

Ejercicios sesión 1 de laboratorio PLG

Albert Rubio

7 de febrero de 2020

Ejercicio 1: *Primer ejemplo con JLex.* En el Campus Virtual de la asignatura en el apartado de “Documentación Laboratorio Análisis Léxico” encontraréis un enlace para obtener *JLex* como archivo “.jar” y otro para obtener el código Java para generar el analizador léxico en “alex.zip”. También encontraréis un ejemplo de especificación del léxico usando JLex “ejemplo.l” y un ejemplo de entrada para el procesamiento final en “input.txt”. Descargad todos estos archivos en una carpeta y descomprimid “alex.zip”.

Para más información o versiones más recientes de JLex visitad <https://www.cs.princeton.edu/~appel/modern/java/JLex/>.

Podéis instalar JLex en vuestra cuenta usando el archivo “.jar” o lo podéis usar directamente en la línea de comandos. También podéis utilizar el entorno para desarrollo de código Java que prefiráis. Deberéis definir los caminos necesarios en el CLASSPATH o usar “-cp” al llamar a java.

Para procesar el archivo `ejemplo.l` con JLex, tenéis que ejecutar:

```
$ java -cp jlex.jar JLex.Main ejemplo.l
```

Esto generará un archivo `ejemplo.l.java` cuyo contenido debería coincidir con el archivo `alex/AnalizadorLexicoTiny.java` (si no es el caso, borrad este último y reemplazadlo por `ejemplo.l.java` renombrado). Compilad el código java de la carpeta “alex”:

```
$ javac alex/*.java
```

y ya podéis usar el analizador. En la versión del `Main` que se adjunta, se lee de la entrada de un fichero que se pasa por parámetro. El fichero “input.txt” contiene la siguiente entrada

```
evalua
166.386 * euros + 1.66386 * (centimos1 + centimos2)
donde
euros = 567,
centimos1 = 456,
centimos2 = 10
```

Ejecutad

```
$ java alex.Main input.txt
```

desde el directorio el directorio de trabajo y obtendréis los tokens de la entrada (puede ser necesario añadir “-cp .” si no tenéis “.” en el CLASSPATH).

Ejercicio 2: *Variaciones.* Realizad pequeños cambios en la especificación JLex y el código asociado. Por ejemplo, podemos ampliar la representación de números para admitir números en binario y hexadecimal con la notación 0b01101101 para binarios y 0x1faCd5 para hexadecimales.

Ejercicio 3: *Patrones.* Suponemos que nos pueden entrar palabras (formadas solo por letras mayúsculas y minúsculas), patrones (formadas por letras mayúsculas y minúsculas y un carácter ‘*’) y cualquier otra cadena de caracteres (sin separadores). Dada una entrada, debemos extraer la palabras, los patrones y el resto de cadenas como unidades léxicas distintas (PALABRA, PATRON y OTRO) y para cada patrón indicar el número de palabras que le encajan: palabras que coinciden con el patrón reemplazando el ‘*’ por una cadena de caracteres. Así, con la entrada

```
hola
ho*a
laz+dfg
ho*
ho*+sd
pista
+fdsghf
hoja
a*a
hoyo
ala
a
```

la salida debería ser

```
[clase:PALABRA,fil:1,col:0,lexema:hola]
[clase:PATRON,fil:2,col:0,lexema:ho*a]
[clase:OTRO,fil:5,col:0,lexema:laz+dfg]
[clase:PATRON,fil:3,col:0,lexema:ho*]
[clase:OTRO,fil:6,col:0,lexema:ho*+sd]
[clase:PALABRA,fil:7,col:0,lexema:pista]
[clase:OTRO,fil:8,col:0,lexema:+fdsghf]
[clase:PALABRA,fil:9,col:0,lexema:hoja]
[clase:PATRON,fil:4,col:0,lexema:a*a]
[clase:PALABRA,fil:10,col:0,lexema:hoyo]
[clase:PALABRA,fil:11,col:0,lexema:ala]
[clase:PALABRA,fil:12,col:0,lexema:a]
[clase:EOF,fil:13,col:0]
PALABRAS:
```

```
hola
pista
hoja
hoyo
ala
a
PATRONES:
ho*a: 2
ho*: 3
a*a: 1
```

Debéis crear una nueva definición léxica con JLex y modificar convenientemente el código java en ClaseLexica.java y ALexOperations.java. Finalmente tenéis que añadir código al Main.java para que haga el procesamiento final de las unidades léxicas obtenidas.