Práctica 1: Construcción de un analizador léxico para *alike*

Procesadores de lenguajes

Dpto. de Informática e Ingeniería de Sistemas,
Grado de Ingeniería Informática
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

1. Objetivos

Los objetivos para esta práctica son:

- Conocer las características del lenguaje de programación ***alike***, que manejaremos a lo largo de las sesiones de laboratorio
- Aprender las características y funcionamiento del metacompilador JavaCC
- Implementar un analizador léxico para ***alike***, incluyendo alguna técnica de recuperación de errores léxicos

2. Introducción

El documento titulado "*El lenguaje **alike***" contiene la descripción de las principales características del lenguaje (que puedes completar a partir de la lectura de los distintos ejemplos contenidos en la batería de test suministrada). En esta práctica se pide implementar un analizador léxico para el lenguaje.

Con el fin de comprobar que, efectivamente, reconoce los tokens del lenguaje, se va a pedir que muestre información relativa al token, como se describe a continuación.

3. Tareas a realizar

Obligatoriamente, hay que realizar las siguientes tareas:

1. Construir un analizador léxico para ***alike***. El fichero fuente a analizar se suministra como un parámetro en la invocación desde la línea de comandos. Para cada token reconocido, debe mostrar por la salida estándar la información correspondiente, de la siguiente manera, donde el par entre paréntesis corresponde a la línea y columna donde se ha localizado el lexema correspondiente al token:

```
(25,73): identificador "edad"
(170,333): operador asignación ":="
(4,90): constante entera "899"
(1004,30): token "while"
...
```

2. En el caso de detectarse un error léxico, deberá mostrar por la salida estándar un mensaje del tipo:

```
ERROR LÉXICO: (<línea, columna>): símbolo no reconocido: <símbolo>
```

Algunos comentarios:

- Recordad que en esta práctica sólo hay que hacer un análisis léxico, no importa si la estructura del programa es correcta o no. De eso se encargarán fases posteriores del análisis.
- De los programas de prueba que se os proporcionan, todos son correctos desde el punto de vista léxico.
- Puede ser de ayuda consultar información sobre `javacc`, la opción `COMMON_TOKEN_ACTION` y la declaración del procedimiento `CommonTokenAction`.

4. Entrega de resultados de la práctica

4.1. Lo que hay que entregar

Como resultado de la práctica, todos los alumnos deberán entregar el fichero `practica_1.zip`. Este, una vez descomprimido, tendrá la estructura que se muestra en la parte izquierda de la figura 1. Los nombres de los ficheros y directorios mostrados deben respetarse. La estructura corresponde a lo siguiente:

- El fichero `alike.jj` contiene el fuente del analizador léxico pedido
- El fichero `README.txt` contiene información sobre los autores: nombres, NIPs, etc.
- El directorio `lib` de momento está vacío, pero en un futuro contendrá librerías que desarrollemos y se necesiten para la compilación del procesador. Por ejemplo, librerías para el manejo de la tabla de símbolos, de procesamiento semántico, etc.

- El fichero `build.xml` contiene la descripción del proyecto. Este fichero es el que se suministra en la web de la asignatura (o una adaptación del mismo cuando sea necesario). El funcionamiento de "ant" es análogo al de un "makefile". En caso de invocaciones sucesivas, solo compilará los ficheros que hayan sufrido cambios desde la última vez que se invocó, y los que dependan de ellos (buscad información al respecto). Para adaptarlo a vuestro entorno, basta con modificar la línea siguiente con el path hasta donde hayáis instalado javacc en vuestra máquina.

```
javacchome="${user.home}/aplicaciones/javacc/target"
```

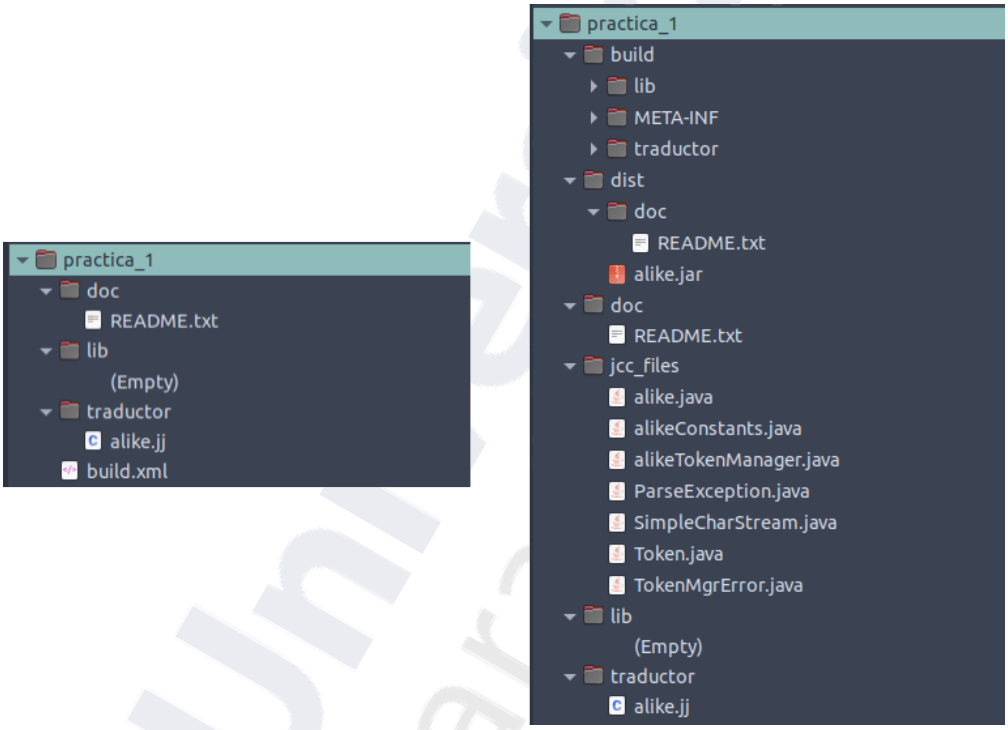

Figura 1: Estructura del proyecto antes y después de su compilación

La ejecución del comando `ant` llevará a cabo la compilación del proyecto. Si todo ha ido correctamente, tras su ejecución, el contenido del directorio será el mostrado en la parte derecha de la figura 1.

En el nuevo estado:

- Se ha creado el directorio `jcc_files` con los fuentes java generados por `javacc`, que implementan el analizador léxico especificado en `alike.jj`
- Se ha creado el directorio `build`, que contiene las clases compiladas, así como otra información especificada en el proyecto.

- Se ha creado el directorio `dist` con los materiales para la distribución de la aplicación. Contiene, por un lado, el directorio `doc`, que es una copia del original. Por otra parte, la aplicación generada, `alike.jar`. Por ejemplo, para analizar el fuente `mcd.al`, la instrucción a ejecutar será

```
java -jar alike.jar mcd.al
```

Por último, la ejecución del comando `ant clean` devolverá el directorio a su estado original, eliminando todo lo que hubiera generado (incluyendo el directorio de distribución, claro).

4.2. Método de entrega

La entrega se debe hacer en *hendrix* mediante la ejecución del programa

```
someter procleng_23 practica_1.zip
```

4.3. Plazos de entrega

El plazo de entrega es el establecido en moodle, y dependerá del grupo de prácticas.