarios interesados en JavaCC.
* Se basa en una análisis sintáctico descendente recursivo.
* Por defecto JavaCC analiza gramáticas de tipo LL(1), pero permite fijar un Lookahead mayor (para analizar gramáticas LL(k)) e incluso utilizar un Lookahead adaptativo.
* Las especificaciones léxica y sintáctica de la gramática a analizar se incluyen en un mismo fichero.
* La especificación léxica se basa en expresiones regulares y la especificación sintáctica utiliza el formato EBNF.
* Junto a la herramienta principal se incluyen dos utilidades: JJTree, para crear automáticamente un generador de árboles sintácticos, y JJDoc, para generar automáticamente la documentación de la gramática en formato HTML.
* La distribución incluye numerosos ejemplos de gramáticas y existen repositorios en internet con la especificación de muchísimas gramáticas en el formato de JavaCC.
* La gestión de errores léxicos y sintácticos está basada en excepciones y contiene información muy valiosa respecto al origen del error y su posición.
* Existe un plugin para Eclipse que facilita la edición y ejecución de la herramienta dentro del desarrollo de cualquier aplicación en Java.

**Instalación del plugin de JavaCC para Eclipse**

La herramienta JavaCC puede integrarse en el entorno de desarrollo Eclipse gracias al siguiente plugin:

http://eclipse-javacc.sourceforge.net/

Existen dos formas de instalar el plugin. La primera consiste en descargar el archivo EclipseJavaCC-1.5.32.zip y descomprimirlo en el directorio de eclipse (por ejemplo, C:\eclipse). La segunda consiste en descargarlo desde Eclipse como una actualización del entorno. En este segundo caso, los pasos a seguir son los siguientes:

1. Desde la ventana principal de Eclipse abrir los menús:

Menu Help / Install New Software

1. En la ventana de búsqueda de actualizaciones, dar clic en el botón Add
2. En esta ventana rellenar los datos:

Name : JavaCC Eclipse Plug-in

URL : <http://eclipse-javacc.sourceforge.net/>

1. Dar clic en OK, llevará a la ventana de Install New Software,
2. Seleccionar " JavaCC Eclipse Plug-in " y pulsar Next
3. Esperar y luego seleccionar SF JavaCC Eclipse Plug-in feature y dar clic en Next
4. Aceptar la licencia y dar clic en Finish.

Usando JAVACC en eclipse.

1. Cree un proyecto en JAVA (Java Project)
2. Una vez que el proyecto esta creado, seleccionar la carpeta src y dar clic derecho sobre ella.
3. Seleccione New->Other.
4. En la ventana que sale escoja JavaCC->JavaCC Template File y de click en Next.
5. Escriba el nombre de la clase por ejemplo Analizador
6. Por defecto el crea un código fuente de prueba. Borrenlo todo y escriban en siguiente código

/\*\*

\* JavaCC template file created by SF JavaCC plugin 1.5.28+ wizard for JavaCC 1.5.0+

\*/

**options**

**{**

**static** = **true**;

**}**

**PARSER\_BEGIN(AnalizadorLexico)**

**public** **class** AnalizadorLexico

{

**static** **int** numero = 0;

**private** **static** **void** grabarLexema (**int** n, String token, String lexema, **int** nL, **int** nC) {

System.out.println(n + "- Token: "+ token + " Lexema:" + lexema + " Linea: "+ nL + "Columna:" + nC);

}

**private** **static** **void** grabarDatosPieza(**int** n, String token, Token pieza) {

grabarLexema(n, token, pieza.image, pieza.beginLine, pieza.beginColumn );

}

**public** **static** **void** main(String args []) **throws** ParseException

{

**try**

{

System.out.println("Ingrese el código a analizar:");

AnalizadorLexico parser = **new** AnalizadorLexico(System.in);

parser.TokenList();

System.out.println("Analisis terminado:");

System.out.println("no se han hallado errores léxicos");

}**catch**(TokenMgrError te)

{

System.out.println("Se han encontrado errores lexicos.");

System.out.println(te.getMessage());

}

**catch**(ParseException e)

{

System.out.println("Analizador: Se han encontrado errores en el analisis.");

System.out.println(e.getMessage());

}

}

}

**PARSER\_END(AnalizadorLexico)**

/\* Espacio en blanco, saltos de linea, tabulares \*/

**SKIP** :

**{**

" "

**|** "\t"

**|** "\n"

**|** "\r"

**|** "\f"

**}**

**TOKEN** : /\* OPERATORS \*/

**{**

**<** SUMA : "+" **>**

**|** **<** MULTIPLICACION : "\*" **>**

**|** **<** ASIGNACION : "=" **>**

**}**

**TOKEN** : /\*PALABRAS RESERVADAS\*/

**{**

**<** ENTERO: "entero" **>**

**|** **<** CADENA: "cadena" **>**

**}**

**TOKEN** : /\*NUMEROS EJEMPLO 1, 1234, 987\*/

**{**

**<** NUMERO : **(**< DIGIT >**)+** **>**

**|** **<** #DIGIT : **[** "0"**-**"9" **]** **>**

**}**

**TOKEN** : /\*IDENTIFICADORES EJEMPLO a, casa, clase1\*/

**{**

**<** IDENTIFICADOR : **[**"a"**-**"z","A"**-**"Z","\_"**]** **(** **[**"a"**-**"z","A"**-**"Z","0"**-**"9","\_"**]** **)\*** **>**

**}**

/\*\*

\* Reconoce el contenido completo de un archivo

\*

\* TokenList -> (TokenElement)\*

\*/

**void** TokenList():

**{}**

**{**

**(** TokenElement() **)\***

**}**

**void** TokenElement():

**{**Token pieza; **}**

**{**

pieza = <SUMA> **{** numero++; grabarDatosPieza(numero, "SUMA" , pieza); **}**

**|** pieza = <MULTIPLICACION> **{** numero++; grabarDatosPieza(numero, "MULTIPLICACION", pieza); **}**

**|** pieza = <ASIGNACION> **{** numero++; grabarDatosPieza(numero, "ASIGNACION", pieza); **}**

**|** pieza = <IDENTIFICADOR> **{** numero++; grabarDatosPieza(numero, "IDENTIFICADOR", pieza); **}**

**|** pieza = <NUMERO> **{** numero++; grabarDatosPieza(numero, "NUMERO", pieza); **}**

**}**

1. Para compilar, clic derecho sobre el archivo .jj y darle compilar con JAVACC. Esto generará varias clases .java.
2. Para probarlo vayan a la clase Analizador.java y dar clic derecho Run as -> Java Aplication
3. Adicionen nuevos tokens.
   1. Palabras reservadas: Si, Sino, Mientras
   2. Simbolos: {, }, ;
   3. Operadores como -, y /
   4. Numero con decimales
4. Modifiquen el token Identificador para que inicie con el símbolo $.
5. Creen un nuevo proyecto e inicien su propio Analizador Léxico

Entregables:

1. Deben enviar al moodle, el archivo con las adiciones y modificaciones propuestas en el laboratorio
2. Enviar al moodle un documento, que muestre lo que avanzaron de su propio analizador léxico.