Tabla de Contenidos

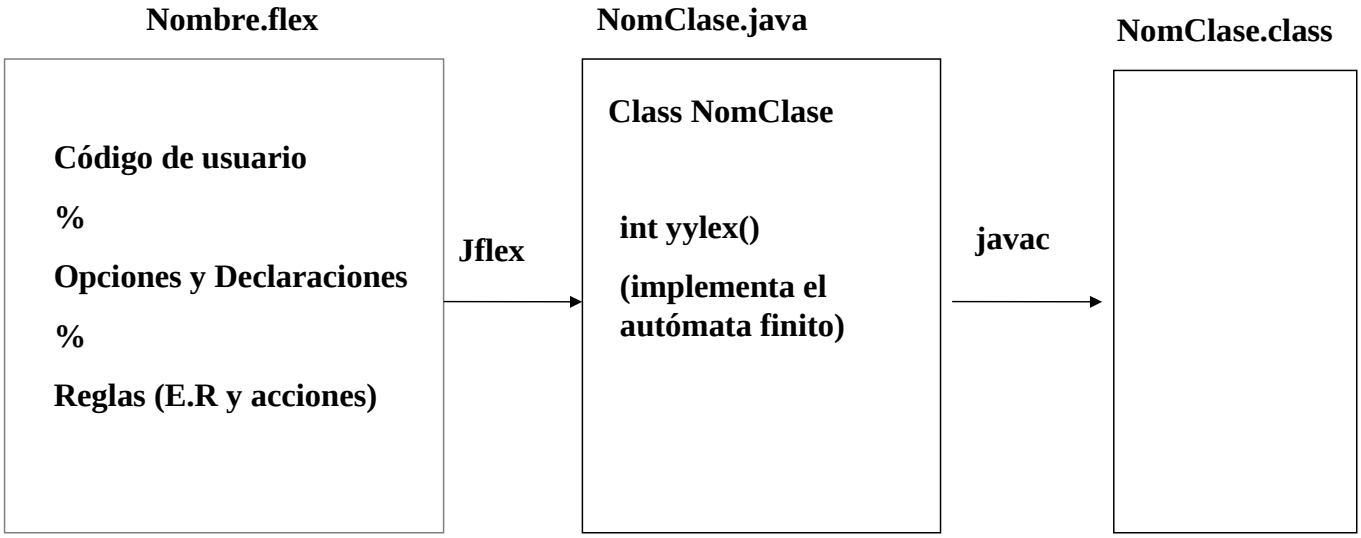
Jflex)

Funcionamiento general de JFlex

- **Jflex** es un generador de analizadores léxicos, escrito en java y que genera código java.
- En líneas generales, Jflex toma como entrada un fichero (nombre.flex) donde se han definido una secuencia de expresiones regulares y genera un fichero que implementa una clase java (NomClase.java), que contiene, entre otros métodos, al método yylex(), el cual implementa a un autómata finito determinista que reconoce si una palabra de entrada pertenece a alguno de los lenguajes definidos por las expresiones regulares.
- La ejecución del método yylex() produce la lectura de caracteres de entrada y la ejecución del autómata hasta llegar a un estado final.

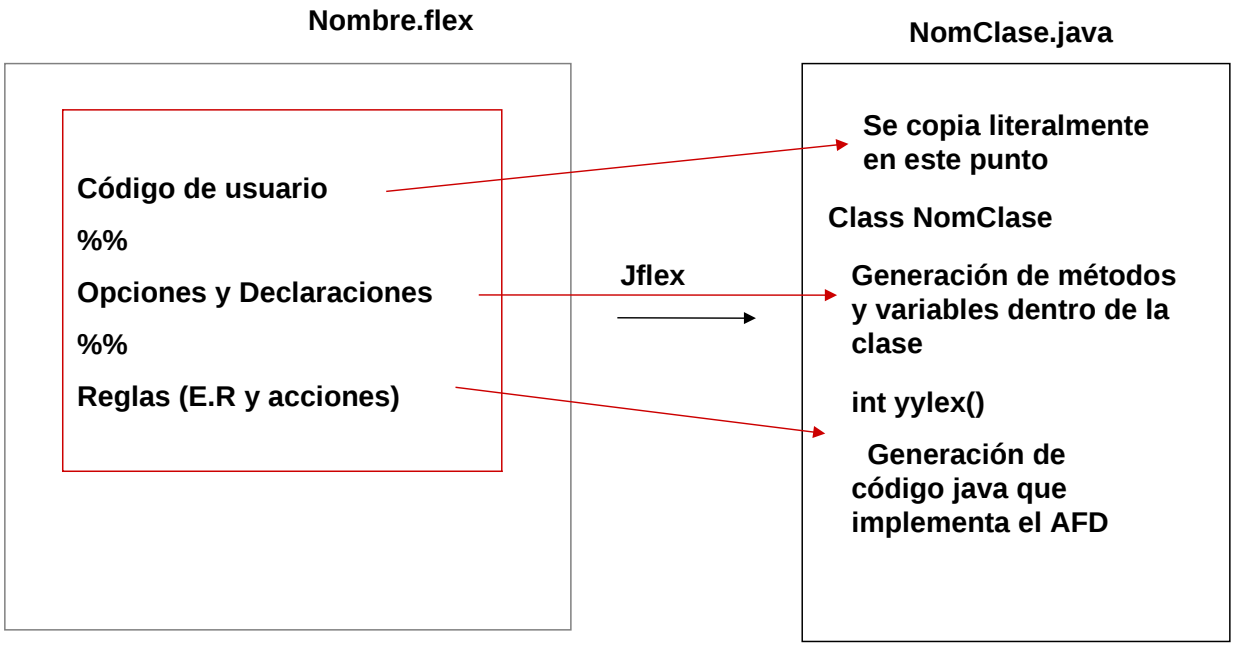
Jflex

Funcionamiento de Jflex



Jflex

Funcionamiento de Jflex



❑ La estructura de un archivo *.flex consta de tres secciones que permiten "rellenar" el esqueleto de la clase de java generada, cuyo método principal (yylex()) implementa un Autómata Finito Determinista a partir de una secuencia de Expresiones Regulares.

Sección 1: Código de Usuario

❑ En esta sección se incluye código java que será copiado literalmente en el fichero .java, delante de la declaración de la clase (Class NombreClase).

❑ Las declaraciones usuales son sentencias del tipo *import o package*

Sección 2: Opciones y Declaraciones

❑ Esta sección sirve para definir características especificas de la clase que se genera.

❑ Las opciones se define con la directiva **%** seguida de la opción.

❑ Algunas opciones posibles son las siguientes:

- ❑ **%class** <NombreClase> permite decidir cual es el nombre de la clase que se va a generar.
- ❑**%unicode** decide que el conjunto de caracteres que va a ser usado por el analizador sea de 16 bits. Otras opciones son:
  - ❑**%16bit** análoga a %unicode
  - ❑**%8bit o %full**. Define al conjunto de caracteres con 8 bits
  - ❑**%7bit** Define al conjunto de caracteres con 7 bits.

Sección 2: Opciones y Declaraciones

- **%line**. Provoca que se implemente un contador de lineas que puede ser consultado mediante la variable yyline. %colum y %char provoca la implementación análoga respecto al número de columnas y el número de caracteres, usando las variables yycolum e yychar respectivamente.
- **%standalone**. Crea una función main en la clase generada que recorre un fichero de entrada, ejecutando el método yylex() mientras haya caracteres en el fichero.
- **%table,%swich,%pack**, sirven para elegir la forma en que se genera el código del automata.

Declaraciones

- Se pueden definir métodos y variables para que sean insertados en la definición de la clase generada con las directivas:

```
%{   Métodos y variables (código java)
 %}
```

- El código que escribamos entre **%{ y %}** será copiado literalmente dentro de la clase.
- **%init{**… Código Java **%init}**. El código escrito en esta declaración será copiado literalmente dentro del constructor de la clase.
- **%eof{**….. Código java **%eof}**. El código incluido en esta declaración será ejecutado una sola vez cuando se termina de leer el fichero de entrada. Este código se copia dentro del método yy_do_eof(), el cual es llamado al encontrar el fin de fichero.

Opciones y Declaraciones

- En la sección de declaraciones se pueden definir macros, es decir, podremos dar un nombre a una E.R. para usarla en la definición de E.Rs con ese nombre.

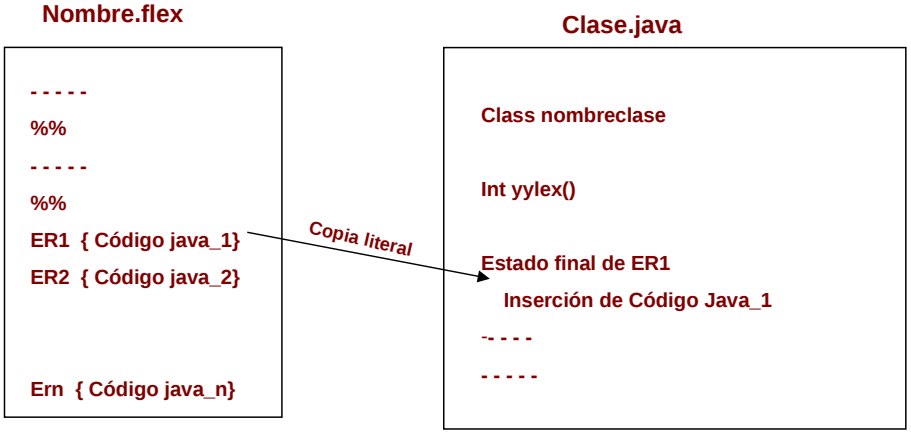
Sintáxis: NombreExp = ER

- El uso de la macro en una ER deberá cotener al nombre de la macro entre llaves: {NombreExp}

Sección 3: Reglas y Acciones

- Definición de una secuencia de Expresiones Regulares

- Cada Expresión Regular tiene asociado una secuencia de acciones (instrucciones en código java) que son ejecutadas cada vez que se reconoce una palabra del lenguaje asociado a la E.R



Sección 3: Reglas y Acciones

- Si el analizador encuentra una palabra que pertenece a mas de un lenguaje (definido por mas de una E.R.) se escoge la primera ER que incluye al lenguaje (ejecutando sus acciones asociadas).

- El método yylex va a reconocer a la cadena mas larga que pertenece a un lenguaje.

Sintaxisis para la definición de Expresiones Regulares (Reglas)

- Cualquier símbolo (carácter ascii) usado en la E.R. es un símbolo del alfabeto. Ejemplo: a b 7¡ etc. y por tanto una E.R.
- Los caracteres: **| ( ) { } [ ] < > \ . * + ? ^ $ / . “ ~ !** son caracteres especiales usados por jflex para definir la especificación. Si queremos usarlos como símbolo del alfabeto antepondremos la barra \ delante del carácter. Ejem \{.
- Los operadores básicos de una E.R son:
  - **Unión** cuyo operador es **|** . Ejem, a | b
  - **Concatenación.** Se produce definiendo dos E.R juntas. Ejem. ab
  - **Clausura** * y clausura positiva **+.** Ejem. a*****

Sintaxisis para la definición de Expresiones Regulares (Reglas)

- **Cadena de caracteres**: **“**hola**”** . Es la concatenación de los caracteres ASCII que están dentro de las comillas.

- **Conjunto de Caracteres**. Cualquier carácter escrito entre un corchete abierto y otro cerrado. Ejem. **[abc]** equivale a (a|b|c).
  - **Rango.** Define una rango de caracteres. Ejem: [a-zA-Z] define a cualquier carácter Ascii entre la a y la z y la A y Z (mayúsculas).
  - **Complemento.** Define cualquier carácter que no está en el conjunto. Ejem: [**^**abc].

- Otros operadores:
  - **?.** Cero o una vez. Ejem. a**?** es equivalente a (a |λ).
  - **a{n}** . Define la concatenación de a n veces. Ejem. a{3} es aaa
  - **a{n,m}.** Define la concatenación de a un número de veces entre n y m.

- ~a. define a una cadena en la entrada que finalize con la cadena a.
- El carácter punto **.** denota a la E.R que representa cualquier carácter (excepto el retorno de linea \n). La definición de esta expresión al final de la secuencia de E.Rs hace que cualquier carácter que no pueda formar parte de algunos de los lenguajes, sea reconocido por esta E.R

Algunos métodos útiles

- El método String **yytext()** es generado dentro de la clase. Devuelve la cadena que acaba de ser reconocida.

Instalación de JFLEX

- Disponible para Windows y Linux a través de la página http://www.jflex.de o bien descargando el fichero comprimido de MOODLE.
- Pasos en la instalación para Windows:
  - Descomprimir el fichero con extensión ZIP.Por ejemplo en C:\
  - Edita el fichero JFlex.bat (en el ejemplo la ruta es C:\JFlex-1.4.1\bin\JFlex.bat) y especifica el valor de las variables siguientes:
    - JAVA_HOME indica el directorio donde se encuentra el Java JDK instalado
    - JFLEX_HOME indica el directorio donde se encuentra el JFlex
  - Incluye el directorio \bin de JFlex en tu ruta. (Mi PC->Propiedades->Opciones avanzadas -> Variables de entorno-> Variables de sistema: Editar la variable Path y añadir al final la cadena ;C:\JFlex-1.4.1\bin

Ejemplo del fichero jflex.bat

```
@echo off

set JFLEX_HOME=C:\JFLEX-1.4.1

set JAVA_HOME="C:\Archivos de programa\Java\jdk1.5.0"

set CLPATH=%JAVA_HOME%\lib\classes.zip;%JFLEX_HOME%\lib\JFlex.jar

REM for JDK 1.2
java -Xmx128m -jar %JFLEX_HOME%\lib\JFlex.jar %1 %2 %3 %4 %5 %6 %7 %8 %9
```


Instalación de JFLEX

- Pasos en la instalación en Linux:
  - Normalmente está disponible en la distribución así que podemos usar el comando de instalación en modo administrador. Por ejemplo en Ubuntu (también instalamos cup):
    - sudo apt-get install jflex cup .
    - Ejecuta en un terminal:

      echo"export CLASSPATH=$CLASSPATH:/usr/share/java/cup.jar:/usr/share/java/JFlex.jar" | tee -a ~/.bashrc
  - También podéis coger el fichero rpm de la página web de Jflex (http://www.jflex.de) y , en modo administrador ejecutar:
    - rpm -U jflex-1.4.3-0.rpm

Uso de JFLEX

- Abrir una ventana del sistema y ejecutar los siguientes comandos.
  - Jflex <opciones> <ficheros .flex>. => Ejecuta Jflex sobre los ficheros de entrada.
  - Javac clase.java => Compila la clase que generó Jflex.
  - Java clase fichero_entrada => Ejecuta clase sobre un fichero con las cadenas que queremos reconocer.
- Ejemplo:
  - Jflex numeros.fle => Genera el fichero numeros.java
  - javac numeros.java => Genera la clase numero.class
  - java numeros numeros.txt => Ejecuta numeros sobre el fichero de texto numeros.txt

Ejemplo: Contador de naturales

Numeros.flex

```
%%

%class numeros
%standalone
%8bit

%{
  public int cont;
%}

%init{
    cont=0;
%init}
%eof{
    System.out.print("Naturales = "+cont);
%eof}

%%

[0-9]+ { cont++;}
.   {}
```

Ejemplo de JFLEX

Clase generada por Jflex

Numeros.java

```
class numeros
...

  public int cont;

 numeros(java.io.Reader in){
        cont=0;
      …..
}
 yy_do_eof()
{       …..
        System.out.print("Naturales = "+cont);
}
int yylex()
 ...
 <estado final>
                { cont++;}
```


Ejemplo de JFLEX

Clase generada por Jflex

Numeros.java

```
• public static void main(String argv[]) {
•    for (int i = 0; i < argv.length; i++) {
•      numeros scanner = null;
•      try {
•        scanner = new numeros( new java.io.FileReader(argv[i]) );
•      }
•      ……..
• ……..
•      try {
•        while ( !scanner.yy_atEOF ) scanner.yylex();
•      }
•      ……
•      }
```

Código generado al final de la clase causado por la directiva %standalone

Ejercicios propuestos

Construir una especificación en JFLEX para los siguientes problemas:

1.) Sacar por pantalla la suma de todos los múltiplos de 5 de un fichero de texto.

2.) Encontrar y sacar por pantalla todas las palabras de un fichero que contengan a las vocales a y e.

3.) Se tiene un fichero de texto que contiene información de tráfico con la siguiente estructura: <nombre> <puntos_carnet> <matrícula_coche>, donde <nombre> está compuesto por la cadena “Nombre:” seguido de una cadena de letras con espacios en blanco; <puntos_carnet> es un entero y <matrícula_coche> es una cadena con 4 dígitos un guión y 3 letras. Se pide:

a) Pintar por pantalla todas las matrículas de los coches
b) Pintar por pantalla todos los nombres cuyos puntos son 0.
c) Pintar por pantalla todas las matrículas cuyos puntos sean 0.

Ejemplo de fichero:

Nombre: Juan Gomez Puntos: 8 Matrícula: 3495-BSL
Nombre: Pedro Sanchez Puntos: 0 Matrícula: 6549-CAl
Nombre: Ana Garcia Puntos: 12 Matrícula: 4296-AGF
Nombre: Luis Mendoza Puntos: 0 Matrícula: 64542-DCA

Ejercicios propuestos

4.) Dado un fichero con la siguiente estructura: Nombre <cadena que representa al nombre>.... día/mes/año donde se representa información del nombre de una persona y de su fecha de nacimiento (por ejemplo: Nombre Pedro ..... 15/08/80)

a) Sacar por pantalla el siguiente mensaje para cada nombre encontrado,suponiendo que el formato de la fecha tiene 2 dígitos para el día , dos dígitos para el mes y dos dígitos para el año:
1) La fecha de nacimiento de <nombre encontrado>es <fecha encontrada>.
2) <nombre encontrado>nació el día <día encontrado>del mes<mes encontrado>del año <año encontrado>

b) Realizar el problema anterior suponiendo que el formato del día,del mes y del año es variable, es decir, el día y el mes pueden ser uno o dos dígitos y el año puede ser dos o cuatro dígitos.