# Grado en Ingeniería Informática

# Procesadores de lenguajes – Análisis léxico

# JFlex – Hoja de problemas

## *Objetivos de la práctica*

- Conocer el funcionamiento de los generadores de analizadores léxicos.
- Saber construir traductores léxicos utilizando la herramienta JFlex (http://jflex.de/).

## *Actividades a realizar*

1. Instalación JFlex
2. Uso de JFlex
3. Probar ejemplo en JFlex
4. Ejercicios de traductores léxicos

### 1. Instalación de JFlex

Para instalar JFlex hay que completar los siguientes pasos:

1. Descargar la heramienta: http://www.jflex.de/download.html
2. Descomprimir en un directorio el fichero descargado en un directorio conocido y accesible.
3. JFlex se puede ejecutar de forma **interactiva** ejecutando directamente el fichero **`JFlex.jar`** descargado.
4. También se puede ejecutar como un comando desde una **consola de comandos**, ver manual para esta opción: http://jflex.de/manual.html.

### 2. Uso de JFlex

Para utilizar JFlex hay que dar los siguientes pasos:

1. Editar en un fichero de texto la especificación léxica. Resultado: **fichero .flex**
2. Pedir a JFlex que genere el traductor. Resultado: **fichero .java**
3. Compilar el programa generado por JFlex. Resultado: **fichero .class**
4. Ejecutar el programa proporcionándole un fichero de entrada apropiado para la ejecución.

## 3. Ejemplo de traductor

El siguiente traductor genera una lista de token reconocidos de la forma < token >.
Donde token puede ser:

1. Numero entero: ristra de dígitos decimales.
2. Número real: dos ristras de dígitos decimales separadas por un punto “.”.

El código de la especificación JFlex es la siguiente:

```
%%

%standalone

%%

[0-9]+”.”[0-9]+     {System.out.println("<Real>");}
[0-9]+              {System.out.println("<Natural>");}
```

1. Se guarda este código en un fichero de texto llamado **`prueba0.flex`**.
2. Se ejecuta JFlex en modo interactivo (**`JFlex1.6.1.jar`** se encuentra en el directorio **`lib`** del fichero comprimirdo descargado)
3. Como resultado se habrá creado **`Yylex.java`**
4. Compilar el fichero **`Yylex.java`**. Si todo está bien se creará un fichero fichero **`.class`**
5. Crear un fichero de texto de prueba **`pr0-0.txt`** para ejecutar con números naturales y reales en coma fija.
6. Ejecutar el fichero **`.class`** ya compilado: **`java Yylex pr0-0.txt`**
7. Comprobar que la salida es la correcta.

Es recomendable leer el manual de la herramienta: http://jflex.de/manual.html

## 3. Ejercicios de traductores léxicos

**Atención: crea un fichero de texto y ve guardando las soluciones, cuando termine la clase envíalas por correo al profesor a través del campus virtual.**

Diseñar una especificación JFlex que:

1. Dado un fichero de texto de entrada, cambie todas las vocales minúsculas por vocales mayúsculas.

2. Reciba un fichero de texto y una palabra y cuente el número veces que aparece dicha palabra en el fichero.

3. Copie el archivo de entrada en uno de salida, sumando 5 a todo número positivo que sea múltiplo de 3.

4. Dado un fichero de texto de entrada con letras mayúsculas y minúsculas, signos de puntuación y saltos de línea, codifique cada palabra utilizando el siguiente código:
    a. Si la palabra termina en vocal, se cambian todas sus letras por la letra anterior en el alfabeto.
    b. Si la palabra termina en consonante, se cambian todas sus letras por la letra posterior en el alfabeto.
    c. El resto de símbolos se dejan igual.
    d. NOTA: después de la "z" se entiende que va la "a". Se recomienda usar la clase Carácter de Java.

5. Dado un fichero de texto de entrada obtenga las siguientes estadísticas:
    a. Número total de caracteres utilizados (excepto espacios en blanco), palabras (ristras de letras), frases (terminadas con un ".") y párrafos (terminados con una línea en blanco).
    b. Número de medio de: caracteres por palabra, palabras por frase y frases por párrafo.

6. Dado un fichero ofrezca como salida un fichero en el que se han eliminado de la entrada las líneas en blanco, los espacios en blanco al principio o al final de línea y que tan sólo deja un blanco entre cada dos palabras consecutivas.

7. Hacer un programa que interprete rangos de números de página. Para la entrada: 1-4;5;8;45-47;60 debería generar: 1,2,3,4,5,8,45,46,47,60

8. Dado un fichero de texto de entrada lo procese de la siguiente forma:
    a. Si al comienzo de la línea se encuentra una "m", cambiará todas las letras mayúsculas por minúsculas.
    b. Si al comienzo de la línea se encuentra una "M", cambiará todas las letras minúsculas por mayúsculas.
    c. En otro caso deja la línea como está.

9. Dado un fichero de texto de entrada, genere el mismo fichero con las líneas numeradas:

```
Primera línea
Segunda línea

Cuarta línea, la 3ª está vacía
```

```
1:Primera línea
2:Segunda línea
3:
4:Cuarta línea, la 3ª está vacía
```

10. Dado un fichero donde hay constantes numéricas en las siguientes bases:
    - *Binaria*: "b" seguido de una ristra de dígitos binarios.
    - *Octal*: "x" seguido de una ristra de dígitos octales.
    - *Decimal*: ristra de dígitos decimales) y
    - *Hexadecimal*: "0x" seguido de una ristra de dígitos hexadecimales).

Donde las constantes numéricas pueden ser enteras o reales con signo opcional. Las constantes reales se pueden expresar en formato de punto fijo o coma flotante. A continuación se muestran algunos ejemplos.

| | **binaria** | **octal** | **decimal** | **hexadecimal** |
|---|---|---|---|---|
| **Entero** | -b011 | +x56 | 098 | -0x34A5 |
| **Real punto fijo** | b01.11 | x54.01 | -83.91 | +0xff.A4 |
| **Real coma flotante** | +b.001e-111 | x.74e230 | +.54e-4 | 0x.4Ae+FF |

NOTAS:
- Como funciones de conversión sólo se permite usar las de las clases Character e Integer.
- Los dígitos hexadecimales no numéricos son siempre letras mayúsculas.
- El exponente de las constantes reales en coma flotante se empieza siempre con "e".

Se pide devolver los tokens correspondientes: <token, valore numérico en base decimal>

11. Realice el ejercicio anterior suponiendo que sólo nos interesan aquellas constantes numéricas que se encuentran dentro de comentarios, de línea (// …) o multilínea (/* … */)