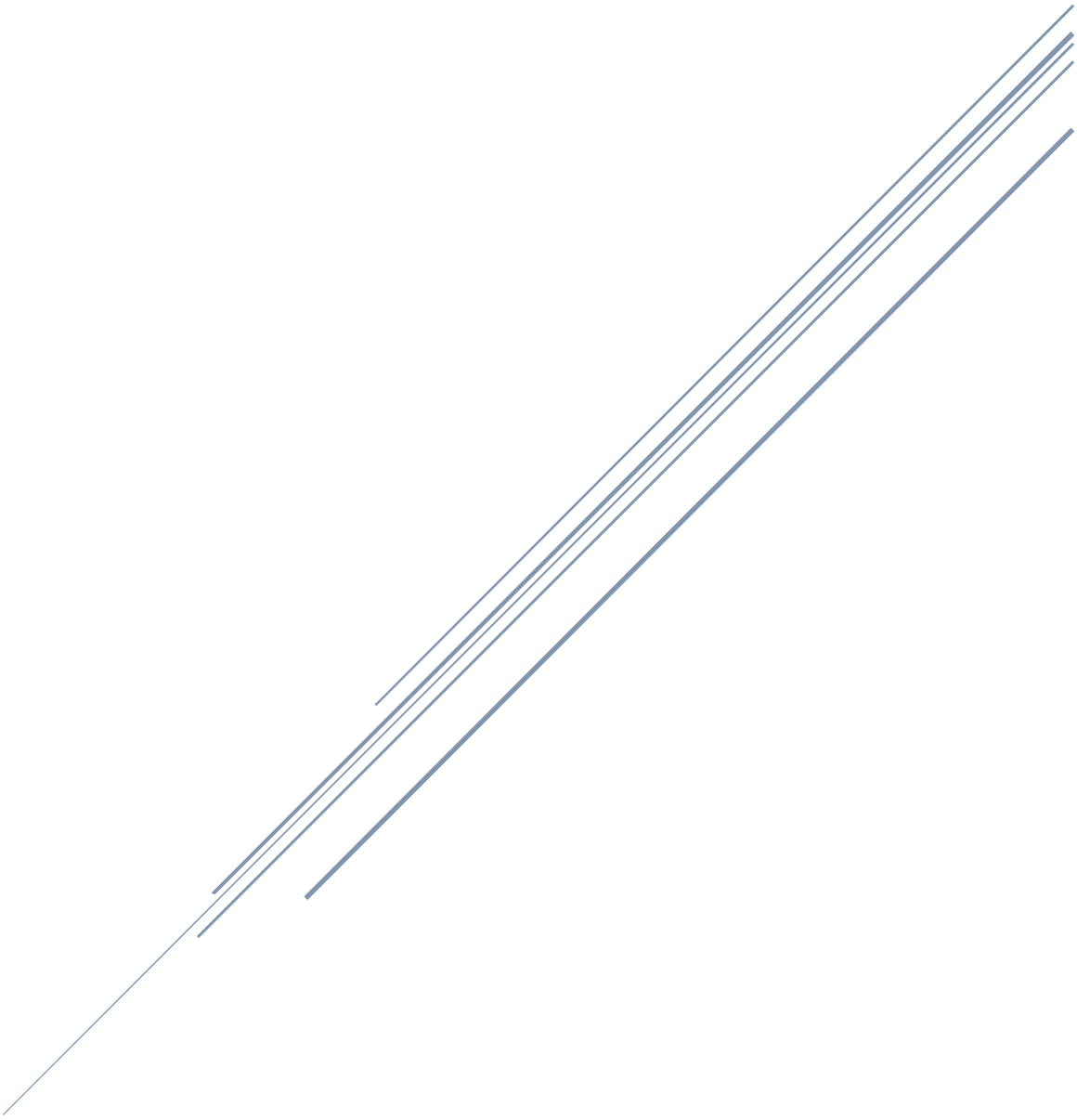
PROCESADORES DEL LENGUAJE

PRACTICA 1.1

Universidad Carlos III Madrid, Ingeniería Informática



IVAN AGUADO PERULERO – 100405871
JORGE SERRANO PÉREZ – 100405987

Contenido

1.- INTRODUCCIÓN

En el presente documento se van a describir brevemente las decisiones de diseño tomadas frente a los problemas abordados en esta primera práctica.

El objetivo de ella es desarrollar un proyecto desacoplado del analizador sintáctico, es decir, crear un analizador léxico independiente haciendo uso de *JFLEX*.

2.- PASOS SEGUIDOS

En primer lugar, hemos eliminado todo lo relacionado con el proyecto cup. A continuación, hemos dado una estructura jerárquica inicial y hemos añadido todos los ficheros expuestos en el enunciado para poder generar, compilar y ejecutar el código sin ningún problema (clase *Symbol.java*, interfaz *SimbolosTerminales.java*, clase *Driver.java*, *lexer.jflex*).

Tras realizar esto, hemos pasado a modificar la interfaz *SimbolosTeminales.java* añadiendo los nuevos símbolos que deben poder ser leídos según el enunciado de la práctica (*numeroReal, numeroHex, DNI,* …).

A continuación, en el fichero *lexer.jflex* se han añadido las expresiones regulares que permiten identificar los símbolos necesarios. Algún ejemplo sería:

- *RealNumber = [0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?*
- *HexNumber = "0X"[0-9A-F]+|"0x"[0-9A-F]+*
- *Email = [A-Za-z0-9]+@[A-Za-z0-9.]+\.[A-Za-z]{2}[A-Za-z]**
- *Date = (3[01]|[12][0-9]|0[1-9])\/(1[0-2]|0[1-9])\/[0-9]{4}*

Tras añadir todas las expresiones que íbamos a necesitar, hemos definido cada una de las acciones asociadas que se ejecutarán cuando se detecte alguna de ellas. Como en esta práctica simplemente se pide identificar los tokens, todas las acciones devuelven el símbolo detectado, excepto las que requieran ser ignoradas, como los comentarios, que no tendrán una acción asociada.

En la parte que nos permite ejecutar la aplicación, es decir, en el *main*, hemos realizado una serie de cambios para facilitar la resolución del problema. Por un lado, hemos modificado el código para que siempre lea los datos introducidos mediante un fichero. Para ello, en *Project/Properties*, dentro del apartado *Run/Debug Settings*, hemos modificado los argumentos pasando el fichero *input.txt*, donde se encuentran los distintos tokens que se quieren leer.

Por otro lado, hemos utilizado distintos bucles que recorren el fichero. Uno hace uso de un buffer para contar el número de líneas y el número de caracteres, y el otro lee los distintos tokens mediante la función *next_token*.

Queremos destacar también que hemos optado por crear distintos *arrayList* en los que almacenar cada categoría especificado en el enunciado, ya que esto nos ha facilitado

enormemente el mostrar por pantalla los resultados de manera correcta. Dependiendo del tipo de token introducido, lo almacenamos en el *arrayList* correspondiente.

Por último, para comprobar que nuestro código es válido y devuelve la salida esperada se ha probado con el fichero de prueba *input.txt,* el cual tiene un ejemplo de cada token aceptado. Este se encuentra en la carpeta entregada.

3.- CONSIDERACIONES PARA EL CORRECTOR

Para que el programa funcione correctamente se deben tener en cuenta las siguientes consideraciones:

1. Como se ha mencionado en el apartado anterior se debe pasar como argumento de entrada el fichero *input.txt* o cualquier otro que se quiera probar.
2. Los nombres y apellidos deben comenzar por letra mayúscula.
3. Solo se aceptan direcciones de email que comienzan por letras o números, sigan por @, luego letras, números o puntos, para seguir con un punto obligatorio y al menos dos letras al final.
4. La letra de los DNI debe ser mayúscula.
5. Las matrículas aceptadas están formadas por 4 números seguidas de 3 letras mayúsculas.
6. Solo se aceptan fechas en el formato dd/mm/aaaa.
7. Se ha optado por clasificar los símbolos + y - como operadores, en vez de asociarlos a un número.
8. No se aceptan tildes.

4.- CONCLUSIONES Y PROBLEMAS ABORDADOS

En primer lugar, queremos destacar lo beneficiosa que ha sido la práctica 0 para la correcta resolución de esta práctica. Nos ha ayudado a comprender y ver claramente el funcionamiento de la herramienta *JFLEX*.

Por otro lado, el manual de *JFLEX* también nos fue de gran utilidad y ayuda, ya que en él se proporcionan ejemplos muy representativos.

En cuanto a los problemas que nos hemos encontrado, nos costó al principio volver a utilizar Eclipse, ya que este *IDE* no nos avisaba correctamente de los errores al compilar.

En conclusión, se trata de una práctica muy útil que nos ha ayudado a fijar y poner en práctica los conocimientos teóricos adquiridos.