Universidad Rey Juan Carlos

ETSII Escuela Técnica Superior de Ingeniería Informática

Generación de analizadores léxicos con JFlex

Análisis Léxico

Contenidos

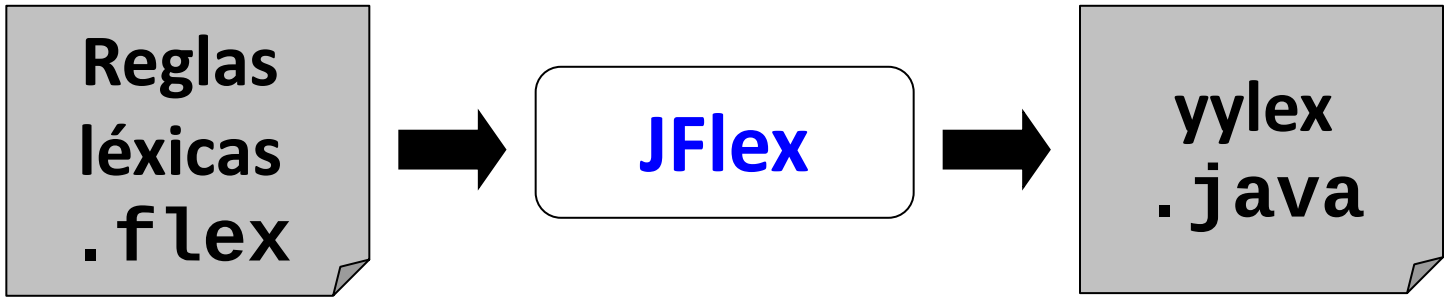
- Introducción
- Especificación léxica
  - Código de usuario
  - Opciones y declaraciones
  - Reglas léxicas
- Funciones en JFlex

Contenidos

- **Introducción**
- Especificación léxica
  - Código de usuario
  - Opciones y declaraciones
  - Reglas léxicas
- Funciones en JFlex

Introducción

- Generador de programas java diseñado para **procesamiento léxico**
- Parte de un conjunto de **reglas léxicas**
- JFlex produce un programa llamado **Yylex** que **reconoce** las **cadenas** que **cumplen** dichas **reglas**



Introducción

- Reglas ≈ Expresiones regulares
- Expresiones regulares → Autómata finito determinista
- *yylex* ≡ **implementación** del A.F.D.
- A cada **regla** se asocian un conjunto de **acciones**
- Cuando *yylex* encuentra una cadena que **cumple** un regla **ejecuta** las acciones asociadas a la regla

Introducción

- Diferentes usos:
  - Transformaciones simples
  - Análisis
  - Estudios estadísticos
- Tamaño de la cadena de entrada → **velocidad**
- Cantidad y complejidad de reglas → **tamaño del autómata**

Pregunta de introducción a JFlex

Selecciona la opción correcta. La aplicación JFlex permite:

1. Construir un programa ejecutable a partir de una especificación léxica indicando lo que se debe ejecutar cuando se encuentre algún patrón de los especificados
2. Construir una especificación léxica a partir de un conjunto de patrones indicando lo que se debe ejecutar cuando se encuentre algún patrón de los especificados
3. Construir un programa java a partir de una especificación léxica indicando lo que se debe ejecutar cuando se encuentre algún patrón de los especificados
4. Construir una especificación ejecutable a partir de un conjunto de patrones indicando lo que se debe ejecutar cuando se encuentre algún patrón de los especificados

Contenidos

- Introducción
- **Especificación léxica**
  - Código de usuario
  - Opciones y declaraciones
  - Reglas léxicas
- Funciones en JFlex

Especificación léxica

Formato general

```
{Código de usuario}

%%

{Opciones y declaraciones}

%%

{Reglas léxicas}
```

Especificación léxica

Formato general

```
{Código de usuario}

%%

{Opciones y declaraciones}

%%

{Reglas léxicas}
```

Especificación léxica

Código de usuario

- El **código auxiliar** necesario para el traductor léxico
- El contenido se **copia tal y como aparece** en la especificación, al **principio** del código fuente generado por JFlex
- Ejemplos: package, import, ...

Especificación léxica

Formato general

```
{Código de usuario}

%%

{Opciones y declaraciones}

%%

{Reglas léxicas}
```

Especificación léxica

Opciones y declaraciones

- Contenido:
  - Opciones del código generado
  - Código fuente específico
  - Macros y estados

Especificación léxica

Opciones y declaraciones

- Opciones del código generado:
  - **`%class nombre`** Especifica el nombre de la **clase generada** como salida de JFlex.
  - **`%line`** y **`%column`** activan **cuenta** de líneas (**`yyline`**) y columnas (**`yycolumn`**) respectivamente.
  - **`%standalone`** Genera **programa** que acepta un **fichero de entrada** en línea de comando.
  - **`%debug`** Durante la ejecución muestra: **nº línea** de la especificación, **lexema** reconocido y **acción** ejecutada.

Especificación léxica

Opciones y declaraciones

- Código fuente **específico**:
  - **`%{ … %}`** Código **copiado tal cual** en la clase
  - **`%init{ … %init}`** Código **copiado tal cual** en el constructor de la clase
  - **`%eofval{ … %eofval}`** Código que se ejecutará cada vez que alcanzamos un **final de fichero**

Especificación léxica

Opciones y declaraciones

- Macros y estados:
  - Macros ≈ **definiciones regulares**

```
DigitoHex = [0-9a-fA-F]
NumeroBinario = “b” [01]+
```

  - Los **estados** condicionan las **reglas léxicas** que se comprueban

```
%state nombre1, nombre2, …
%xstate nombre3, nombre4, …
```

Pregunta de código y opciones de JFlex

Selecciona la opción u opciones **falsas** sobre la siguiente cuestión.
Respecto a la especificación léxica, JFlex permite:

1. Incluir código java extra para integrar el resultado con otras clases
2. Cambiar el nombre de la clase resultante
3. Incluir atributos y/o métodos propios en la clase resultante
4. Especificar código propio que se ejecute cuando el procesamiento del fichero sea exitoso
5. Especificar código propio que se ejecute en el constructor de la clase
6. Especificar código propio que se ejecute cuando se termina el fichero que se está procesando
7. Asegurar la correcta compilación del código java existente en la especificación tanto dentro como fuera de la clase

Especificación léxica

Formato general

```
{Código de usuario}

%%

{Opciones y declaraciones}

%%

{Reglas léxicas}
```

Especificación léxica

Reglas léxicas

- Formato:

```
expresión  { … acciones …}
```

- Funcionamiento:

Cuando se **detecta un lexema** que cumple el **patrón** definido en la expresión se **ejecutan las acciones** asociadas (código java)

Especificación léxica

Reglas léxicas

- ≈ expresiones regulares:

sean a y b expresiones válidas

a | b	Unión
a b	Concatenación
a*	Repetición 0 o N veces
a+	Repetición 1 o N veces (= **a a***)
a?	Opcionalidad
!a	Negación
~a	Cualquier cosa que termine en **a**

Especificación léxica

Reglas léxicas

- ≈ expresiones regulares:

  `{ nombre }` Utilización de una macro

  `"…"` Cadena de caracteres

  `[…]` Clases de caracteres

  `[:letter:]` → Letras

  `.` → Cualquier carácter excepto **`\n`**

  `[^…]` Clase complementaria

Especificación léxica

Reglas léxicas

- Caracteres especiales:

| ( ) { } [ ] < > \ . * + ? ^ $ / . " ~ !

- Utilización de **estados**:

<estado1,estado2,…> expresión {…acciones…}

  - **Activación** de un estado: **yybegin(estado)**
  - **Desactivación**: activación de otro distinto a él.
  - Estado activo por **defecto** **<YYINITIAL>**
  - Las expresiones **sin estado asociado** corresponden al estado **<YYINITIAL>**

Especificación léxica

Reglas léxicas

- Funcionamiento, comprobación de expresiones:
  - Las acciones asociadas a una expresión se ejecutarán si el **lexema concuerda** con la expresión **y**:
    a) **O** la expresión **está asociada a un estado activo**, ya sea inclusivo (**`%state`**) o exclusivo (**`%xstate`**).
    b) **O** la expresión **no está asociada a ningún estado** pero **el estado activo es inclusivo**.

`<YYINITIAL>` es un estado **inclusivo**

Especificación léxica

Reglas léxicas

- Funcionamiento, comprobación de expresiones:
  - **Varias** expresiones posibles: lexema **más largo**
  - **Varias** expresiones posibles con = longitud de lexema: la que se haya **especificado antes**
  - **Sin** expresión posible: se **para** e informa del error, excepto con la opción **`%standalone`**, que **imprime en la salida estándar**

¿qué haría esta especificación?

```
%standalone
%%
```

Pregunta de reglas léxicas en JFlex

Selecciona la opción u opciones **falsas** sobre la siguiente cuestión. Con las reglas léxicas de Jflex se:

1. Puede elegir si una determinada regla se comprueba siempre o sólo en ciertos momentos
2. Debe asegurar que todas son excluyentes entre sí, dos reglas no pueden reconocer el mismo lexema
3. Pueden definir patrones utilizando operadores como los de las expresiones regulares
4. Pueden definir distintas acciones asociadas a una misma regla
5. Puede usar patrones comunes a utilizar dentro de varias reglas distintas

Contenidos

- Introducción
- Especificación léxica
  - Código de usuario
  - Opciones y declaraciones
  - Reglas léxicas
- **Funciones en JFlex**

Funciones de JFlex

`yytext()` Devuelve el **lexema reconocido**

`yylength()` Devuelve el la **longitud** del lexema

`yycharat( int n )` Devuelve el **enésimo carácter** del lexema reconocido

`yypushback( int n )` Considera los **`n`** últimos caracteres del lexema reconocido como **no procesados**. Se volverán a leer, pero **no se tendrán en cuenta** para: **`yytext()`** e **`yylength()`**

Pregunta de funciones en JFlex

¿Tiene sentido ejecutar la siguiente llamada dentro de una acción asociada a una regla?¿por qué?

```
yypushback( 2 * yylength( ) );
```